UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Ingegneria Informatica

User Behaviour Analysis and IoT Secure Applications in Smart City Context

Candidate
Angelo Difino

Supervisor
Prof. Paolo Nesi

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXII, 2016-2019

Università degli Studi di Firenze, Dipartimento di Ingegneria
dell'Informazione (DINFO).

To Yukiko

Acknowledgments

I would like to acknowledge the efforts and input of my supervisor, Prof. Paolo Nesi, and all my colleagues of the Distributed System and Internet Technologies (DISIT) Lab who were of great help during my research.

In particular my thanks go to Pierfrancesco Bellini, Irene Paoli, Claudio Badii and Imad Zaza who collaborated on the main parts of my research work and all the others who were at DISIT Lab when I started my research. Many thanks to Stefano Bilotta, Luciano Alessandro Ipsaro Palesi, Simonetta Ceglia, Gianni Pantaleo, Michela Paolucci, Nicola Mitolo, Daniele Cenni, Mirco Soderi, Ala Arman, Lorenzo Massai, Daniele Bologna.

I would like to thank Yukiko my beloved wife for supporting me all the time.

I would always be grateful to my father Giuseppe Difino and my mother Irene Specogna.

Abstract

This thesis is concerned with Smart Cities.

Smart Cities are urban area that make use of Information and Communication Technologies (ICT) to enhance the quality of their services such as energy, transportation, safety and healthcare. One of the focus of research in Smart City Context is on optimizing resource consumption to reduce the overall costs and enhance the quality of living for their citizens.

The purpose of this thesis is firstly to study and develop algorithms and systems for user behaviour analysis in a Smart City context, in order to realize an User Engagement platform. The implemented tools can be employed by City Operators to understand human behavior, predict common patters, evaluate citizen-related measures and policies and create a communication channel between governors and users to promote virtuous behaviors in respect of the Smart City resources.

The second purpose of this thesis is to study and develop algorithms and systems to implement a secure environment GDPR compliant where applications for Smart City can be designed and executed. User privacy and security are taken strongly into account by design and by default due to the nature of the exchanged data, that can be highly personal and sensitive.

Contents

Chapter 1

Introduction

This thesis describes the PhD activity carried out at the Distributed Data Intelligence and Technology (DISIT) Lab of the Department of Information Engineering (DINFO) at the University of Florence. The work involves area of studies in Computer Science such as Computer Application (Artificial intelligence) and Computer System (Computer security and cryptography, Computer architecture) and it is composed of five parts, each dealing with a specific aspect of the Smart City ecosystem.

With the term Smart City, an urban area can be identified by the use of different types of electronics to manage assets and resources in an efficient way. The advent of the Smart City concepts was facilitated by two important phenomena that emerged during the latest years of the XX century: the spread of urbanization across the western world and the progress in the Information and Communication Technologies (ICT) [82]. Despite the numerous disadvantage associated with urban agglomeration, the world population has been steadily concentrating in cities [68]. Problems related with urban agglomeration have usually been solved by means of creativity, ingenious solutions and cooperation (eventually bargaining) between the relevant stakeholders. In the recent years, the extreme miniaturization of electronics' component [76] and the advancement in wireless broadband networks technologies [114] make realistic the realization of an Internet of Thing (IoT) to address solutions for the challenges that the urbanization pose in terms of mobility, resource and energy management, pollution, resilience, disaster recovery, etc. [151].

The Smart City concepts are usually implemented on top of complex in-

terrelations between the Smart City components and actors [141] and require the knowledge of different heterogeneous aspects. In any case, the main focus is on the importance of the new ICT regarding the building of modern infrastructures within urban area, even if expertises in social sciences can give an advantage. Concepts of smart governance [205] and citizen participation are essential to promote the new set of technologies' paradigms: researchers aim at understanding human behavior in order to predict common patterns and evaluate citizen-related measures and policies [64]. Citizen participation is the process which provides private individuals an opportunity to influence public decision [196]: it implies to establish a communication channel between governors and citizens. DISIT Lab participates in Sii-Mobility, a Smart City national founding project supported by the MIUR, where the context of application is on mobility management. It strongly focuses on improving sustainable mobility [103] and propose a efficient use of the city's services and of the environment's richness. In this context, the goal of the thesis is to produce a better quality of life for the citizens, in term of less pollutions, less social costs and better living condition. A user engagement system is essentials to stimulate virtuoso behaviors, targeting specific part of the population to promote strategies defined by governors [66].

The chapter 2 of this thesis describes an original solution for city users engagement. The proposed integrated system enables the stimulation of the city users towards taking sustainable behaviors, in mobility and energy, through the formalization of strategies via a simple and flexible language that can be easily adopted by city operators. The solution makes use of predictive models, assessing user behavior via data analytic (semantic computing, machine learning), deriving contextual information, detecting city users changes of context, providing engagements, collecting feedbacks and follow ups.

The chapter 3 presents a novel approach for user's transportation modality classification. The proposed automated system permits to understand whether an individual is stationary, on the move, walking, on a motorized private or public transport, with the aim of delivering to users personalized assistance messages to promote for example a sustainable mobility or healthy activities. The approach has been designed to provide results and thus collect metrics in real operating conditions (imposed on the mobile devices as: a range of different terminal kinds, operating system constraints managing applications, battery consumption manager, etc.).

In order to facilitate the interoperability between components in a Smart City ecosystem, an IoT platform for Smart City has to be realized [147]. It has to be able to receive and manage huge amount of information generated by heterogeneous data sources [161], thus entering into the Big Data domain. The DISIT Lab implemented his own solution (Snap4City) in response of a research challenge launched by Select4Cities PCP (Pre Commercial Procurement) H2020 research and development project of the European Commission (`https://www.select4cities.eu`).

The chapter 4 illustrates the innovative Snap4City architecture, that permits to rethink the way the Smart City infrastructures are designed, managed and used. The Snap4city solution allows to setup heterogeneous and complex scenarios that integrate sensors and actuators as depicted in IoT architecture in an overall scenario of Big Data, Machine Learning and Data Analytics. A detailed and complex case-study has been presented to validate the solution. This case study composes several building blocks of the IoT platform, which demonstrate that a flexible and dynamic set-up is possible, supporting off-grid, cloud and mixed solutions.

The chapter 5 describes several aspects of the Snap4City platform regarding its privacy and security support respecting the GDPR of European Commission. Due to the potentially very sensible nature of the managed data in a Smart City platform based on a IoT architecture, privacy and security aspects have been taken into account by design and by default. In addition, an end-to-end secure solution was implemented to guarantee a safe environment to final users for their personal data, in transit and storage, which have to remain under their full control. The system has been validated via two exhaustive penetration tests performed by leading industries with security and privacy expertise.

Finally, the chapter 6 presents a suite of MicroServices for the Snap4City platform that enable the creation of a wide range of IoT Applications for Smart Cities. They involve the use of several component present in the IoT architecture, like Dashboards, IoT Devices, Data Analytic, IoT Discovery, etc. A complete list of requirements that focus on IoT Application in Smart City context has been collected and the solution proposed has been validated against a large number of IoT Applications for the cities of Florence, Antwerp and Helsinki. In addition, three applications have been presented for validation, putting in evidence the relationships among MicroServices, Dashboards and Data.

Chapter 2

User Engagement Engine for Smart City Strategies

This chapter describes an original solution for city users engagement. The proposed integrated system enables the stimulation of the city users towards taking sustainable behaviors, in mobility and energy, through the formalization of strategies via a simple and flexible language that can be easily adopted by city operators. The solution makes use of predictive models, assessing user behavior via data analytic (semantic computing, machine learning), deriving contextual information, detecting city users changes of context, providing engagements, collecting feedbacks and follow ups. [1] [2]

2.1 Introduction

Most of the Smart City solutions are focused on providing services and information to the city users about transport, energy, tourism, culture, etc. The simpler and faster approach for the entry level Smart City implementa-

[1] Part of the work presented in this chapter has been published as "User engagement engine for smart city strategies" in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)* [50].

[2] *Acknowledgments:* this work was partially supported by the MIUR, with the Smart City national founding project Sii-Mobility SCN_00112, the University of Florence and the companies involved for co-founding Sii-Mobility

tion could be to expose open data portal (e.g., the CKAN data management system [9]) with the hope of having someone producing services exploiting them. The further step consists in providing Smart City API to access at single data set or data aggregations. For example: CitySDK makes transformations on each dataset in order to obtain and manage uniform data [8]; Transport.API provides aggregated open data in the UK about transport information [38]; Navitia.io is an open source project exploiting OpenDataSoft for open data aggregation in France [29]; Km4City Smart City API provides semantically integrated information for app and services [152] with its API [152]. The successive step in Smart City implementation consists in deeply understand the city usage and thus the user behavior [153]. The analysis of the city users' behavior, on the basis of the knowledge of the city infrastructure and context, should allow identifying possible directions for the city improvements. Once the objective of city improvement is identified, a suitable strategy has to be put in place and persecuted to change city user attitudes. Thus, strategies for city improvement may be defined to a more sustainable exploitation of city services by the city users. For example, strategies to reduce the number vehicles in certain areas, increase the usage of public transport in certain time slots, move tourists in diversifying paths in the city visit, go also on minor museums, consume less energy in certain time slots, select less busy parking areas, etc.

On this regard, the first step consists in being capable to identify the right users that should change their habits in using the city, reach them with convincing stimuli that may change their habits (especially for citizens and commuters and less relevant for other city user kinds). Well known solutions and techniques produce suggestions and recommendations guessing "what the user would like", by similarity [159], by indexing [60], by filtering, etc., may be according to the advertiser strategies. The engagement should be more capable to understand the story of the user behavior, and predicting possible new future attitudes [69] according to the city strategies. The context of Smart City is much more complex than the simple media advertising since spatial reasoning is added to the classical temporal reasoning [112]. Sensors and mobile phones are capable to monitor the context in which the city users move around and substantially can instrument city users for engagement [157]. Level of engagement, point of engagement, disengagement and re-engagement are some of the phases in which the user is involved during the day. In literature, several methods analyze and evaluate

users' participation, measure the focus attention, the endurability and the novelty [45] as well as retention rate, click depth, popularity [136] has been deeply studied to evaluate how good an application is on taking care of the user's experience; and for gamification techniques for a more enjoyable and smooth experience [144].

In this chapter, an integrated solution to pass from strategies to stimuli at the city users via mobile phone is described. The solution is grounded on assessing user behavior, deriving contextual information, computing conditions of users, detecting city users changes, providing engagements, collecting feedbacks and follow ups to perform analysis. The whole approach has been developed and enforced into Sii-Mobility Smarty City National project on mobility and transport of the MIUR (Italian Ministry of University and Research) `http://www.sii-mobility.org`, with its Mobile Application (Mobile App) and Smart City infrastructure based on the Km4City ontological model `http://www.km4city.org` [60] and Smart City API [152].

This chapter is organized as follows. In Section 2.2, a brief survey of related work on user engagement system is presented. Section 2.3 report the main requirements and architecture. Section 2.4 presents the User Engagement Engine providing details about conditions, user behavior data analytics, actions and feedback rules. In Section 2.5, implementation notes are reported. Section 2.6 reports the experimental results putting in evidence the effectiveness of the solution in understanding the user behavior, with trajectories and precision in providing specific engagement rules to city users. Considerations are presented in Section 2.7.

2.2 Related works

Historically, the first engagement systems focussed on the productivity of employees (work engagement): the engagement could be defined as a state including vigour, dedication and absorption [188]. Engaged workers are more creative, more productive and more willing to go the extra mile. They have high levels of energy and are enthusiastic about their work [54]. Moreover, they are often fully immersed in their work so that "time flies" [145]. Subsequently, similar techniques were applied to the educational field of studies [104]. Today, these approaches can be applied to improve the user experience in accessing online services. The quality of the interaction and the level of participation is essential for service that offer social civic service, since the

participation can be translated in the real world after an interaction with online service is consumed. In the following subsections we present the most relevant techniques used by traditional user engagement system.

2.2.1 Gamification

During the recent years, gamification has been a trending topics [87] and a subject to much hype as a mean of supporting user engagement [119] and enhancing positive patters in service use, such as increasing user activity, social interaction or quality and productivity of actions [119]. These desired use patters are considered to emerge as a result of positive intrinsically motivating [180] and joyful experience [125] brought out motivational affordance implemented in a service. As result, gamification is touted as a next generation method for marketing and customer engagement in popular discussion [218].

Gamification has been defined [118] as a process of enhancing services with motivational affordance in order to invoke joyful experience and further behavioural outcomes [117]. The [108] try to properly define a model based on the gamification theory: firstly, the goal is to design an instructional program that incorporate certain features or characteristic of a games. Secondly, these features trigger a cycle that include user judgment or reaction such as enjoyment or interest user behaviours such as greater persistence or time on task and further system feedback. To the extend that we are successful in pairing instructional content with appropriate game feature, this cycle results in recurring and self-motivated game play. Finally, this engagement in gamer play leads to the achievement of training objectives and specific learning outcomes.

In a very passionate manuscript [79], Yu-kai Chou report in detail his experience in gaming and try to review any different previous characteristics in a common platform, meanwhile [150] confirm that gamification's main goal is to rise the engagement of users by using game-like techniques such as scoreboards and personalized fast feedback [105] making people feel more ownership and purpose when engaging with tasks [163].

2.2.2 Flow theory

The flow theory focus on the state of optimal experience, thinking to generate happiness, and is defined as "the state in which people are so involved in an

activity that nothing else seems matter; the experience itself is so enjoyable that people will do it even at great cost, for the sheer sake of doing it" [84].

Flow theory move forward the idea that certain factors add to the enjoyment of an experience [209], and the more of these elements are present, the more enjoyable, engaging and immersive the experience is. The most important points of the theory are the following:

- a challenge that requires skill to achieve with an attainable goal and known rules;
- complete absorption in the activity and concentration in the task in hand;
- clear goals;
- immediate feedback;
- loss of self-consciousness and transformation of time.

The flow theory is useful as starting point for considering the nature of engagement, but not the measurable level of indicators. In his analysis of flow theory [95] add that engagement only occurs when a connection exist between the activity and the player's core values and beliefs, while Salen and Zimmerman [183] argue that flow is not intrinsic in a game, but it depends on the state of mind of the players as they play the game. The engagement is considered to be as a subjective state observable only on the individual taking part in the activity, so that it is not the activity itself that is engaging but rather an individual's interaction with the activity at a specific time and in a specific way. Being in a state of "flow" is considered to be a similar state of being highly engagement.

2.2.3 Analytic hierarchy process

The analytic hierarchy process (AHP) was proposed at the end of the 1970 by Thomas Saaty [181] and further developed nowadays. It is a mathematical method that allows analysing complex decision with multiple attribute and for establish priorities in decision making. AHP has been used in different fields to evaluate/compare/rank different planning options. In [85] and [126] a proposal to use such technologies to address the problem of public engagement was proposed. The method, in general, can assess and prioritise

actions, such as program, intervention strategies and goal. Thus, it can be used to support rational decision, to mallee the best decision for different targets. In their proposal, the AHP addresses most of the characteristic that public engagement should have:

- it can be used to engage;
- it provides quantitative input for planning process;
- it is easy to update;
- it may be calibrated independently of planning option.

The field of application used to demonstrate the validity of this approach is towards the involvement in strategic transport planning process. The public should be engaged and should give their feedback from the beginning of the planning and through the entire process. Public engagement is seen as involvement in the decision making with the purpose of influencing the choices being made [158].

2.2.4 Customer engagement behavior

Since 2000, customer management research has evolved and produced a significant impact on the marketing discipline [204]. In a reassuringly networked society where customers can interact easily each other even directly with big firms through social network and other media, the proposal for customer engagement became an important asset. Customer engagement is considered as a behavioural manifestation toward the brand of the firm that goes beyond the transactions. Several models have been proposed to describe the process of conceptualization of customer engagement. In [202] a strong differentiation between antecedents and consequences is made. For antecedents two types are examined: customer base firm base and context base. For consequences a model for customer, firm and others is considered. In the middle, a engine that evaluate the antecedents to produce the consequences is considered. A different approach is made in [206] where a connected graph is depicted to model the customer engagement process, where also customer participation and involved are antecedent and value, trust, affective commitment, loyalty and brand community involvement as consequence.

2.2.5 Civic engagement

Many theorists have long extolled the virtues of public deliberation as a crucial component of a responsive and responsible democracy. Building on these theories, in recent years practitioners (from government officials to citizen groups, no-profits, and foundations) have increasingly devoted time and resources to strengthening citizen engagement through deliberative forums. Although empirical research has lagged behind theory and practice, a body of literature has emerged that tests the presumed individual and collective benefits of public discourse on citizen engagement [70]. This involvement is called civic engagement and has attracted a new field of research on heterogeneous scenarios. In [66] the reinterpretation of the role of policy making and service to the public domain is taken into account. The management of the services are no longer just the preserve of professionals and managers, but also user and other member of the community are playing a large role in shaping decision and outcomes. A conceptual framework is developed to allow detailed characterization of the relationship between user and communities and professionalized public service. Meanwhile in [142] the concept of civic engagement is deployed in a real case scenario: the public engagement is view as a means for involving the public in decision making about science issues [123]. Here the term "engagement" means as a personal state of connection with the specific issue, i.e. the climate change, in contrast to engagement solely as a process of public participation in policy making.

2.3 Requirements and architecture

Most of the Smart City solutions for sustainable Smart City are focused on implementing specific strategies and cases: for sustainable mobility [56], energy, etc. To enable the definition and the implementation of strategies we propose an engagement engine to keep track of the user evolution and context, providing mechanism of reward and gamification (flow experience), promoting virtuous activities and discouraging other ones. The aim of the Sii-Mobility User Engagement Engine (UEEngine) is to cover a range of cases by proposing a flexible language to formalize strategies and to pass from them to actual implementation of mobile platform to stimulate changes into city users. The main requirements of an engagement solution for sustainable mobility would be as follow. Thus, an UEEngine should be capable to:

- **assess the user behaviour** (evolution and context) according to its status and evolution in space and time: his personal points of interest, trips, modality of moving, interaction with the city infrastructure, etc. The assessment should be as much as close to real-time and, if possible, be available even if the user is not using actively the platform (background service);
- **identify the city users' habits** which may match with the behavior to be corrected according to the city strategies. Collective city users' habits are the recurrent activities that, if not sustainable, may be worth to change thus creating a relevant and tangible benefit for the society;
- **provide a sort of programming visual language** to formalize the strategies to stimulate the city users in changing habits. The formal language has to be: (i) easy understandable by city operators, (ii) expressive and flexible to allow the formalization of a wide range of use cases, (iii) track the evolution of engagement discourse established with the city users, (iv) been able to generate different kind of stimulus, information, guidance/ help, requests, bonus, surveys;
- **provide tools** that enable (a) City Operator to **monitoring** their defined strategies (how many times a strategy has been triggered, which percentage of people replied to a stimuli) (b) final user to **browse and enrich** their profile, checking how their behavior is in-line with the proposed City Operator (c) **assess** the city condition before and after the defined campaign, to provide feedback for example to marketing advertisements.

The main architecture of the UEEngine is highlighted in Figure 2.1. From the left side, the City Operator define Strategies by using an Engagement Rule Editor in which also define eventual bonus, tickets to be spent on strategies according to some relevant evidence (for example, a pay per: click, attitude chance, wrong attitude detection, etc.). Some of the Strategies can Reward the city users when they change behavior, or just stimulate with bonus to change the habits (e.g., we offer you a free ticket for the parking area if you park outside instead of going down-town). To support the definition of this Rewards (Feedback's rules), the UEEngine rely on a digital Wallet, a software realized internally ad-hoc for a generic engagement purpose, to distribute digital points to users, upon observation on their activity, and to distribute goods or prize in exchange of the collected digital points.

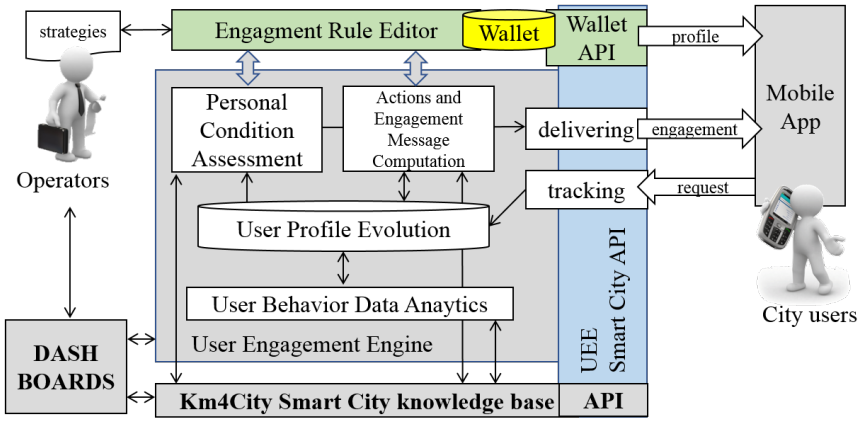


Figure 2.1: Sii-Mobility User Engagement Architecture

The defined Strategies are automatically translated in Conditions to be assessed/detected and Actions propagated to the final City Users based on (a) User Profile Evolution directly observed from the Mobile App, (b) User Behavior Data Analytics and (c) the city context which can be directly accessible on Km4City [152]. The combination of Conditions and Actions (Engagement's rules) are defined using the Drools Workbench, the web User Interface of the Drools tools-suite [11].

Therefore, data about changes in the City Users status are collected continuously, and the system receives requests from the Mobile App periodically. In some cases, the UEEngine may have elaborated some Actions according to the detected changes, and thus may be ready to provide new short or long-term engagements. The list of the Engagements sent to the users are recorded for future analysis, avoid replications, and matching the follow up, etc. If the user really read/consume an Engagement, it will be labelled as Viewed. Thus, the User Behavior Data Analytics may assess the follow up, as well as compute eventual bonus, etc. The City User may be interested to verify the status of his/her Wallet via specific API.

The UEEngine keeps an updated version of the user's context and, on top of this, monitors which Conditions (specified in the Engagement rules' set) are valid. Whenever all the Conditions specified in an Engagement rule became true, the platform prepare and delivery the specified Action (provide

an Engagement to the user). At the same time, the platform monitor if/when the delivered Engagement is really carried out (i.e. if a request of survey is sent to the user, the platform checks if the survey has been viewed, filled and submitted) and eventually update the context according the information specified in the Feedback's rules.

With a set of Dashboards, the City Operator can assess the status of the UEEngine and check whenever the Strategies are correctly delivered by the system and receipt by the citizens. Eventually can refine the Strategies to enlarge or narrow the user's target. The user, via the Mobile App or via an ad-hoc web portal can browse his profile and eventually update, review the Rewards he collected and check the scoreboard of the most virtuous users (for privacy issue, just his points are presented and also the best virtuoso user's points, even if in an anonymized way).

2.4 User Engagement Engine

UEEngine takes in charge the Strategies which are translated into two kinds of Rules:

- Engagement Rules: the combinations of the Conditions with the Actions;
- Feedback Rules: the kind of User Profile Evolution updates that may occurs in case of engagement/disengagement directly computed by the User Behavior Data Analytic.

A Strategy is converted in one or more Engagement Rules [124] in the form of:

$$IF{<}set{-}of{-}firing{-}events{>}AND{<}set{-}of{-}conditions{>}THEN{<}action{>}$$

Where: each $<firing-events>$ (at least one) is a special formula $F(C_{ts,te}(u))$ that checks (on the User Profile Evolution database) if some context's characteristics of the user u assume a certain value in a specific time range $[ts, te]$. They are the firing conditions to evaluate the $<conditions>$ (Personal Condition Assessment) part of the rule - e.g., the arrival in a place, the answer to a question. The firing events and conditions have to be verified for each user with a sufficient frequency in order to be ready to produce the

actions, stimulating the city user in the right moment, not too far from the occurrence of the firing events. The <*actions*> may be more or less convincing suggestions and, in some case, may provide a value, such as a bonus, a discount, an event ticket, etc.

In the following subsections, Conditions, User Behavior Data Analytics, Actions, and Feedback Rules are better described and formalized. Please note that Actions issue Engagement Messages to users.

2.4.1 Personal condition assessment

A Condition $C(u,t)$ is the conjunction of a set of contextual characteristics $C_1(u,t), C_2(u,t), ...C_n(u,t)$, which represent the characteristics $1, 2, ...n$ of the User Profile Evolution in which each user u may be at a certain time instant t:

$$C(u,t) = C_1(u,t), C_2(u,t), ...C_n(u,t)$$

The set of characteristics can be potentially infinite and enormously detailed. It can be subdivided in six macro groups as:

- **User kind**, i.e. citizen, tourist, commuter, operator, student;
- **User info**, i.e. gender, age, nationality, spoken languages, special needs, velocity, mean of traveling, acceleration;
- **Place/location**, i.e. Global Positioning System (GPS) coordinates, municipality, region, nation, terrain topology, area description, population of the area.;
- **Date/Time**, i.e. day, date time, day of the week, holiday day, season, month, year, temporal window (from-to date);
- **Environment/sensors**, i.e. weather condition, level of water in a river, pollution, temperature;
- **City context**, as closest to (i.e. a Point Of Interest 'POI' with its description, street, cycling path, railway, highway) or inclusion into (i.e. parking, fuel station, green area).

The values assumed by a $C_i(u,t)$ can be represented as a number, free text, date/time/duration, enumerates, as well as complex structured data

values. The sequence of values $C_{ts,te}(u) = C(u,ts), C(u,ts+1), ...C(u,te)$ describes the evolution of each Conditions $C(u)$ in an interval of time $[ts,te]$ and can be visualized as in Figure 2.2(a)(b). Moreover, the set of User Profile Evolution include also similar descriptors for collective profiles and for the typical User Profile Evolution. Thus, a Markov Chain of probable status with related transition is constructed among possible, $C_{ts,te}(u)$, representing data from the past, and possible predicted status for the future and transitions [111] [156]. In Figure 2.2(c) a simple example of this scenario is presented highlighting, with numbers on transition exiting from a status, the probability of a user to mode from that particular status towards a new one.

2.4.2 User behavior data anaytics

Many of the above-mentioned characteristics $C_{ts,te}(u)$ can be directly obtained from the mobile phone data. Others can be computed from the contextual data by the User Behavior Data Analytics. For example, the GPS coordinate may provide information about the region and nation in where the user is located in, the temperature for that location may be known, as well as the weather status. In general, a set of processes/formulas $F(C_{ts,te}(u))$ may be set up and computed by using:

- Semantic computing: SPARQL Protocol and RDF Query Language (SPARQL) queries exploiting inference on the Km4City knowledge base by exploiting spatial and temporal reasoning. For example, to know the closest POI, the inclusion in an area, the weather, the next buses arriving at the bus stop, the entertainment events in the city in a given time slot, etc. [152];

- Data analytics: machine learning, clustering and/or statistical approaches based on collected characteristics in the time interval as well as in the past. For example, by creating algorithms capable of computing/predicting:

  - the means of traveling (car, bike, bus, etc.) can be derived from the velocity, acceleration, proximity to bus line path, etc. ;

  - the density of people, thus minimizing/maximizing the probability of meeting other people [153];

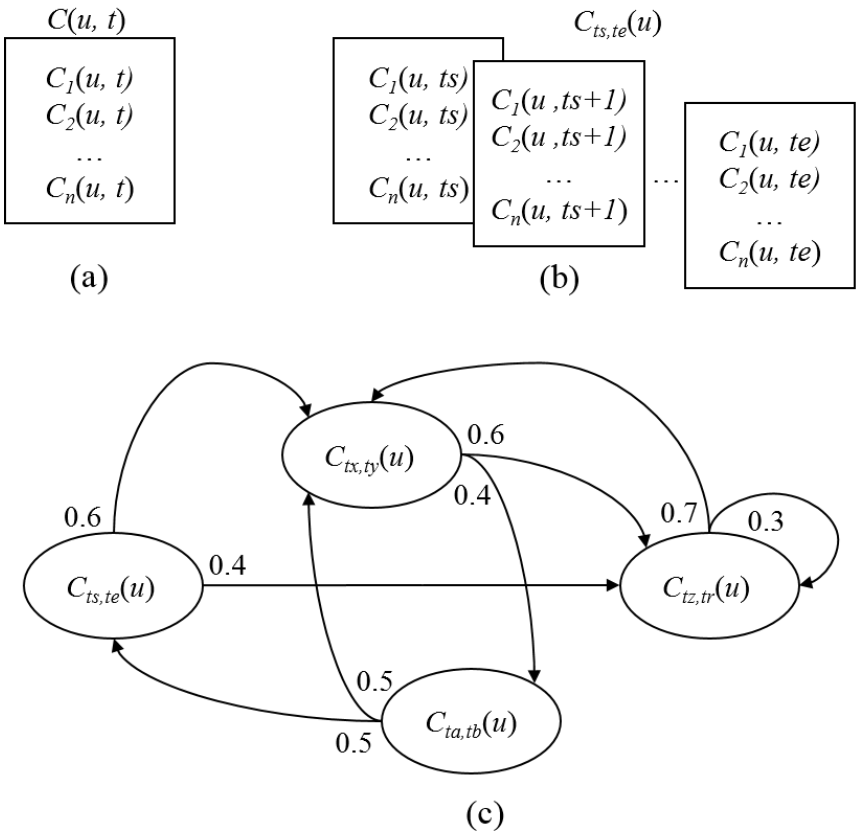


Figure 2.2: User Profile Evolution: (a) simple condition in a specific time t; (b) condition evolution in an interval of time $[ts, te]$; (c) example of transition between conditions

- the traffic flow and/or parking in other areas may be known, while may be predicted for the future, etc. ;
- the next possible target location for each given user according to his/her personal story.

The evaluation of a generic $F(C_{ir}(x)(u))$ may be time consuming. For most of the feature, it can be computed offline for many different users at

the same time and storing the results in common storage and knowledge base, or in the User Behavior Evolution if the value is specifically associated with that user. In Sii-Mobility, most of the computational activities assessing the Conditions are scheduled as periodic background processes and managed by the Distributed Smart City Engine Scheduler (DISCES) [49]. Strategies of caching for the different user's characteristics and automatic refresh in background are taken in consideration to optimize the access of such data.

2.4.3 Actions and Engagement Messages computation

The Actions a rule can specify are rich messages delivered to the user via a Mobile App and can be classified as Inform, to present some engagement or assistance messages, i.e. alert of traffic disruption, parking availability in the area; Ask, to request a specific action to make, i.e. reply to a survey, provide a comment. Usually the Actions are enriched by a valid time interval a set of rewards that can be earned if it's carried out during its validity time. An Action A is defined as a box container for an Engagement Message (EM), ruled and enriched by a list of action's characteristics $A_1, A_2, ...A_m$, representing the modality of delivery of EM for a given user u:

$$A(u) = A_1, A_2, ...A_m, EM(u)$$

Example of Action's characteristics can be the sending rate of a particular type of engagement or the maximum number of engagements to be delivered simultaneously, or provided in a time frame, according to strategies.

The UEEngine takes care of the management of the pool of Actions prepared to be sent (fired rules upon Condition's validation) and to deliver the chosen ones according the specified modality (i.e., to avoid user flooding and to fine tuning, the orchestrator can decide that some type of engagement should be delivered just every hour). The scenario is sketched in the Figure 2.3.

The computation of an Engagement Message is performed based on the Strategy defined by the Operator by putting together several different elements of information to be delivered to the Mobile App, depending on the Engagement kind. An Engagement Message EM(u) is defined as a list of elements $EM_1, EM_2, ...EM_n$, delivered to the user's u mobile phone:

$$EM(u) = EM_1, EM_2, ...EM_n$$

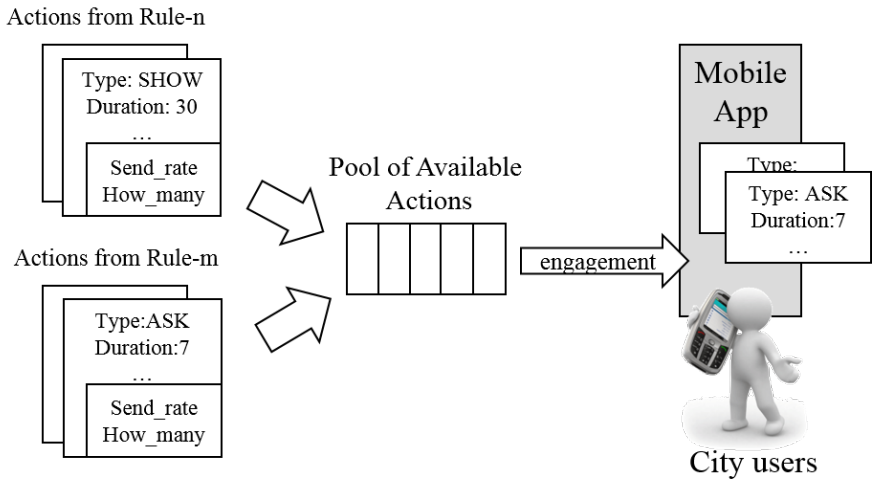


Figure 2.3: Actions pooling for engagement delivery

The elements of an Engagement Message include:

- the type of the Engagement:
  - SHOW: to inform the user by showing - e.g., alert of civil protection, weather forecast, if you parked here please remind to take the ticket;
  - ASK: to request an action to the user - e.g., take a photo, provide a comment;
  - CHECK: to ask confirming an analysis - e.g., 10 minutes ago you moved via: car, bus, bike;
- target destination of the engagement: it may be user via Mobile App interface, or could be sensor/display in the user's proximity;
- deadline: time duration of the engagement into the Personal Assistant page and notification in the Mobile App. It is usually expressed as a time out over that the message is removed;
- additional data: details for the action.

2.4.4 Feedback rules definition

A set of feedback rules are initially defined by the platform to take care the explicitly update of the user's profile with the observed fact, to enable future analysis and discovery user's behavior with similar pattern. A set of rewarding rules can also be defined by the City Operator using the Wallet user interface to provide the user a flow experience (feedback loops) in term of rewarding schemas, incentives (i.e., providing discount, offer bonus, manage fidelity card) and gaming experience (i.e., compile a leader board, promote challenges). They can be classified in 4 different types:

- Engagement: involve the user to get micro-contribution: take a photo, submit a rating, submit context's information. After a number of this action the user can be reward with incentives;
- Assistant: basing on the user's behaviors and context some stimulus (suggestions) are delivered towards virtuous behavior, sustainable and healthy. For example, push user to use more public transport, save time and gasoline, improve the personal healthy;
- Promotion: commercial stimulus and strategies for the Smart City are proposed, not directly connect to the main goal (i.e. public mobility), but focused on side issues, like tourism distribution and flow or to support the inclusion of people with disadvantage or difficulties;
- Incentive: whenever the execution of the Actions is carried out, the system rewards the user with virtual points, bonus, service's discount for having taken an active rule in the system. Virtual Point can be used to present a scoreboard of the most virtuous users and can enable the user to request more and more prizes whenever some thresholds are reached. The strategies define these thresholds and the prizes distribution (amount/rate).

The Feedback Rules involve the definition of strategies that strongly depend on the target scenario of the integration. The engine, offers a pluggable runtime system for defining monitoring tools (one for any type of engagement), that analyses the user behavior and upon the observation that a user has been actually engaged in some kind of activities (after an engagement has been received and properly consumed) thrown a signal that can be catch by a listener that implement a feedback rule. The listener can specify a

timeout, as the maximum amount of time to wait until the engagement is feedback.

2.5 Implementation notes

The UEEngine has been realized in the context of the Sii-Mobility project and thus integrated in the general framework which includes the Km4City knowledge base and tools. In the context of that project UEEngine has been integrated according to Figure 2.1, and providing specific Smart City API towards the Mobile App. Thus a couple of mobile applications have been implemented and are available on Google Play Store, Apple Store, Windows Market, and Windows 10, named as "Firenze, where, what,..", "Toscana, where, what,..", and supporting 5 different languages (ITA, ENG, ES, DE, FR). In the Figure 2.4 some screenshots of the user engagement interface are presented.

The UEEngine has been realized by using JAVA and using MVFLEX Expression Language (MVEL) [25] and/or the Friendly Enough Expression Language (FEEL) [13]. The MVEL is a hybrid dynamic/statically typed, embeddable Expression Language and runtime for the Java Platform, meanwhile FEEL is the description language promoted by the BPMN consortium. Both of them are currently supported in several platforms and stable enough to have some graphical presentation (DROOLS workbench) [11]. The Drools' box tool has been used to execute the Engagement Rules and their triggering.

A FEEL expression is a formula that produces a value. FEEL expressions can reference variables, but unlike a programming language, it cannot create variables. FEEL defines just a few datatypes: string, boolean, number, and a few date, time, and duration types. These types can be restricted to enumerated values or ranges, and can be combined to form complex data structures. FEEL variables may include lists and tables, as well. FEEL provides a large number of built-in functions and operators. For example,

$$\text{Order.Item}$$

$$price >= 10$$

means from a table Order, select all rows (Item) with a price greater than or equal to 10. In addition to table queries and joins, FEEL expressions can do iteration, datatype validation, list sorting, and lots of other powerful things. FEEL expressions can invoke reusable functions, like

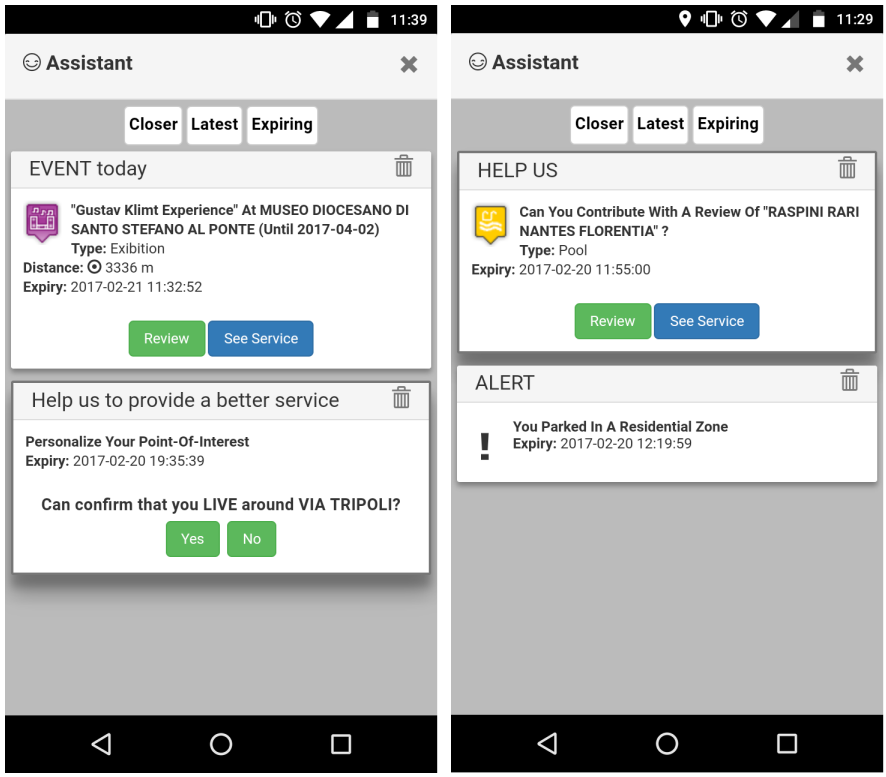


Figure 2.4: User Engagement on Mobile App

paymentAmt(principalAmt, ratePct, termMonths)

What that means is that instead of handing off iteration to BPMN and handing off queries to SQL, DMN logic can do the end-to-end decision logic itself.

When there is one or more Rule Match on the Agenda (list of available rules) they are said to be in conflict, and a conflict resolution strategy is used to determine the order of execution. The Drools strategy is very simple and based around a salience value, which assigns a priority to a rule. Each rule has a default value of 0, the higher the value, the higher the priority.

As a general rule, it is a good idea not to count on rules firing in any particular order, and to author the rules without worrying about a ”flow”.

However when a flow is needed a number of possibilities exist beyond salience: agenda groups, rule flow groups, activation groups and control/semaphore facts.

2.6 Experimental results and validation

The UEEngine has been used to stimulate the city users to:

- have a more sustainable mobility;
- contribute at the information collected on the knowledge base;
- improve the mobile application;
- have more attendance on events.

Specific User Behavior Data Analytic processes have been implemented to assess:

- closeness of a point: over bus line, train line, highway, cycle paths, etc.; inside areas: parking, parks, etc.;
- user mobility mode: analyzing the user's device context characteristics (speed, acceleration, GPS, ...) and the transport public network, determinate if the user is: still, walking, running, driving, biking, using the motorbike/tram/bus/train;
- change of mobility mode: deriving the dynamic evolution of the user mobility mode and the transport public network determinate if the user is: parking the car, taking the car, taking the bus, etc.;
- identify the Personal POI, PPOI: analyzing the user mobility mode in specific day-slots, determinate the GPS coordinates of the user's home, the user's place of work or other PPOIs [43];
- Identify the habits for a number of users in moving among their PPOI [217]: work-work, home-work, work-PPOI, etc.

A number of engagements to push to the city users have been formalized are:

- ask the user to fill a survey (one-shot question, detailed survey);

- ask the user to comment a poi (take a photo, comment, rate);
- notify the user a message (an alert, a description message);
- notify the user an event (its coordinates, its duration).

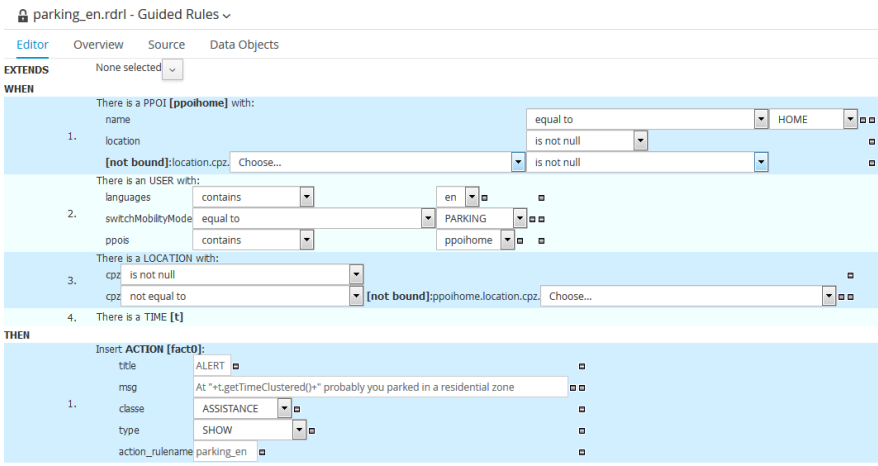
With these lists of conditions and actions a total of 45 rules have been put in place. For example, in Figure 2.5, a rule to SHOW an ALERT in English on the device when the UEEngine detects that the city user is parking in a controlling parking zone different from that of his residence is presented, by using MVEL language and on Drools workbench tool.

The UEEngine, with the above-mentioned rules is up and running for validation since the 1st of September 2016. It has been able to produce 411.177 engagements on a total number of 11480 users (belonging to different categories, citizens, tourists, etc.). About 62.000 daily users' profiles have been analyzed.

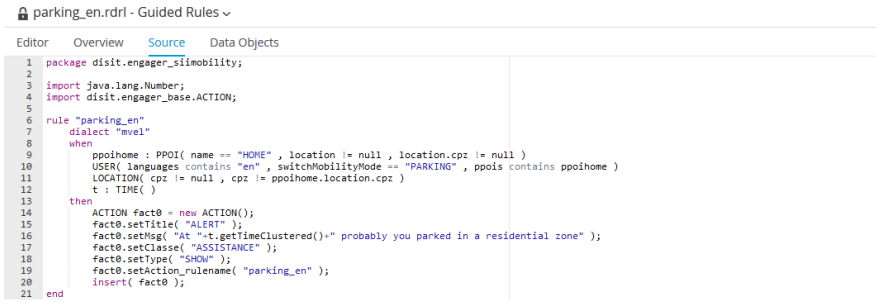
The execution of the User Behavior Data Analytics on the citizens detected the HOME PPOI locations (in which the city user pass at night several hours) for the 89.43% of users, the WORK PPOI location for the 65.21% of users. The tracked temporal window was set at 8 weeks and thus, users with less than 20 recurrent positions have been discarded as well as locations with low GPS accuracy less than 100m. The execution time for the whole analytics was put in execution every 6 hours to keep update the lists of PPOIs. During the validation test with a restricted number of 15 users from the lab, the percentage of precisely classified PPOI as HOME or WORK has been of the 93.4%.

Moreover, in the temporal windows from 2016-12-12 to 2017-2-6 (8 weeks) the Mobile App have been put in the hands of the final users detecting a total of 344 Personal POIs on 200 users. Among them, 182 PPOIs have been classified as HOME or WORK. Plus, 162 PPOIs on 109 city users have been identified having different classification with respect to HOME or WORK. The Figure 2.6 presents the distribution of number of city users with additional POIs with respect to those classified as HOME or WORK, identified in the same timeslot.

In the same 8 weeks, it was possible to compute the most frequent trajectories of the users between their PPOIs has been analyzed. A total of 3588 trajectories has been identified for 84 different users. A clustering based on the day of the week and the slot of the day (night, morning, launch, dinner, evening) has been applied. Among the above mentioned trajectories 1799



(a)



(b)

Figure 2.5: Example of Engagement Rule: (a) visual representation on Drools workbench tool (b) MVEL transcription

presents a number of repetitions as reported in the Figure 2.7. These trajectories can be classified according to their relation with HOME, WORK or X (other PPOIs) as reported in Figure 2.8.

All these 3588 trajectories have been used to construct a Hidden Markov chain for each user: the probability of a user move from a PPOIs to the next, in the weekday and time slot the observation. The prediction model was

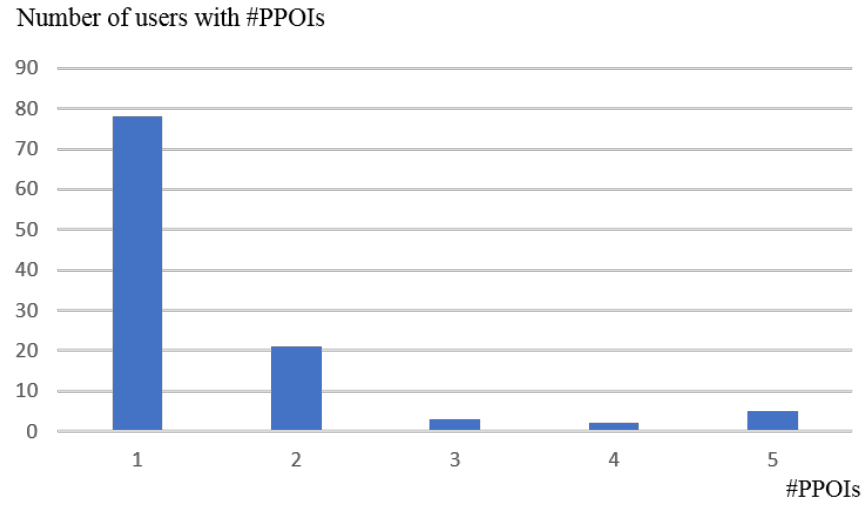


Figure 2.6: Distribution of number of city users with additional PPOIs with respect to those classified as HOME or WORK

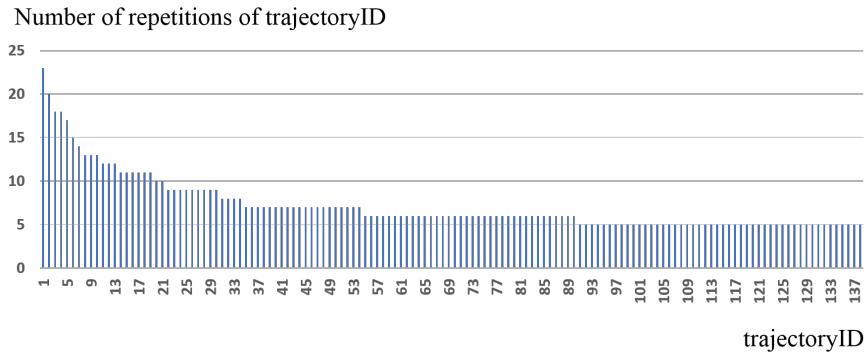


Figure 2.7: Distribution of repetition for trajectories IDs

tested in the timeslot [7-2-2017, 14-2-2017]. During the validation predictions have been delivered only when the Markov probability was above the 65% (same days, dayslot, starting PPOI, etc.), and the prediction was delivered

within 3 minutes from the start. Thus the 76% of predictions have been proved to be correct, observing the arrival at destination predicted PPOI.

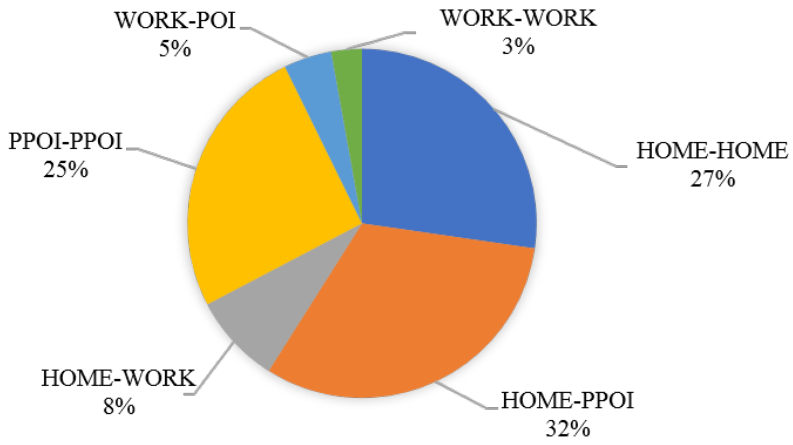


Figure 2.8: Distribution of different trajectories

As a final consideration, the UEEngine also delivered a number of simple engagements such as those to simply inform and stimulate the city users against facts. In this context, we observed that the rate of engagements viewed against the engagement sent strongly depend on the time validation for the engagement (the permanence duration of the engagement into the mobile phone of the city user). Even some the same kind of engagements, those remaining available for 30 minutes have the 1.9% of views, while those remained active for 12 hours have been viewed 30.15%. This measure has been possible instrumenting the Mobile App.

2.7 Considerations

The new challenges in the Smart City context are mainly related to the stimulation of the city users towards sustainable behaviors. For example, in mobility, exploiting energy, using city services. On this context, many ad-hoc solutions have been proposed for informing the city users and/or for engaging them with specific wired rules toward virtuous models. In this chapter we proposed a flexible languages, exploiting analytics and predictive

models, which can be used to identify users that need to be pushed towards a range of virtuous habits. On this regards, the main problems are (i) the computation of user behavior via data analytic (semantic computing, machine learning), as well as (ii) the formalization of strategies via simple and well defined language and tools for producing engagements to the city users. The language has to usable by city operators which are those that put in place the strategies on the input of political parties. In this chapter, the solution for city users engagement studied and implemented for Sii-Mobility Smart City national project in Italy has been presented. The solution has been implemented thanks to the exploitation of Km4City model and semantic computing, and a number of data analytics algorithms. The chapter also reported the first results in assessing user behavior and in using the engagement rules, putting in evidence some aspects, which can be exploited for setting up more complex rules in the future.

Chapter 3

Users' Transportation Modality Classification

This chapter presents a novel approach for user's transportation modality classification. The proposed automated system permits to understand whether an individual is stationary, on the move, walking, on a motorized private or public transport, with the aim of delivering to users personalized assistance messages to promote for example a sustainable mobility or healthy activities. The approach has been designed to provide results and collect metrics in real operating conditions (imposed on the mobile devices as: a range of different terminal kinds, operating system constraints managing applications, battery consumption manager, etc.). [1] [2]

3.1 Introduction

With the complete digitalization of the public and private transportation networks, the capability of understanding the users' behavior and the mean of transportation have become important. The presence of GPS, accelerom-

[1] Part of the work presented in this chapter has been submitted and is currently under review as "Automated Classification of Users' Transportation Modality in Real Conditions" for *Journal Mobile Networks and Applications, Springer publication* .

[2] *Acknowledgments:* this work was partially supported by the MIUR, with the Smart City national founding project Sii-Mobility SCN_00112, the University of Florence and the companies involved for co-founding Sii-Mobility

eters, sensors on mobile phones have made possible to create solutions exploiting the users' behavior and context. Information from GPS and accelerometers available on mobile phones can be combined with contextual information regarding city and mobility and transport information to assess and predict user behavior. The understanding of user behavior is the first step for providing suggestions and assistance to people on the move via mobile phones. For example, to push them in taking more virtuous behavior, consume less energy, making more sustainable their transportation, having a healthier life walking more, saving money parking closer. The research addressed in this article aims to understand the users' mean of traveling taking into account contextual data and data coming from the phones. The correct classification of transportation means can be also used for providing suggestions in the context of public or private transportation. For example, it may have sense to suggest getting down the bus at the next stop to walk a bit, or suggest parking the car in different place, respectively, using public transportation instead of the private one. Thus, the above described problem is re-conducted to the classification problem of the transportation modality/mean (car, bus, walk, bike, etc.), exploiting real time data coming from the devices and contextual information. Please note that, the contextual data are strongly different in different part of the city, and also change over time, for example busses have different timeline and paths: so that users are moving in the real space.

As described in the following Section of related works, the problem of understanding the mean of travelling of users has been many times addressed, but not working in *real operating conditions*. Most of the solutions, assume data collected from the mobile phones with high rates and high precision, identifying models only taking data in strongly controlled conditions: such as limited number of device type, limited number of users and directly engaged to keep the mobile application running in foreground, etc. This means that those solutions have note addressed real operating conditions. In the presence of real conditions, (i) data are sporadic, (ii) the rate is low and not constant, (iii) the quality of data is not uniform since sensors of different mobile phones have different precision and response, (iv) operating system may push the Mobile Application (Mobile App) in background and may apply energy saving rules to applications. These are just examples of the complexity of understanding the mean of transportation in real operating conditions. This also means that both data and methods have to be

completely re-elaborated in the latter case.

This chapter is organized as follows: in Section 3.2 the related works are presented. In Section 3.3, the general architecture for data collection, from devices to server, and related data analytics are briefly described. Section 3.4 provides a description of the classification methods adopted to identify and validate the predictive models and framework. In Section 3.5, a list of the identified metrics is reported, mainly related to: baseline, GPS, accelerometer, and historical data. Section 3.6 proposes a comparison of predictive models exploiting the data collected, to arrive at identifying the best resulting approach in terms of classification precision and recall. Considerations are presented in Section 3.7.

3.2 Related works

The problem of classifying users' mean of travelling has been addressed by a number of approaches in different research areas [168]: *Location Based Services* (LBS), *Transportation Services* (TSc) and *Human Geography* (HG). The LBS solutions aim to understand the transportation meaning in real-time to provide useful information to the user whenever he/she asks. Note that, in LBS approaches the velocity of response has been privileged with respect to the correct segmentation of a trajectory. On the contrary, in the TSc approaches, the correct segmentation of a trajectory is privileged with respect to velocity of response: TSc solutions aim at generating reliable statistics and activity-travel diaries about the transportation mode of a user when performing daily activities [63] [194]. The HG approaches focus on the segmentation of a trajectory into parts with domain-specific semantics: it is common to first split trajectories into segments where the object is stationary or moving.

On the other hand, the aim of the research presented in this article consists of developing a solution to help users during their travelling in real time, LBS solutions have been better analyzed and reported. In LBS, the transportation means' classification is regarded as an online process: an algorithm that provides the current transportation mode of the user in real-time or quasi real-time. Moreover, those algorithms should generate information that can help to understand better how the user is travelling. To this end, different types of data/sensors have been exploited: GPS [193], accelerometer [122] [213] [208] and the combination of GPS and accelerome-

ter [175] [143] [190]. On this topic, [193] have compared five different models using data collected from GPS classifying the users' travelling means in six categories (walk, train, driving, stationary, bus, bike). Their work exploits a transportation network dataset with FGIS information together with the real-time positions of buses and trains, and they have identified a set of seven features (average accuracy of GPS, average speed, average heading change, average acceleration, average bus closeness combined with candidate bus closeness, rail line trajectories closeness). [193] have used the GPS position sampled every 15s and a window frame of 30s, presenting 92.8% of accuracy by using the Random Forest algorithm. Please note that, today, with the high attention to power safe of the present mobile devices and operating systems, 15s of stable sampling rate is somehow realistic only if the application providing data is in foreground as a navigator, while on all the other cases, is not, so that it is not realistic today, since if the user is walking, moving on in bus or train do not use the navigator, obviously.

[122] have proposed a study that involves only accelerometer data. They have obtained an 80.1% accuracy and an 82.1% recall for seven transportation modes, by using both AdaBoost and Decision Tree (two-stages classification). The authors have collected 150 hours from 16 users considering 3 different devices with accelerometer sampler at 0.01s (100Hz) of frequency. On the same line, [213] have extracted 22 features in the time-domain and 8 in frequency-domain for the evaluation of five classes of transportation modes as motorcycle, car, bus, tram, train and high-speed rail. [213] have compared three different classifiers (Decision Tree obtaining an 84.81% average accuracy, AdaBoost with a 87.16% average accuracy, and SVMs with a 90.66% average accuracy). About 8311 hours of data (100GB) have been used for the learning process, with a sampling rate of sensors of 0.03 sec (30Hz), collected using a single device (HTC One mobile with Android system). [208] have considered a small data-set of 12 hours (5544 samples of six transportation modes) from 7 different users, obtaining a 70% accuracy with a Decision Tree algorithm. [175] have demonstrated that, taking into account of both GPS and accelerometer the accuracy can be improved. The authors have considered five different transportation modes (still, walk, run, bike and motor), gathering 15 minutes of data for each transportation modes (six terminals per person, 1 type of device), and the total amount of data collected across all 16 individuals was 120 hours (a small data set, produced in controlled conditions). They have achieved a 93.6% precision using a combination

of Decision Tree and Hidden Markov Model (two-stages classification), with both accelerometer and GPS features involved, using a sampling rate of [190] made a distinction among different type of non-motorized motion (walking, running, biking), vehicular and random movements, using the accelerometer sensors of mobile. They have used a Decision Tree classifier and have applied a Markov model smoother on top of the output of the Decision Tree. Thus, obtaining a 91.53% precision, based on a small data set, produced in controlled conditions: 50 hours of collected data from 15 distinct individuals (with android devices), an accelerometer sampling rate of 0.01 sec (100Hz) and a gps sampling rate of 5s. On the same idea, [143] have trained a Decision Tree classification model obtaining an 82.14% accuracy (with a gps and accelerometer sampling frequency rate of 1s and 0.04s respectively), while [169] have obtained a 90.8% accuracy using a Random Forest algorithm on seven different modes of transportation (walk, bike, car, bus, subway, train, ferry), both using a single device. [121] combined GIS features and speed with socio-demographic characteristics and travellers personal preferences to facilitate better transportation modalities detection (stationary, walk, bus/car, rail). The authors focused their research on mobility-affecting disabilities users, achieving an overall a 78% accuracy with the Random Forest classifier.

Recently, [44] have proposed a two-layer hierarchical classifier to predict five classes of transportation mode (car, bus, walk, run, bike), achieving a 97% accuracy. They state that a hierarchical approach can increase the accuracy with respect to the traditional classification algorithms. As first step, they applied a multi-class classifier to select the two transportation modes with higher probability, given the test data. As second step, they used a binary classifier to discriminate between the chosen pair: the binary classifier is specialized in the pair of modes identified in the first step and uses a specific feature subset. Ashqar et al., have used the GPS, accelerometer, gyroscope and rotation vector sensors, sampling the data with the highest possible frequency, during the workdays and working hours, only using two different devices. In particular, an accelerometer frequency rate of 0.01s (100Hz) and a gps frequency rate of 0.04 sec (25Hz) were applied. [186] evaluated the transportation mode detection through a three-steps algorithm: a segmentation, a fuzzy rule transport mode detection and a consistency correction. The authors have achieved a 75% accuracy considering five transportation modes, walk, bike, bus, car and train, tracking users position at least every 4s without information about public transport opportunities. Thus, also in

this case, the rate for data acquisition is incredibly high and unrealistic for actual applications (with a regular gps sampling rate of 1s). [211] have presented a Convolutional Neural Networks (CNN) based method to automatically extracting features for the identification of transportation means, thus achieving a 98% accuracy to distinguish between train, bus, car, metro. The authors have conducted the experiment using a single device (Android smartphone) and collecting 200 hours of transportation data and a accelerometer sampling frequency of 0.01s (100Hz). In Table 3.1, a summary of the state of the art solutions for understanding the travel means is reported. The comparison is putting the attention to a number of aspects that may enable those solution to work on real operating condition or not. And in particular on: (i) the data exploited (both sensors and contextual data if any), (ii) the sampling rates in seconds or sampling per second (regular and/or unregular), (iii) the number of users involved in the training, (iv) the number of features used, some features have to be computed on client and/or server side since the mobile device has not all the contextual information accessible in real time (GIS information), (iv) the number of device types the variability of sensors quality of devices connected with different operating systems and versions, and (v) the precision accuracy obtained. As it can be noted from Table 3.1, almost all the state of the art solutions adopted very high rates for GPS data acquisition, with limited number of devices. Thus, the energy consumption and the corresponding network bandwidth for data transmission are issues that are limited by the present mobile operating systems, that apply energy safe rules on all the applications. So that, those solution are almost unfeasible in real operating conditions. Mobile operating systems allow to keep the high rates (in the order of seconds) only when applications are running in foreground. On the other hand, some of the operating systems, to save energy force the services into the Mobile App to sleep and wake up only after few minutes.

3.2.1 Research aim

The aim of our research has been to realize a solution overcoming the previous solutions at the state of the art to classify the transportation modes to deliver at the users, personalized assistance messages for:

- sustainable mobility, to incentivize ecological transportation choices, suggesting alternative public mean of transport (bus/tram) instead of

Table 3.1: Related Work implementation overview

Authors	Classes	Sensors	Sampling	#users	#feature	Precision accuracy
Wang et al	Stationary, Walk, Bike, Bus, Car, Metro	Accel	35Hz	7	23	70
Manzoni et al	Stationary, Walk, Bike, Motorcycle, Car, Bus, Metro, Train	Gps Accel	1Hz 25Hz	4	1	82.1
Hemminki et al	Stationary, Walk, Car, Bus, Train, Metro, Tram	Accel	100Hz	16	27 +5	84.9
Prelipcean et all	Walk, Bike, Car, Bus, Metro, Train, Ferry	Gps Accel	50m 5Hz	9	11	90.8
Shah et al	Stationary, Non-Vehicle (Walk, Bike, Run), Vehicle	Gps Accel	0.2Hz 100Hz	15	6	91.5
Yu et al	Stationary, Walk, Run, Bike, Vehicle (Motorcycle, Car, Bus, Metro, Rail, Train)	Accel	30Hz	224	22 +8	91.5
Stenneth et al	Stationary, Walk, Bike, Car, Bus, Train	Gps Gis	0.07Hz	6	7	92.8
Reddy et al	Stationary, Walk, Run, Bike, Vehicle	Gps Accel	1Hz 32Hz	16	4	93.7

the private car/motorbike. This is feasible only having a tools for identifying who is taking the private car with a relevant frequency;

- healthy suggestions, better and enjoyable life, to stimulate users in dedicating a part of their time and moving needs to exercising their body. For example, suggesting getting out of the bus in advance, park in other locations, etc.;
- implementing city strategies to change city user attitudes. For ex-

ample, to: reduce the number vehicles in certain areas, to increase the usage of public transportation in specific time slots, to stimulate tourists in taking diverse paths in the city visits, and to select less busy parking areas, etc. [50], [53].

With this purpose, the real-time identification of a private transportation mode (car or motorbike) has a central role in assistance messages delivery. Therefore, according to the above described real operating conditions, the techniques have to produce high classifications accuracy to identify transportation modality of a user, in the presence of (i) large discontinuities samples of data (from sensors and sporadic communications to the central computation modules), (ii) relevant differences which may be due to the different kind of mobile phone features in terms of sensors and precision. The capability of working on real operative conditions, it also mandatory to avoid the App to be identified by the power safe procedures that are nowadays installed into the mobile phones to reduce battery consumption. Therefore, the proposed solution overcome the above mentioned solutions at the state of the art, for the aspects focused on sensor energy consumption factors and real conditions. The solution has been tested on a real application (delivered to the users via official App stores such as Google Play Store, Apple App Store, and accepted by common users, see "*Tuscany where what...*" on the stores). As described in the following, it is capable to cope with the constrains introduced by terminal manufactures on battery usage for background and foreground services. Moreover, no restrictions on the modality of mobile device usage have been imposed, differently to what has been imposed in the state of the art experiments in which the devices have been asked to keep: (i) the application running in foreground to get more precise GPS data, (ii) the device in a proper position/orientation during the usage; and/or to (iii) use specific devices.

3.3 Architecture and Data Collection

The proposed solution relies on a client-server architecture, where the mobile application can be installed on different operating systems (Android, iOS and Windows devices), with different versions [49]. The sensors' values collected on the mobile device (client-side) are sent to the server that enriches them with additional context information derived data (GIS, geographical information system and knowledge), etc., as described in the sequel. At the

same time, the server executes the real time classification algorithm to compute the transportation mean classification for each user. The information is stored on server as a report on the preferred user's travel mean. On this basis, it is possible to set up complex strategies that may be triggered when the user behavior reaches certain specific conditions for example to assist and/or engage the users in their daily activities (even rewarding them, in the cases of virtuous behavior; for example, when a suggestion has been followed) as explained before in the Chapter 2. For example, a strategy for stimulating the city users may be based on a firing condition which sends a suggestion to all city users that take their private car to perform the same trip path at least 3 times per week, and at the same time the trip could be easily performed by using public transportation. Thus, the system may inform those city users of the possible alternative, and some of them may follow the suggestion. As a result, by exploiting the user behavior analysis, the solution may detect the acceptance of the suggestion by detecting of change of behavior and may automatically reward the user with a bonus or discount, and deliver congratulations. See [50], on rules and strategies. Figure 3.1 provides a high-level overview of the software architecture and its main components.

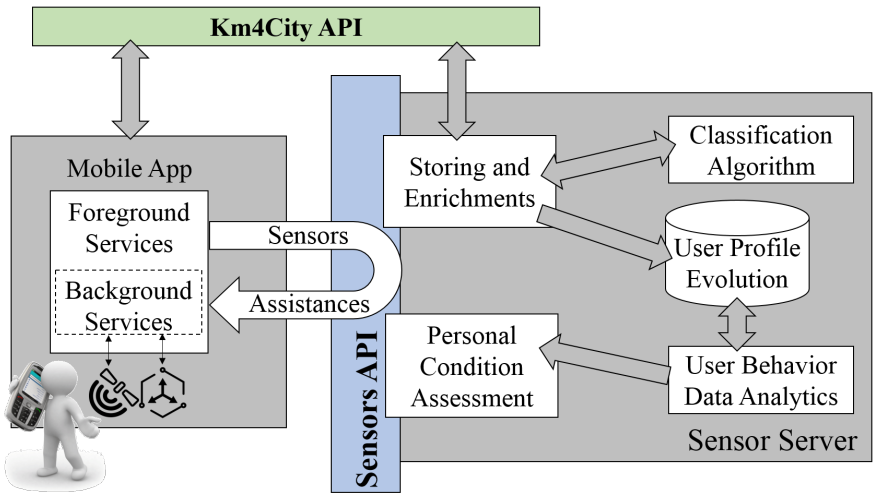


Figure 3.1: System architecture

The **Mobile App** may execute services in background or just in foreground depending on the operating systems. Therefore, different data collection strategies are used on different devices and operating systems. Android terminals accept the background processes; thus, the data collection service may work in background remaining always active. In other operating systems, running background services are allowed, therefore, data are collected when the App is in foreground. Both services (foreground and background) follow the same workflow of operations independently on the implementation languages: Java for the background service on Android and JavaScript for other operating systems, since the Mobile App has been developed in Apache Cordova.

The data collected on the mobile applications are called by us as "***Sensor Data Package***" (considering the mobile device as a sensor), and are: (1) positions and movements of the user's device through sensors' and derived information such as GPS latitude and longitude, speed and acceleration; (2) device characteristics as the type of operating system, application version, device model, to provide also statistics on the mobile application usage and to highlight the most widespread configuration; and (3) the user characteristics (language, user profile) to personalize the mobile user experience. The collection of Sensor Data Package aims at gathering a correct GPS position at a given minimum refresh time of 30s, while it is managed by the operating system power safe strategy. As location/position data for the mobile, the GPS position is preferred. On the other hand, some devices and operating systems derive the position from the network connection (cellular connection and/or Wi-Fi hotspots), or by using some fused/mixed strategies, which actually combine all of them. In some cases, the GPS is not available, and this may mean that the user has disabled the location detection, the GPS. Thus, the **Location Measure kind** can be GPS, Network (cellular position or Wi-Fi) or mixt. If, at least one location mode is active, two different cases may occur: (1) it is possible to detect the location of the device at that time (for example, the user is inside a building where the GPS signal fails to retrieve information from satellites) or (2) the detected location is not up to date (for example, a position is obtained by the device, then the user entered inside a building: the position may be the last obtained and sadly it is too old). In both these cases, a position update is required: if the updated value is not accessible, a simply ALIVE message is sent to the server, communicating the other values of the Sensor Data Package. In the absence of a local

measure, the accelerometers information could be collected anyway from the device and thus sent to the server.

The Sensor Data Package is collected in a buffer and sent by the mobile application to the server periodically by the active background or foreground services. When the sending of the last packages is successful, the data sent are deleted from mobile device.

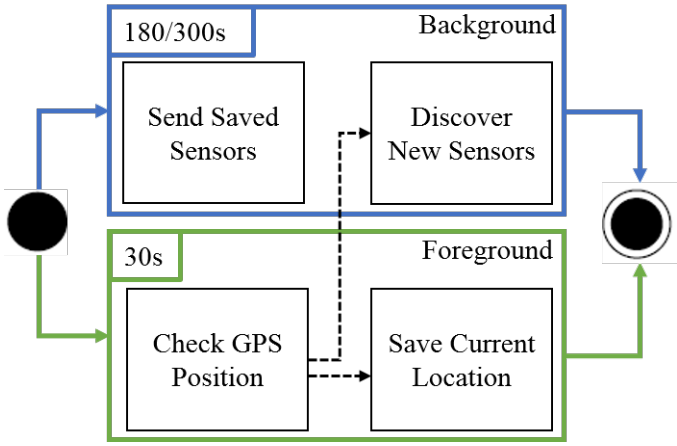


Figure 3.2: Data Recovery Service Timing

In the Figure 3.2, the protocol's operations performed by the background service are shown. The block called "Save Current Location" stores (every time the GPS position is available) a list of data used to calculate further features of the terminal movements at a given time, whenever these data are sent to the server-side. Both background and foreground services perform the same operations in their corresponding operative conditions of the App.

The **server-side process** (see Figure 3.1) collects all the data sent by the mobile applications and computes several features for enriching each the single record of data (package) associated with a given timestamp of measure. Among the computed features, the distance/proximity of the GPS coordinated of the mobile with respect to the railway, or bus lines, or highway, cycling path, and/or parking zones. This kind of geographical information is retrieved from the Km4City knowledge base in Big Data technology by using the Smart City API [152], interface to semantically integrated information for applications and services [60]. Other computed features are the average

velocity of the last period, etc., as described in the sequel.

Finally, the process on the server puts in execution the classification algorithm to retrieve the user's transportation mean. This information is also stored point by point for further processing, and to eventually produce contextual assistant messages to be delivered to the user's device by some rule editor according to the strategies identified by the municipality or by some city operator [50]. On the other hand, the classification approach of transportation mean has to predict the travelling means for the next time slot, in order to provide suggestions in time and not only with the delay due to the data collection - sending - and computing process.

3.4 Classification techniques compared

This Section presents an overview of techniques considered to create a solution for classifying the transportation mean of the users in the move. During the experiments, several unsatisfactory techniques have been tested, thus, among the possible techniques, we have chosen to present here only the comparison of the most promising approaches, namely: **Random Forests, Extremely Randomized Trees, Extreme Gradient Boosting, Super Learner and Hierarchical approach**. Please note that Classification Trees are machine-learning methods which are used for constructing exploration, description, and prediction models. The usage of Classification Trees approaches, i.e., *Extremely Gradient Boosting* and *Random Forests* methods, have potential advantages in the predictive model's construction. *Classification Trees* are free of distributional assumptions and they can handle different types of responses, such as categorical, numeric, multivariate, censored and dissimilarity matrices; they are invariant to predictors monotonic transformations; the presence of missing values in the predictors are handled with a minimal loss of information. On the basis of the above described properties, the adoption of classification and regression trees - i.e., Extremely Randomized Trees or Random Forest methods which are free of distributional assumptions potentially provide an advantage for the construction of predictive models. As further step, the multi-class problem has been divided in a collection of binary classification problems: they were analyzed using the **Super Learner** algorithm, combining the different learning techniques above. Moreover, a **Hierarchical** approach has been proposed and compared with the above approaches based on a single multi-class classi-

fier. Therefore, for completeness, a short overview of the above-mentioned approaches is reported in the next subsections. In Figure 3.3, a schema of the processes adopted is reported for both classic classifiers (a) and Super Learner (b).

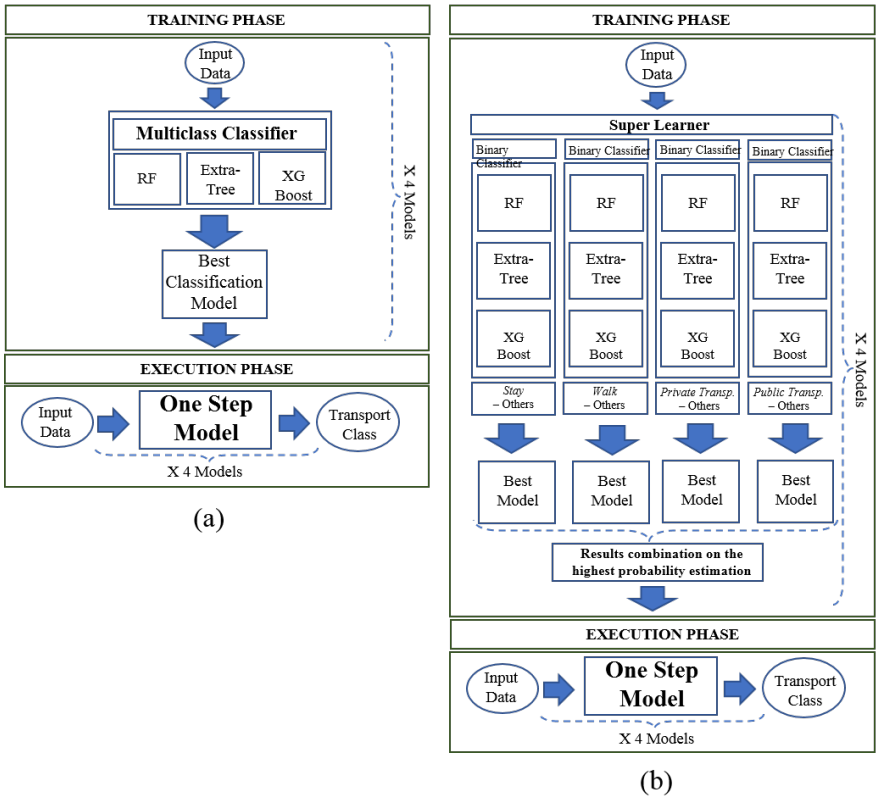


Figure 3.3: Comparison of One Step models: (a) classic and (b) Super Learner

3.4.1 Random Forest

Random forests algorithm has been proposed by Breiman in [67] as an improvement of the Tree Bagging approach. In Random Forests, a different bootstrap sample from the original data was used to construct each tree. For each tree of the collection, each split is determined by a randomly chosen subset of predictors; this procedure reduces the correlation between predictors of the individual trees, and each tree has the same expectation.

3.4.2 Extreme Gradient Boosting

Gradient Boosting [107] is a way to reduce the variance, respect to other decision tree methods. The Boosting Trees algorithm is an evolution of the boosting methods application. Boosting methods [106] performs classifications by weighted majority vote, and they have the advantage to fit many trees of different dimensions to reweighed versions of the training data. In Gradient Boosting, small regression/classification trees are built sequentially from the gradient of the previous tree loss function (pseudo-residuals), and in order to produce an incremental improvement in the model, at each iteration, trees are built from random sub-samples of the dataset. Basically, given a modality i with a vector of covariates X_i , a K additive functions is used to predict the output of the tree ensemble model.

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F$$

where F is the set of all possible trees. The f_k function maps the value in x_i to a certain output, at each step k. Extreme Gradient Boosting tries to minimize the regularized object as follow:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y_i}, y_i) + \sum_k \Omega(f_k)$$

where $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda||\omega||^2$.

In the above equation, l is a differentiable complex loss function (Mean Square Error). while the second term penalizes the complexity of the model in terms of number of leaves in tree T and vector of scores on leaves ω to avoid over-fitting. In general, boosting procedure outperforms the random forests. Extreme Gradient Boosting in [77] is an efficient and scalable implementation of the proposed Gradient Boosting framework by Friedman [107].

3.4.3 Extremely Randomized Trees

The Extremely Randomized Trees is a tree-based ensemble method for supervised classification and regression problems. It involves on the randomization of both attributes and cut-points choices during the splitting of a tree node. In the extreme case, trees are totally randomized and the structures of them are independent of the learning sample output values. The strength of the randomization can be tuned to problem specifics by the appropriate choice of a parameter [110]. With respect to random forests, the Extremely Randomized Trees idea is to drop the use of learning sample bootstrap copies, and instead of trying to find an optimal cut-point for each one of the K randomly chosen features at each node, it selects a cut-point at random [110]. From a statistical point of view, the idea to drop the bootstrap copies leads to an advantage in terms of bias because the cut-point randomization has an excellent variance reduction effect.

3.4.4 Super Learner

In this case, the multi-class problem can be divided into a binary classification problems collection: considering 4 classes (C_1, C_2, C_3 and C_4), it is possible to adapt binary classifiers for C_1 vs C_2 or C_3 or C_4, C_2 vs C_1 or C_3 or C_4, C_3 vs C_1 or C_2 or C_4 and C_4 vs C_1 or C_2 or C_3, and then combine the results to make a past classification on the highest probability estimate. Note that, the approach presents a certain flexibility for the classifier for the different categories. In general, it is not possible to know *a priori* which learner will produce the best performance for a given prediction problem [201] [166] and the relative performance of various learners depends on the true-data generating distribution. The idea is to apply a set of candidate learners to the observed data and choose the optimal learner for a given prediction problem by using a cross-validation risk approach. Thus, the learner algorithm with the minimal cross-validation risk is selected. Super Learner performs asymptotically to produce the best possible weighted combinations among the set of candidate learners considered [199], [200]. Therefore, considering $L(n)$ a collection of learners $\hat{\Psi}_l$, $l = 1...L(n)$, in parameter space Ψ. Super Learner is defined as

$$\hat{\Psi}(P_n) \equiv \hat{\Psi}_{\hat{L}(P_n)}(P_n)$$

where $\hat{K(P_n)}$ indicate the cross-validation selector.

$\hat{K}(P_n)$ selects the best performing learner in term of cross-validated risk:

$$\hat{L}(P_n) \equiv \arg\min_{l} E_{V_n} \sum_{i,V_n(i)=1} (y_i - \hat{\Psi}_l(P^0_{n,V_n})(X_i))^2$$

In particular, $V_n \in \{0,1\}^n$ indicates a random binary to split the learning data set into a training set $\{i : V_n(i) = 0\}$ and validation set $\{i : V_n(i) = 1\}$. The empirical probability distributions of the validation sample and training sample are denoted by P^1_{n,V_n} and P^0_{n,V_n} respectively [201]. In the performed experiments, the Super Learner has been applied with 10-fold cross-validation to select the optimal learner given the following set of candidates: Random Forest, Gradient Boosting, Extremely Randomized Trees.

3.5 Data and Feature Definition

The features taken into account by the classification algorithm have been selected from a larger set considered during the preliminary analysis and experiments. The process of features reduction has been performed by assessing their relevance in the contexts of the algorithms tested as partially mentioned in Section 3.3. The aim was to identify the smallest subset of features without reducing significantly the precision of the travel mean's classifications. As a result, Table 3.2 includes the selected metrics and the features, classified in 4 categories, collected from the mobile as **Sensor Data Package**, with those computed from the server-side to be used by the classification algorithm. Some of these features can be used for both users' travelling mean classification, and for creating firing conditions for implementing strategies. In Table 3.2 "Where" can be: "D" when the measure is produced on the Device, and "S" when is computed on server-side. Each measure is collected/referred at a given **Day and Time**, and from this value can be easily derived from the device or from server if the day is a working day or not (**Non-Working Day**). The same approach can be followed to detecting the **Time Slot** in which the measure has been collected. The Time Slot strongly influences the attitude of the city users to move by using different means.

Table 3.2: Overview of Sensor Data Package feature measured at a given time from the mobile or computed on server-side ("Cat" means Category)

Cat	Metrics	Description	Where
Day/Time Baseline and GPS	Day and Time	Day and Time of the sample package	D
	Non-Working day	1 if weekend or vacation, 0 if it is a working day	D/S
	Time Slot	Slot of the day (morning, afternoon, evening, night)	S
	GPS latitude and longitude	Position of the device in GPS coordinates	D
	Accuracy	GPS Sensor's Accuracy of Device	D
	Location Measure kind	Types of Location measure: GPS, Network, Mixed/Fused	D
	Speed	Speed as provided by the GPS driver of the mobile (as m/s)	D/S
	Average Speed	Average speed of the measures collected in the last two minutes	D/S
	Phone Year	Year/age of the terminal	D
	BDS	Availability of a BDS compliant GPS Sensor	D
	User Type	User Type: commuter, citizen, students...	D/S
Accelerometer	Average linear magnitude of acceleration	Average of the acceleration magnitude calculate on five measurements	D
	Linear acceleration of X-axis	Acceleration of the device along the X-axis	D
	Linear acceleration of Y-axis	Acceleration of the terminal along the Y-axis	D
	Linear acceleration of Z-axis	Acceleration of the terminal along the Z-axis	D
Proximity	Rail Line	Device in proximity of a rail line	S
	Sport Facilities	Device is in proximity of a sport facilities (Bool)	S
	Tourist Trail	Device is in proximity of a tourist trail (Bool)	S
	Green Areas	Device is in proximity of a green areas (Bool)	S
	Bus/Light-rail Line	Device is in proximity of a bus line or a light-trail line (Bool)	S
	Cycle Paths	Device is in proximity of a cycle path (Bool)	S
Temporal window	Previous speed	Speed of the device of the previous 12 minutes slot	S
	Previous average speed	Average speed on the measures collected in a 12 minutes time slot	S
	Previous median speed	Median speed on the measures collected in a 12 minutes time slot	S
	Speed distance	Speed (m/s) calculated on the distance between two consecutive coordinates and the time passed between the observations	S

As described in Section 3.3, the information about the user's movements is collected from the device sensors. If the user has the Mobile App in foreground, the data are sent to the server every 1 minute and 30 seconds (sending interval). This interval can be reduced by the user (via the setting of the App) to an update up to 30 seconds, to have a more accurate assistance. If the App is not used, the data collection is performed in background modality, thus the measures and sending rates may become up to 3/5 minutes, forced by the operating system/device, which in some cases can hibernate the App. Therefore, in order to make the solution viable in real conditions (differently from the state of the art solutions), a set of strategies and robust classification algorithms have been put in place. Among them, technics for filtering noise and GPS errors, and for smoothing the sequence of the user locations (user trajectory) have been used.

A Sensor Data Package l_i represents the user context at a specific time t_i and is composed by the **GPS latitude and longitude** (according to a Location Measure kind), speed, and accuracy of the measure plus a list of N additional features *(feat-1...feat-n)*:

$$l_i = \{latitude_i, longitude_i, speed_i, accuracy_i, feat-1_i, ..., feat-n_i\}$$

user trajectory t_{ir} is a sequence of l_i that describes the movements of a user to move from l_i to l_r:

$$t_{ir} = \{l_i, ..., l_r\}$$

A segment s_{uv} is a trajectory t_{uv} in t_{ir} where a user keeps the same mobility mean:

$$l_i \rightarrow mobility - A \rightarrow l_u \rightarrow mobility - B \rightarrow l_v \rightarrow mobility - C \rightarrow l_r$$

where $l_u \rightarrow mobility - B \rightarrow l_v = t_{uv}$

The distance between l_u and l_v can be approximated by using flat-surface formulae between the two coordinates $< latitude_u, longitude_u >$ and $< latitude_v, longitude_v >$.

A measure of the terminal **Speed** can be directly retrieved from the GPS sensor (for example, every 30 seconds or at the rate imposed by the device). On the other hand, the above mentioned **Average Speed** of Table 3.2 is

calculated over the sequence of l_i in the same sending slot from the mobile device, to cut out eventual errors coming from GPS sensor. If the Mobile App is in foreground the Average Speed is computed every 2 minutes (4 measures of 30s, if any). If the Mobile App service for collecting data is in background, and 2 minutes passed before a measure is available (probably the operating system put the application in hibernate mode). The service tries to wake up whenever it is possible (if the operating system on the device allows us to wake the service up), to retrieve a bounce of new l_i to calculate a more precise Average Speed. If at a given time, the measure of Speed is not available, a *valid* value may be obtained on the server-side by using the distance between the two last GPS coordinates and the current refresh time. Figure 3.4 overviews the scenario.

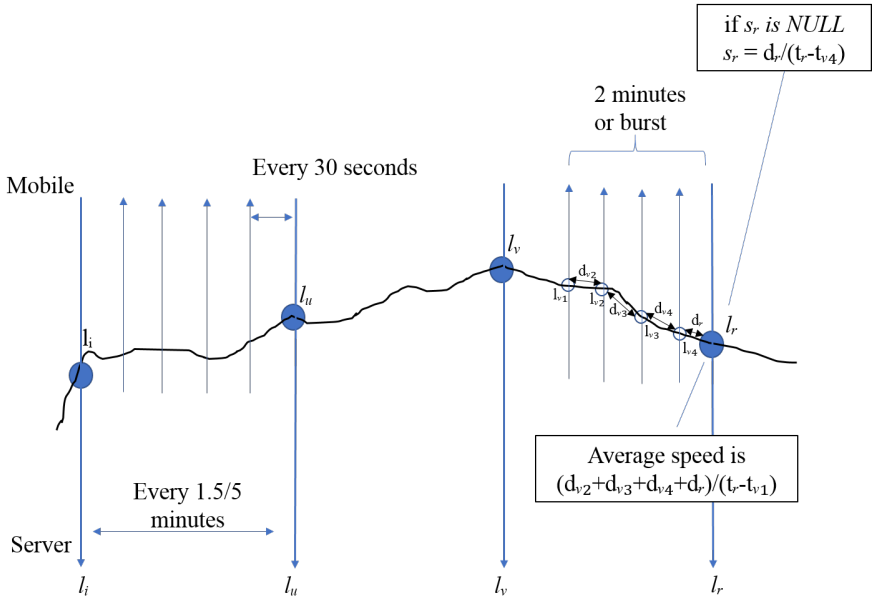


Figure 3.4: Speed and average speed

The **Location Measure kind** is an important feature to understand the location measures reliability. Usually, the measures obtained and marked as "GPS" by the mobile device are quite accurate, even if they suffer time by time of well-known problem of shading (e.g., urban canyoning) or blocked

(under the bridge) [148]. The location measures, labelled as "Network", resume the position from the location of the available Wi-Fi hot spots or GSM/4G/5G in the mobile connection; while those marked as "Mixed" modality is obtained by the operating system by merging the previous strategies according to different algorithms that may depend on the operating system kind, sensor kind, etc. The Location Measure kind strongly depends on the factory settings of the device, that make very difficult to force a pre-determinate modality from the App. On the other hand, the approach permits the users to optimize the battery use and to have more precise positioning in critical cases, thus the switch among modalities is not under control of the App. The **Accuracy** of the GPS measure is reported in meters from the device and can be used from the classificatory algorithm to eventually discharge entries.

Terminal model and its characteristics are also tracked and passed to the classification algorithm. Thus, the **Phone Year** of production of the device and the characteristics of the GPS sensors strongly influence the reliability of measure and thus have and have been considered as variable, differently from the state of the art solutions. Old terminals usually support just A-GPS modality, meanwhile new ones' support also GLONASS and BDS standards. We have been also capable to experiments on new GALILEO compatible sensors available on new Samsung models. As a result, there is evidence that the type of the sensor influences the accuracy of location retrieval [139].

3.5.1 Accelerometer Features

Values from the **Accelerometers** of the terminal/device are always available and are sampled. Using the linear acceleration of the device avoids taking measures influenced by device orientation (horizontal or vertical, in the hand or in the pocket). Not all the mobile devices provide this information (some of them just return the non-linear values, influenced by the gravitational acceleration, and orientation, thus needing a de-rotation). On the other hand, almost all the relatively new devices already have this aggregated measurement available (for Android 8 [2]]). Phone Year variable allows us to take this into account. Thus, the three measures of linear acceleration on three axes have been considered aggregating five consecutive acceleration measures for computing an average magnitude as:

$$\textit{Average Linear Magnitude of Acc} = \sum_{k=1}^{5} \frac{\sqrt{acc_{x_k}^2 + acc_{y_k}^2 + acc_{z_k}^2}}{5}$$

3.5.2 Distance Feature

On the server-side, the Sensor Data Package collected from the devices via the App are enriched by computing and, in most cases, exploiting the Km4City knowledge base of the City via Smart City API. This allows to retrieve contextual information about the closeness of the device/user with respect to: Railway Line, Sport Facilities, Tourist Trail, Green Areas, Bus/Light-rail Line, and Cycle Paths. The closeness features are binary values that specify if the location is closer to those structures, in the range of 30mt. This derived information is very valuable for understanding some transportation means. For example, to be close to a Rail and/or Bus/Light-rail line for a number of points of a trip permits to infer bus/train modality (train, bus and light-rail run just in their closeness) with a very high probability. On the other hand, the closeness to a cycle path cannot directly infer that a user is using a bike because the user can be in its proximity by a car and with similar speed or a bike can run also away for the cycling path (see Figure 3.5).

3.5.3 Temporal Window Feature

Besides having instantaneous measurements about device/user's mobility, the speed values in the last 12 minutes time-frame is also computer on server-side, as well as the average and mean value between these measurements. This allows to reduce the noise overcoming disruptive mobility conditions mainly related to traffic congestion or temporary signal absence. This is also due to the fact that the service for collecting data on the mobile device runs on a real application (foreground/background) conforming to the policy of "energy saving" of the user to have shortage of data for up to 3/5 minutes. So that, in real conditions, it is very important to avoid battery drainage warning, that may stimulate the user to un-install the App from the device. In order to perform an addition refinement on speed measures, mean and median speed and distance between GPS coordinates are also computed. The User Type specified by the user in the App during installation or setup

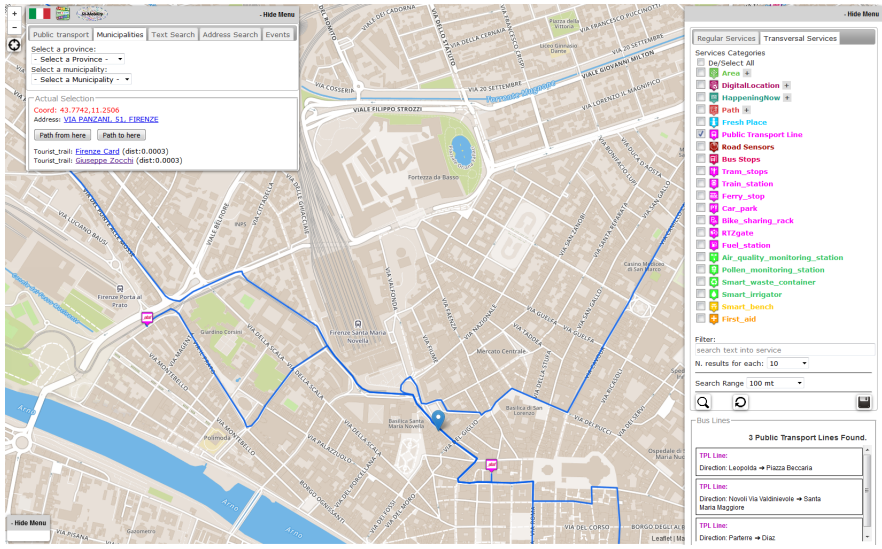


Figure 3.5: Bus-line in proximity computation

permits contribute to the classifications and to the strategies. The **User Types** are: citizen, commuter, student, tourist, etc. We noticed that different profiles present a different approach in everyday mobility and, so on the transportation mode they normally use.

3.6 Results From Classification/Prediction Models

According to the above described data, the challenge was to predict the transportation mode, whether an individual is stationary, or is walking, or moving on a motorized private transport (car or motorbike) or using a public transport (tram, bus or train). The experiment has been conducted on about 30K observations, collected from April to August 2017 on **38 different users and 30 different kinds of devices**. Note that, each user can use the mean of transport they want. When the mode of transport is changed, the user was asked to notify the change to the App for creating the learning set and for validation. As mentioned above, no restriction was imposed on how the

phone should be held during movement (foreground/background, on hand or bag, etc.). Unlike the experiments reported in the literature, most of the data was collected in the background because the phones were kept in pocket or bag, in fact there is a non-conformity in the frequency distribution of the collected data. In details, the frequency average is equal to 180 seconds and the variance is equal to 13240 seconds. The frequency distribution of the sampling period is reported in Figure 3.6.

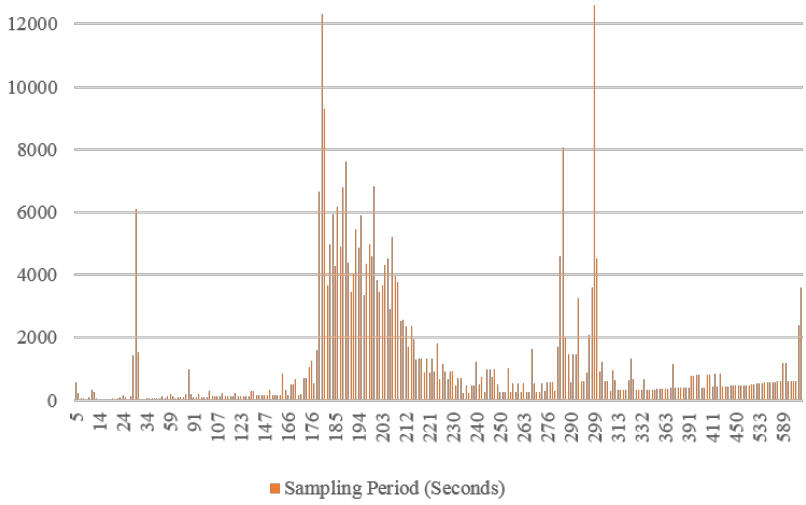


Figure 3.6: Frequency Distribution of Sampling Period

The training set has been created by randomly selecting the 80% of the collected data, while the test set was the remaining 20%. In the general framework, three different approaches were more successfully considered - i.e., Random Forest (RF), Extremely Randomized Trees (Extra-Trees), and the Extreme Gradient Boosting procedure (XGBoost). Those approaches have been tested by using the above presented features/metrics (see Table 3.2), classified by categories as: baseline and GPS features, accelerometer features, distance features and temporal window features. The comparison among those models has been reported in Table 3.3, in terms of resulting data. From the comparison, it is evident that all the approaches are capa-

ble to produce satisfactory predictions (the accuracy for each model exceeds 90%) for the identification of the transportation means. According to the results reported in Table 3.3, the differences among the different approaches are not very relevant, while the results suggest that the **Extra-Trees** resulted to be the better-ranked approach in terms of accuracy and $F_1 score$. In Table 3.3, the$F_1 score$ is reported: $F_1 score$ has been used to measure the models' performances. This is a measure to evaluate the robustness of a model for making predictions, as a compromise between precision and recall:

$$F_1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$Precision = \frac{\#correctlyclassifiedistancesintoclassi}{\#istancesclassifiedasclassi}$$

$$Precision = \frac{\#correctlyclassifiedistancesintoclassi}{\#istancesbelongingtotheclassi}$$

Table 3.3: Classification Models Comparison on four classes of transport mode: stationary, non-motorized, private transport, public transport

Classifier Models	**Accuracy**	**Precision**	*Recall*	F_1**score**
Gradient Boosting	0.947	0.773	0.828	0.800
Random Forest	0.942	0.774	0.869	0.819
Extra-Trees	**0.953**	**0.827**	**0.869**	**0.847**

According to this our first result, the Extra-Trees algorithm achieves an accuracy of 0.953, and a precision of 0.827. It should be remarked that, these results have been obtained and can be produced by observing data coming from a large range of devices and a variable sampling rate (up to 5 minutes). The model produce allows to understand if a user is moving with a public or private transport. On the contrary, in [175], a precision of 0.937 has been obtained by using a single device, Nokia n95, and a constant sampling rate of 60s, which is not realistic with present mobile operating systems. With Reddy's classification was only possible to know if a user is moving with a motorized vehicle. The same considerations apply to: [193] where data come from three different devices and they are taken with a constant rate of 15s achieving a precision of 93.7%; and to [213] achieving a

precision of 91% with accelerometer sensor data only, without distinguishing the type of motorized transport. Moreover, Table 3.4 reports the assessment of the results performed for each travelling mean classification for the Extra Tree procedure according to our first result. The travelling mean class with lower accuracy is Walk. This is probably due to the fact that, it is not easily to understand if a user is walking or not, since the GPS sensors accuracy is very noisy in indoor scenarios, with frequent jumps passing from the different modalities: wifi-mixed, etc.

Table 3.4: Extra-Trees Prediction Model: Statistic by class

Extra Trees Model	**Stay**	**Walk**	**Private Transport**	**Public Transport**
Sensitivity	0.978	0.731	0.869	0.917
Specificity	0.901	0.988	0.987	0.996
Pos Pred Value	0.977	0.770	0.827	0.936
Neg Pred Value	0.904	0.985	0.990	0.994
Balanced Accuracy	0.940	0.859	0.928	0.956

3.6.1 Combining with Super Learner

Subsequently, with the intention of improving the precision, the Super Learner algorithm has been applied by dividing the multi-class problem into four binary classification problems, with 10-fold cross-validation, to estimate the risk on future data and select the optimal learner given the set of candidates above: Extra-Trees, RF, XGBoost. Then the results have been combined to make a classification on the highest probability estimate. Taking into account the classes of transport modality above (i.e., stationary, walking, private transport, public transport), **four different binary classification models** have been constructed:

- stationary vs walking, private transport, public transport;
- walking vs stationary, private transport, public transport;

- private transport vs stationary, walking, public transport;
- public transport vs stationary, walking, private transport.

The results for each binary classification model are reported in Tables 3.5. Where: RCV risk is a measure of model accuracy or performance (at lower value corresponds a lower risk and higher accuracy).

Table 3.5: Super Learner results on each binary Classification Model: Couples A to D

Method	**RCV risk**	**Coef**
Extra-Trees	0.0282	0.5391
RF	0.0287	0.0562
XGBoost	0.0300	0.4047

(A) Stationary vs Walking, Private Transport, Public Transport.

Method	**RCV risk**	**Coef**
Extra-Trees	0.0234	0.6277
RF	0.0259	0.0091
XGBoost	0.0252	0.3632

(B) Walking vs Stationary, Private Transport, Public Transport.

Method	**RCV risk**	**Coef**
Extra-Trees	0.0213	0.6857
RF	0.0235	0.0000
XGBoost	0.0239	0.3143

(C) Public Transport vs Stationary, Walking, Private Transport.

Method	**RCV risk**	**Coef**
Extra-Trees	0.0087	0.6296
RF	0.0108	0.0000
XGBoost	0.0096	0.3704

(D) Private Transport vs Stationary, Walking, Public Transport.

The obtained results have been combined on the highest probability estimation. The complete statistic by class of Super Learner algorithm is reported in Table 3.6. Please note that, the coefficient columns indicate the weight of each individual learner in the overall ensemble, and the weight values are always greater than or equal to 0 and sum to 1.

Table 3.6: Binary Classification Models combination based on the highest probability estimate: Statistic by class

Super Learner Model	Stay	Walk	Private Transport	Public Transport
Sensitivity	0.990	0.662	0.857	0.927
Specificity	0.892	0.993	0.990	0.996
Pos Pred Value	0.975	0.831	0.865	0.953
Neg Pred Value	0.955	0.982	0.989	0.994
Balanced Accuracy	0.941	0.828	0.924	0.961

This is **our second result**. In this case, the average accuracy has been of **0.96**, precision of 0.865, and a recall of 0.857; with a $F_1 score$ equal to **0.861**. Therefore, the results obtained by using the Super Learner overcome those of the Extra-Trees multi-class model (see Table 3.3), compared in terms of average accuracy and $F_1 score$, and those from the literature. For each class of transportation modes, three different algorithms have been compared in terms of RCV risk. Please note, that Extra Trees has obtained the lower risk in all the binary classifications. Therefore, according to Tables 3.6 and 3.4, there are no significant differences in terms of Balanced Accuracy between the Super Learner approach and Extra Trees multi-class model, except for class *Walk* in which Extra Trees model is better ranked in terms of accuracy and sensitivity. *For this reason, the Extra-Trees model could still be the best choice.*

3.6.2 Assessing the Influence of Features

A comparison in terms of accuracy, precision and recall of the Extra-Trees multi-class approach has been computed considering four combinations of the different categories of data (as reported in Table 3.2):

- baseline features and distance feature;
- baseline, distance feature and accelerometer features;

- baseline, distance feature and temporal window features;
- baseline, distance, accelerometer, temporal features together (**Full Model**).

This set of combinations of feature categories permits to assess the flexibility of our approach in real operative conditions, where a variety of devices have to be supported, since not all devices support the full combination of categories. The results obtained by using different subsects of feature categories are reported in Table 3.7. Please note that the differences among the different cases for feature categories are substantial. The results suggest that the best choice in terms of precision is still the usage of model exploiting all the categories together, thus demonstrating that the model is flexible and resilient with respect to the device kind. Please note that the Boolean value detecting a close transportation line (i.e., proximity feature in the table) improves the classification effectiveness: the accuracy passed from 0.91 to 0.92 and higher.

Table 3.7: **Extra Tree Model results** on four classes of transport modality (stationary, non-motorized, private transport, public transport) considering four combinations of the different features

Model features categories	**Accuracy**	**Precision**	**Recall**	F_1**score**
Baseline and GPS	0.910	0.682	0.751	0.714
Baseline and GPS + Proximity	0.924	0.739	0.691	0.715
Baseline and GPS + Proximity + Accelerometer	0.926	0.814	0.744	0.777
Baseline and GPS + Proximity + Temporal window	0.949	0.805	0.787	0.787
Baseline and GPS + Proximity + Accelerometer + Temporal window	**0.953**	**0.827**	**0.869**	**0.847**

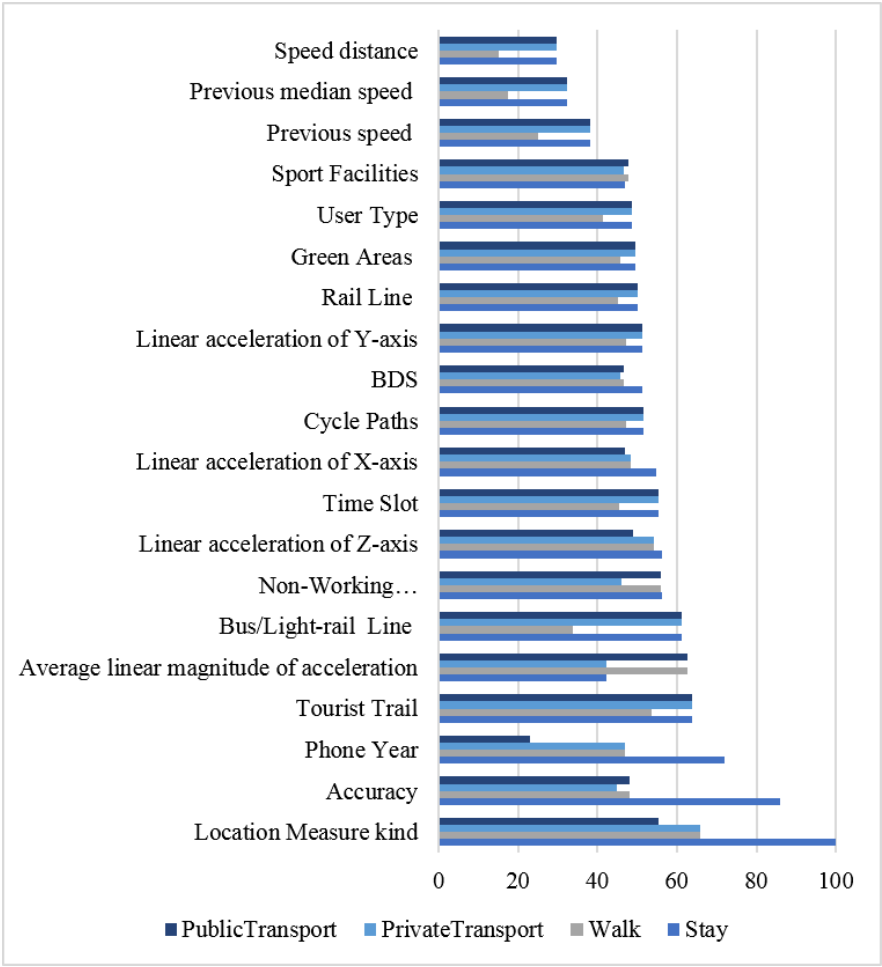


Figure 3.7: Variables Importance across the classes of the Extra-Trees full model

In Figure 3.7, the features listed in Table 3.2 are reported in order of importance across the classes for the prediction of the Extra-Trees Full Model, (the model with all the categories of covariates). The distribution of relevance suggests that the variable Location Measure kind (i.e., GPS, Network or Fused/Mixed) is the most relevant for predicting the class of transporta-

tion mode, due to the fact that in Stay mode (during the night) usually the service is kept in background (thus the terminal operating system use a specific location provider to save battery usage). The relevance of each predictor has been evaluated using the ROC curve analysis [100]. For multi-class outcomes, the problem has been decomposed into all pair-wise problems. The area under the curve has been calculated for each class pair (i.e., Stay vs Walk, Walk vs Private Transport etc.). The maximum area under the curve across the relevant pair-wise AUC's is used as the variable importance measure of a specific class.

3.6.3 Hierarchical Approach

The above presented and adopted learning models reflect the one-step algorithm methodology. In this Section, we compared those models with a two-step hierarchical approach in terms of final accuracy and testing execution time. The hierarchical approach can be considered a combination of the Extra-Tree multi-class classification and the Super learner algorithm. As first step, we can consider the Extra-Tree multi-class classifier (as reported in Table 3.3), from which the two transportation means with higher probability. Subsequently, as second step, the Super learner approach has been used to discriminate between these transportation means. A threshold has been used to decide which class can be considered directly correct at the first step: if the probability of the class is higher respect the considered threshold (0.90), the transportation modality is regarded correct without proceeding to the second step. In this case, it can be difficult to know a priori which machine learning method will work best [166], especially if the two transportation modes that have to be discriminated can variate among the different combination of transportation mode pairs: Super learner can be a solution to compare different approaches and find the best one or the best combination.

The considered machine learning algorithms are the Extra-Tree, the Random Forest and the XGBoost. As reported in Figure 3.8, the classes with higher probability respect to the threshold are not considered in the step two of the hierarchical model: the classification has been considered correct and no corrections have been made in the second step using the second learning approach (Super Learner). In the first step, the Extra-Tree algorithm have achieved a 97.6% precision for the classes with probability higher than the threshold. In the second step, six new binary classification models have been

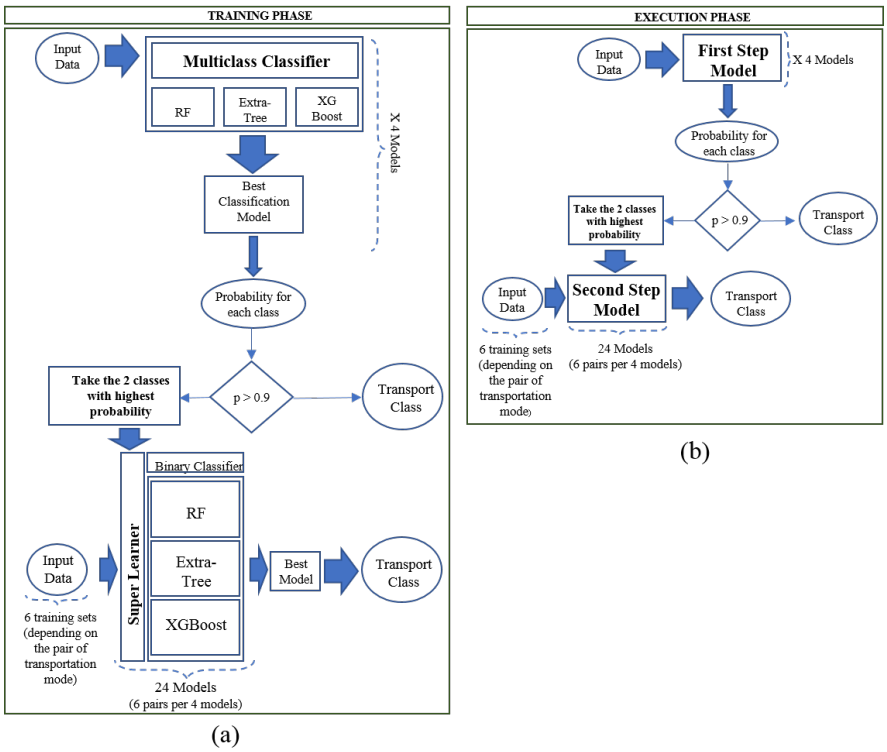


Figure 3.8: Scheme of the hierarchical approach in (a) training and (b) execution

created, one for each combination between pairs of the transportation modes selected during the step-one (the classes with a probability lower than the threshold, i.e., Stay-Walk, Stay-Private Transport, Stay-Public Transport, Walk-Private Transport, Walk-Public Transport, Private Transport-Public Transport).

Table 3.8 reports the confusion matrix related to the hierarchical approach (Extra Tree & Super Learner). Note that, Super Learner algorithms takes a weighted average of the learners using the coefficients/weights, with 10-fold cross-validation. For the two steps hierarchical approach, average accuracy is 0.94, while precision and recall are 0.786 and 0.869 respectively.

Table 3.8: Two-steps hierarchical approach confusion matrix (Extra-Tree and Super Learner considering Baseline, GPS, proximity, Accelerometer and Temporal window features)

Two-Steps Hierarchical Approach		**Predicted**			
		Stay	**Walk**	**Private Transport**	**Public Transport**
Actual	**Stay**	**0.98**	0.30	0.09	0.03
	Walk	0.01	**0.60**	0.02	0.01
	Private Transport	0.01	0.07	**0.87**	0.07
	Public Transport	0.00	0.03	0.01	**0.89**

It is also interesting to note that the average accuracy of each combination of transportation modality significantly decrease respect to the first step. This can be due to a loss of information from the first to the second step. In fact, after the application of step-one (Extra-Tree algorithm), for the classes with a probability below the threshold, the achieved accuracy is higher respect to the average accuracy calculated during the step-two (80.8% and 77% respectively). Moreover, the percentage of transportation modalities correctly classified during the step-one is of the 75%, while the remaining 25% of the test sample is further classified during the step-two.

3.6.4 Real Condition Scenario VS Hierarchical Approach Limitations

Several considerations have been already presented about the critical aspect of working on real operating conditions. Battery drainage and the opportunity to support a contextual service for the users, even with the application in background mode, drove our research, despite little decrease of accuracy and precision. We decided to design a client-server architecture to support a finer classification, using GIS data easier available on the server side (avoiding user terminal network bandwidth usage to eventually download from remote) and to support technologies to aggregate information cross-terminal and user agnostic. Implementing a central server-side classification algorithm leaves open also the chance to auto-update scenario with feedbacks

provided directly by the user. However, a real condition scenario can be affected by some limitations that cannot be solved either if a hierarchical approach is applied. This is due to the fact that the phone/user characteristics can be manifold, e.g., the presence of accelerometer information, the different type/generation of gps sensor, the presence of information related to the temporal window, etc. For this reason, the classification model has to be flexible and the training dataset has to be as much as possible various (e.g., any kind of generations, manufactures, years, characteristics, etc.) without any restriction. The application of a two-steps approach, as demonstrated in the previews subsection, may lead to a loss of accuracy due to a loss of information and can be more time consuming in terms of execution time and number of different training models. In detail, during the second step, six different training models have to be executed, one for each combination between pairs of the transportation modes (selected during the step-one), considering that the classes of transportation means are four. In addition, a specific model has to be created depending on the characteristics of the device and of the users, considering four combinations of the different categories of data (reported in Table 3.2). Therefore, 4 different training models during the first step, and 24 different training models during the second step (6 transportation modality pairs combinations per 4 categories combinations) have to be computed.

3.6.5 Final Solution

In the previous subsections a comparison between different solutions has been presented and discussed. On one hand, a two-steps hierarchical approach has been proposed. In the first step a multi-class classifier algorithm has been adopted to classify the transportation modalities. After the first classification, the classes with a probability lower than a threshold of 0.90 (prob ¡ 0.90) have been re-classified in the second step, while the classes that have a probability higher than 0.90 are considered as correct and excluded from the re-classification test set. During the second step six different binary classification model have been trained, one for each pair of transportation modality. On the other hand, a single step classification model has been presented and different models have been compared. The Extra-Tree algorithm can be considered as the best and final solution: it was found to produce the best performance in terms of average accuracy (0.953) and time consuming. In detail, four different models have been trained to make the approach as

flexible as possible. The necessity of this flexibility is because the solution has to be applied in a real condition scenario, for different phone/user characteristics, in any pseudo real-time context. The advantage of this solution is not only in terms of accuracy but also in terms of number of training models (4 different models vs 4+24 different models in the hierarchical solution).

3.7 Considerations

This research has been focused on presenting a solution to create a classification system that uses mobile devices' sensor values and GIS data (user contextual information) to identify the transportation mean of users: stationary, walking, on a motorized private transport (car or motorbike) or in a public transport (tram, bus or train). The goal has been to define a solution for sustainable mobility, delivering to the user useful personalized assistance messages. A number of metrics and features have been chosen as the *baseline and GPS*, the *distance*, the *accelerometer data* and the *temporal windows data*. The research documented in this chapter demonstrated that a one-step multi-class classifier solution was found to produce the best performance in terms of average accuracy and time consuming if compared to a hierarchical approach. In detail, the Extremely Randomized Trees exploiting all the discussed above data can be a robust approach for reliable, precise and fast estimation of transportation means. The proposed solution overcome those of the literature since it presents a solution that is capable to produce reliable results in real conditions (i.e., real-time applications and background modality of operations) with a real set of devices and in particular: (i) addressing a large number of devices providing different features, different GPS sensors, different accelerometer sensors, etc., (ii) working with time variable samples of the data that may be due to the different operating systems, energy saving setting, etc., which are not under control of the App and thus are a strong constraint to realize real applications, background/foreground modality of operation; (iii) exploiting a number of different features and obtaining results with higher precision and accuracy. For these reasons, features related to the type of phone, e.g., the presence of accelerometer, phone year, location provider etc., have been considered in the prediction model, contributing to perform corrections in the model. The prediction model proposed has been created by exploiting open and real-time data using the Sii-Mobility (national Smart City project of Italian Ministry of Research for

terrestrial mobility and transport, http://www.sii-mobility.org) solution based on Km4City infrastructure http://www.km4city.org in the Florence area, Italy. The solution is deployed as an additional feature on Smart City Apps in the Tuscany and Florence areas for sustainable mobility, which is now in place for stimulating the private mover toward a more sustainable mobility with the collaboration of three major public transportation operators: ATAF, BUSITALIA and CTTNORD. Most of the computations were conducted in R Statistical Environment (https://www.R-project.org/), and then implemented in real time.

Chapter 4

IoT/IoE architecture to enhance Smart City services of mobility and transportation

This chapter illustrates the innovative Snap4City architecture, that permits to rethink the way the Smart City infrastructures are designed, managed and used. The Snap4city solution allows to setup heterogeneous and complex scenarios that integrate sensors and actuators as depicted in IoT architecture in an overall scenario of Big Data, Machine Learning and Data Analytics. A detailed and complex case-study has been presented to validate the solution. This case study composes several building blocks of the IoT platform, which demonstrate that a flexible and dynamic set-up is possible, supporting off-grid, cloud and mixed solutions.
[1] [2]

[1] Part of the work presented in this chapter has been published as "Sii-Mobility: an IoT/IoE Architecture to Enhance Smart City Services of Mobility and Transportation" in *2019 Sensors, vol. 19* [51].

[2] *Acknowledgments:* this work was partially supported by the MIUR, with the Smart City national founding project Sii-Mobility SCN_00112, the University of Florence and the companies involved for co-founding Sii-Mobility

4.1 Introduction

Since the rapidly spread of urbanization in the 1950s across the western world, mobility drove important aspects for growth and progress. At the beginning mobility gave the freedom to the people to move around with new means of transportation, independently from where they live or they work, but today due the strong linkage of traffic and communication system and the fact that the infrastructure is stressed to their limits, it is essential to think about alternative, dynamic routes and paths and a more efficient/optimized transportation system. Smart traffic planning, innovative public transport systems and personal multi-modal routing (for the citizen of city that have already embraced this new paradigm) are central for a complete interconnected infrastructure. To have less traffic jams and near-to-zero emissions with smart means of transportation have been demonstrated to produce relevant impact on the environment and quality of life in Smart City, increasing positive virtuoso habits from the population. Nowadays, the development of key enabling technologies as Internet of Things (IoT), Internet of Everything (IoE) are driving even more rapidly the growth of sustainable ecosystem. The switch to IoT/IoE paradigm is going to change (again) dramatically the way the citizens and city operators interact with its nearby infrastructure: how they move, how they get energy, how they make decision, how the city entities are managed and controlled [195].

The development of next generation IoT infrastructure is still at its early stage and main progress are expected to be made in the next years. According to Gartner [15] by 2020 up to 20.4 billion IoT device will be connected together. New tools to support the new paradigm are needed: smart management of the resources, better security for population, healthcare and engagement of the citizen in their everyday activities are some examples of the different scenario that can be unlocked with an embedded IoT ecosystem support [128].

This chapter presents the work performed on defining a Smart City architecture to enable the integration and implementation of a set of different and heterogeneous scenario for the IoT/IoE paradigm in the context of mobility and transport, where the accent on safety critical aspects is relevant. Thus, a set of requirements and use cases are discussed and addressed, and thus, a complete experiment is detailed: a smart modality to manage a dynamic management of direction of a one-way road segment. This example employs the use of several modules of the architecture and puts in evidence how an

off-grid solution can be implemented beside a complete cloud architecture, also respecting the safety critical constraints. The problem and the solution are explained to catch the value of having a complete platform from the sensing/actuating activities towards a programmable environment where the City Operators can implement their strategies [187], in a fully distributed manner.

4.2 Requirements and architecture

Most Smart City solutions target several different aspects of the infrastructure to improve the sustainability of their services [46]. Several use case scenarios have been taken in consideration and some of them belong to very different application's domains, like healthcare management, smart mobility, citizen's security, efficient farming, etc. [214], [129]. In order to enable the City Operator, to introduce smart solutions, which are capable to help the citizens in their daily activities, a set of heterogeneous and integrated tools are needed. These solutions have to be capable to support different scenarios in a smart and integrated way, to provide a mechanism to senses the context of the city, to drive the data in a singular or aggregated way, to provide at data scientists a set of tools for data analysis and a programming environment to computer scientist. A Smart City platform should be capable to inform the citizens and City Operators, to collect their feedbacks about the city infrastructures and services, to provide an integrated info portal for ad-hoc and personal visual applications (Dashboards) and finally to provide to the platform's admins a way to monitor the functionalities of the solution.

In the context of mobility and transport, in wider view of mobility and transportation aspects of the city, the Intelligent Transport Systems (ITS) should be capable to manage semaphores, dynamic signages, speed control systems, smart parking, car sharing, etc. The revolution of IoT may also impact on mobility and transport applications, in distributed manner. An IoT platform for a Smart City must support scenarios that focus on the way the citizen interacts with their infrastructures (in terms of streets, road, cycle path, public transport station, parking, etc.) [109], the way the City Operator of the mobility and transport unit of the city programs, configures the transportation infrastructure and how it is managed/monitored daily.

The main requirements we identified in the above described context are reported as follows. So that the solution has to be capable to:

- collect a big amount and heterogeneous data from the city infrastructure [101] in a continues way and support different ways for data injection (push, pull, on-demand, offline); support off-grid scenario and completely on-cloud architecture;

- assure an architecture that is complete and consistent due to the fact that some scenario can be safety-critical, and thus the communications among IoT elements, and with the Control Room (traffic congestion area, event mass moving, pedestrian cross paths), have to be safe; as well as their implementation has to be capable to overcome the cases of missing communication; in same case real-time or near-real-time safety mechanisms have to be provided [172];

- guarantee that the solution is distributed and scalable with respect to the points of controls, and that they may be capable to continue to work even if the connections with the Control Room, or each other are lost; the Control Rooms may be used to force situation and monitoring;

- guarantee that the data are managed properly, in respect of their ownership and the explicit signed consents they gave; protect for external intrusion and data tampering, man-in-the-middle attack and data sniffing; monitor suspect activities and enable inspection from law enforcement agencies; inform in case of a breach [98];

- store these data in a distributed storage and index them on the basis of different constraints imposed by the domain application [192]; support an efficient way for data retrieval and provide a complete system of backup and disaster recovery;

- provide a programmable environment to define algorithms for machine learning and data analysis, able to extract characteristics from the collected data (aggregation of data) [167] and elaborated them in order to implement predictive capabilities for guessing their future evolution: smart parking, predictions on traffic flow, traffic flow reconstruction;

- enable visual presentations of data via a set of dashboards composed by widget components that a City Operator can manage graphically; these widgets have to be manageable/configurable by a person that may not know deeply the technicisms behind the IoT infrastructure.

In the context of mobility and transport, the data exchanged and managed by IoT devices, toward the infrastructure in which the local computing capabilities, can be used via an IoT Edge (IoT device with enhanced capabilities of local computing). So that the features that those devices should be requested to locally address have been identified, and are reported as follows:

- dynamic signals on the road with respect to the conditions of the local sensors; display information to drivers and to people on the street: speed limits, presence of active/no-active smart speed meters, digital road display about the traffic condition and best solution for public transportation;
- collect information about traffic's flow sensors on the city road (how many cars in last time slot passed in a specific segment, mean speed of them, density of big truck, cars, motorbikes, etc.), estimate the traffic status in terms of density of equivalent vehicles, thus abstracting from vehicle kind;
- people flow moving in crucial areas based on cameras, laser, Wi-Fi, Bluetooth by counting sensors density of people. For example, placed in front of a bus/train station, in front of an important museum; counting the people flow in some strategic area of the city, in front of gates or around busy common path walk [162];
- implement change directions of traffic on the road with respect to the local conditions or on demand.

While other activities may need to be performed/delegated remotely on cloud in a mobility and transport central station and control room such as:

- predictions of: free car parking slots, delay of buses at bus stops, bikes to be shared from a given point, traffic flow in some points;
- display information to operators in Control Room to alarm in the case of early warning conditions or for current operations;
- traffic flow: catch and visualize the current global traffic situation, predict congestions, suggest alternative ways to reach a point of interest [58];
- video analysis: suggest alternative paths for tourist to avoid congestion areas (long queue) and to promote new and alternative tours;

- public transport: describe in a visual and integrated way the complete public transport infrastructure, enable queries on maps (when a bus will arrive at destination, how long does it take to commute to work) and support multi-modal routing (from here to there using different kind of public transportation means); integrate information of traffic flow to provide a better prediction;

- parking: track the availability of city parking areas, their weekly/daily use, suggest a free parking place in the area a citizen usually visit [53]; predict a parking place in a future assessment or parking area without any restriction due to street cleaning, public or private events;

- user behavior and engagement: analysis the citizen behavior in the use of the city infrastructure, highlight the behavior not sustainable and propose a switch of habits to promote virtuoso ones [50]; track if the suggestions are followed and rewards the "good" citizen;

- actuator: manage smart traffic board for dynamic setup or to inform the citizen about congestion when disruptive events occurs.

According to the above identified reasoning, the architecture designed and implemented resulted to be flexible enough to support the above requirements and use case scenarios, on IoT edge scenarios (off-grid) and cloud (with control rooms). In the context of this chapter, we concentrate the attention to IoT Edge cases with some integration to cloud computing and control room. Figure 4.1 highlights the architecture on the basis of its major macro-blocks.

Starting from the left upper part, the Real World and the environment are inter-connected to our infrastructure via a set of sensors and actuators. Thus, a multitude different of devices are supported and can be also classified on the basis of their energy consumption and distance in terms of network hops to our main backbone. A sensor can be implemented by using a simple microchip (i.e. esp8266 with microcontroller capabilities) that communicates a set of different data values via messages with some protocol and format. It can implement a button that can be connected to our backbone or to an aggregator device (IoT Edge) via a wireless network interface towards an Internet access point with standard communication capabilities (IEEE 802.x), or with patented technologies (such as LoRa, SigFox, OneM2M, ...). Most of this kind of sensors have no or low capabilities of embedding logic,

and are just used to collect and transmit data to the platform or to perform some action on the basis of the data received. On similar capabilities, a microcontroller board (i.e. Arduino) can be used: it adds a slightly bit more dynamic setup, with aggregation capabilities between different type of sensors and actuators. These IoT devices are low energy consumption and usually works using strategies for sending data when they are significant or on-demand.

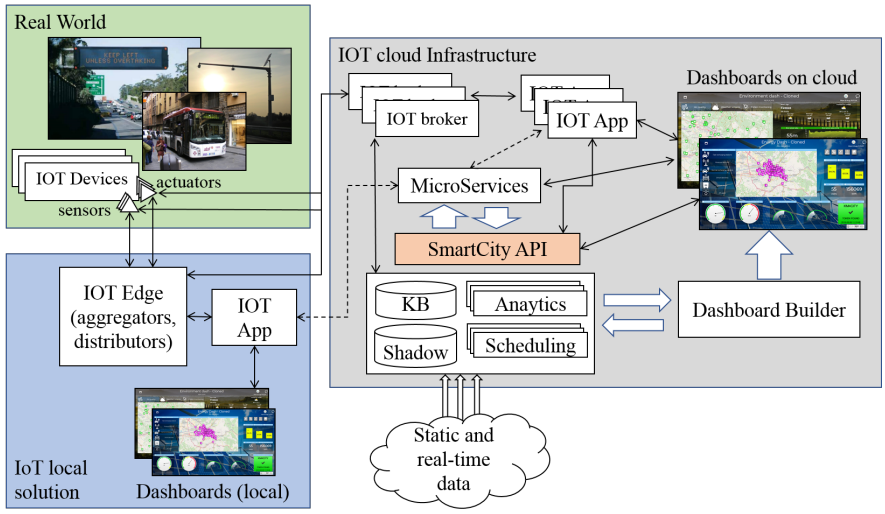


Figure 4.1: High level architecture of our IoT solution

The data generated from the sensor or received by an actuator can be sent/received to/from an IoT Edge that act like an aggregator/distributor and that introduce the possibility to define some kind of logic (IoT Application) to enable off-grid solutions. IoT Edge are more capable devices and they are in substance small single board computers (i.e., Raspberry PI). They can be directly connected to a wireless network via a more powerful linkage, using little more energy, and can be installed where a remote powering is possible. They can act as sensors/actuators, as well as aggregators among the lower level former IoT Devices or as router towards cloud IoT Smart City backbone. Handheld personal computer (i.e., smartphones, tablets) can be regarded as IoT Edge devices and are also supported: they are similar device

to the previous class, but with an embedded network connection to the Internet via 3G-5G capabilities, so they can act and communicate directly to the Smart City infrastructure, even if the energy consumption has to be taken more in consideration; more generally any personal computer and Internet devices can be integrated in the Smart City IoT/IoE solution, whenever energy consumption and price is not an issue. Finally, a virtual sensor and actuators can be also created directly on cloud infrastructure for offering at the Operators and user to send messages and show results on some user interface. So that, a button on the user interface of an IoT Applications can be regarded as Virtual IoT Devices generating a message to an IoT broker. In IoT Edge raw data (or the elaborated ones) can be represented locally (in terms of closeness to the sensors/actuators) via local Dashboards.

In some cases, the logic (IoT Application) can use external services formalized as Microservices and exposed by the Smart City platform. They are remotely exploited and, off course, when the Internet connectivity is lost, the application logic has to adapt the logic to work with the internal data and services according to its purpose. The Microservices provide Smart City infrastructure are implemented using the Smart City API [152] that work directly on top of Km4City Knowledge Base and a set of additional tools (for sake of readability, here we just presented two main macro-blocks: analytics and scheduling; for a more detailed description refer [62]).

In case an on-cloud solution it is chosen, the data can flow toward/from Smart City IoT infrastructure by means of:

- a set of Context Brokers (Mosquito MQTT, Orion NSGI, RabbitMQ, etc.) which collect the data and allow other IoT application to make subscription to services;
- IoT Brokers towards a shadowing and indexing system to enable the collection of all changes performed on IoT Devices sensors/actuators and enable those data to be accessed by means of Smart City API for the IOT Applications, Data Analytics and Dashboards; This means that when the IoT devices goes off-line the system is capable to provide the last value and also the historical values that can also be used for publishing then as new data sets on Open Data Portals, automatically;
- IoT Edge devices on the field (Off-grid) which may have IoT applications inside also working with MicroServices and Dashboard Local or on Cloud via MicroServices;

- IoT Applications on cloud, which may be subscribed to IoT Brokers data, exploit MicroServices and thus services about IoT data Shadowing and Indexing, Data Analytics, Scheduling or processes, and Dashboards.

Several other management tools are available to the IoT Application programming logic in the backbone of the solution. In the context of this chapter, the Dashboard Builder interfaces the IOT Application to the visual presentation of the data (Dashboards on a cloud). This module enables the City Operator to compose in a graphic dynamic and intuitive manner the visual presentations of data using a set of widgets that can be defined on top of raw or structured data. The collection of graphic widgets can be easily extended by developing specific PhP modules. The Dashboard Builder as the whole infrastructure and tools are Open Source.

The way the sensors/actuators are connected to the overall structure [65], which IoT Infrastructure's services are used by the IoT Application programming logics, and what is the best topology to connect the sensing to the elaboration side, strongly depends on the target scenario to implement. And, as well as, on the energy availability around the sensors, on the presence of an Internet connectivity (and its relative QoS) and on the required security (sensibility of the transmitted data).

The proposed IoT architecture of Sii-Mobility on cloud is the backbone of the solution. Completely virtualized on the Internet, it is the boilerplate where a City Operator interacts via a set of graphical tools to program algorithms for data aggregation, data analytics, and thus realizing solutions for computing predictions, anomaly detection, origin destination matrices, trajectories, etc. The data produced by this calculation can be also injected back in the solution knowledge base for further analysis and exposed to the personal dashboards for graphical presentation.

Important to note, the data generated from the IoT devices are linked to the information about the user that own the device, so the platform is able, whenever in any tools of the ecosystem, to enforce and assure that a data is threaded properly and presented just when/where the owner of the data gave his explicit consents. The data from a sensor are initially set private of the owner and thus he/she the only one that can visualize and utilize the data. Techniques of identification and authorization are implemented in any tools of the proposed solution exploiting existent service when it is possible (i.e. to provide a user single point of authentication we employ the use of the open-

id connect protocol) or implementing new ad-hoc solutions where it was no available (i.e., to verify the device authorization to access an Orion broker (for example for a subscription) via a programmable firewall compliant with NGSI protocol). In the communication all data are encrypted to assure no other people can access the sensible information, and the connection protocol is also signed by a certification authority to prevent tempering (identity stealing). Thus, any transmission between the tool's component is protected using secure channels. Rule management for different kinds of rights and organizations in groups for classification of the users permit a flexible architecture to better organize the permissions in the overall managements of the user privacy. Finally, a delegation system is also available to permits the users to expose their data (providing specific right consent). The right management is enforced at level of IoT Devices and single IoT data (delegated device/datavalue), and can be delegated to other users, or to public (anonymously). More details about protection and privacy management on Device and Personal Data are presented in the Chapter 5.

4.3 Experiment key study

In this experiment, a one-way variable road segment is taken into consideration: a road segment in which the direction of traffic flow can be dynamically inverted at any time according to the needs of the City, under the decision of the City Operator managing the section. As depicted in Figure 4.2, the solution consists of a number of elements to manage the traffic and their management by means of a sect of interconnections. The main requirement, of course, is to be able to reverse the direction of travel without causing any accidents, putting in critical condition the drivers on the road.

The system developed consists of two Smart Gate located on both sides of the segment to be managed, realized with a traffic light flanked by a display where information is shown to the drivers. The Smart Gates are controlled and connected to an IoT Edge each (they could be Android Device as well as Raspberry Pi or Arduino) that contain the IoT App logic to control the red-light, and the messages showed on the dynamic plate display (informing the that road cannot be accessed, or providing speed limit, etc.). The two IoT Edge Devices, in turn, are connected to a Local Supervisor IoT Edge device (also an Android Device) in the box of the Smart Gate system (usually on the side of the road segment) that can be used by the authorities to reverse the

travel's direction of the segment under consideration. This Local Supervisor shows a simple and intuitive visual interface via a Dashboard (see Figure 4.3) and allows the reversal of the road with a simple press of a button in the interface.

The information coming from the Smart Gates through the Raspberry Pi and the commands sent from the Local Supervisor must have a high reliability and the City Operator must be sure that what he/she is seeing in the Local Supervisor is what the Smart Gates are showing to drivers at that precise moment. For this reason, the connection between the elements of the system are preferably performed by wired to guarantee higher reliability, but connections 5G would be a good replacement for the future.

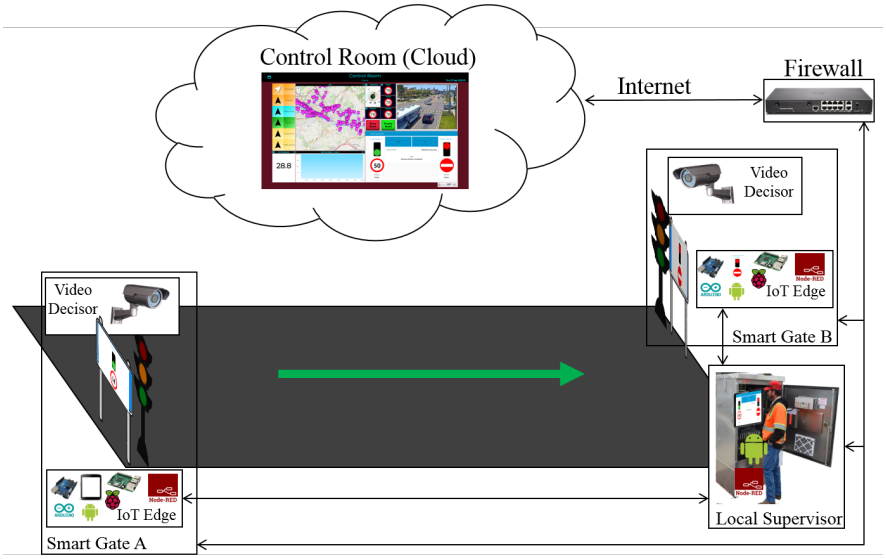


Figure 4.2: Configuration of a one-way variable system on a road segment

Depending on the geometry of the road segment, one or more cameras have to be installed, to be used by the Video Decisor subsystem to decide whether at a given time it is possible or not to complete the reversal operation; by controlling if the road is free of vehicles or whether the road is busy. In that case, the system has to wait until it become empty. The Smart Gates always have a complementary aspect: free access on one side and no access

on the other. When switching, it is needed to check that the road is empty: the safety of people is at stake, so the software cannot be decided itself on it, but needs active human help.

Figure 4.3: The interface (Dashboard) used to reverse the direction of travel on the road segment

Clearly, the decision to reverse the one-way direction of a road can only be implemented safely if there are no vehicles in transit at that time. For this reason, a Video Decisor is used to detect the state of "Empty Road". This module is not directly connected to the integrated actuator but to the Control Room for safety reasons. In fact, it is not possible to rely on the judgement of an intelligent electronic device to acquire information that, if incorrect, has a direct impact on the safety of people. The information must necessarily be confirmed by a human operator on the Control Room.

For these reasons, the Video Decisor communicates directly to the Control Room, transmitting visual information based on which the operator can confirm and authorize the switching. The overall operational sequence of actions consists of:

- setting up of both actuators at the ends of the road in a state of "Prohibited Access";
- wait for confirmation of successful switching;
- wait for the "Empty Road" information from the Video Decisor and the Control Room;
- switch one of the two integrated actuators to the "Access Allowed" state.

These operations are described in detail in the sequence diagram in Figure 4.4.

The first operation that starts the data flow is done by the operator who presses one of the two reverse gear buttons (at the top with the arrow shown in Figure 4.3). The system (1) checks if the connection with the Smart Gates is active and (2,3) records the target status that the Smart Gates will have according to the button selected by the operator.

It (4,5) saves locally the status "Prohibited Access" for both the Smart Gates and it (6, 12) sends the "Prohibited Access" signal to both Smart Gates and waits to receive from them an ACK message with the status they intend to change: in this case a "Prohibited Access" ACK must return (7,13). Once this ACK has been received, the system (8,14) responds with a second ACK containing the status "Prohibited Access".

If any ACK is lost, a message is shown on the screen indicating the possible inconsistency of the Smart Gates respect to the Local Supervisor and the Smart Gates show "Prohibited Access" until they receive another command and the "Smart Gate Change" protocol starts again.

If all the ACKs arrive at their destination, the Smart Gates (9-10, 15-16) turn to status "Prohibited Access" and they send a notification of the change made (11, 17, 18). The Video Decisor waits for the video to notify that the road is empty by the Control Room (19,20). When the "Empty Road" signal is notified (21), the Local Supervisor (22) reads the target status that the Smart Gates should have, in accordance with the operator's choice, and the "Allowed Access" signal is sent to the correct Smart Gate.

The same "Smart Gate Change" protocol as described above is started with this target Smart Gate (23-30). If the ACKs reach their destination, the Local Supervisor notifies that the reversal has been successful. If any ACK is lost, a message is displayed on the screen indicating the possible

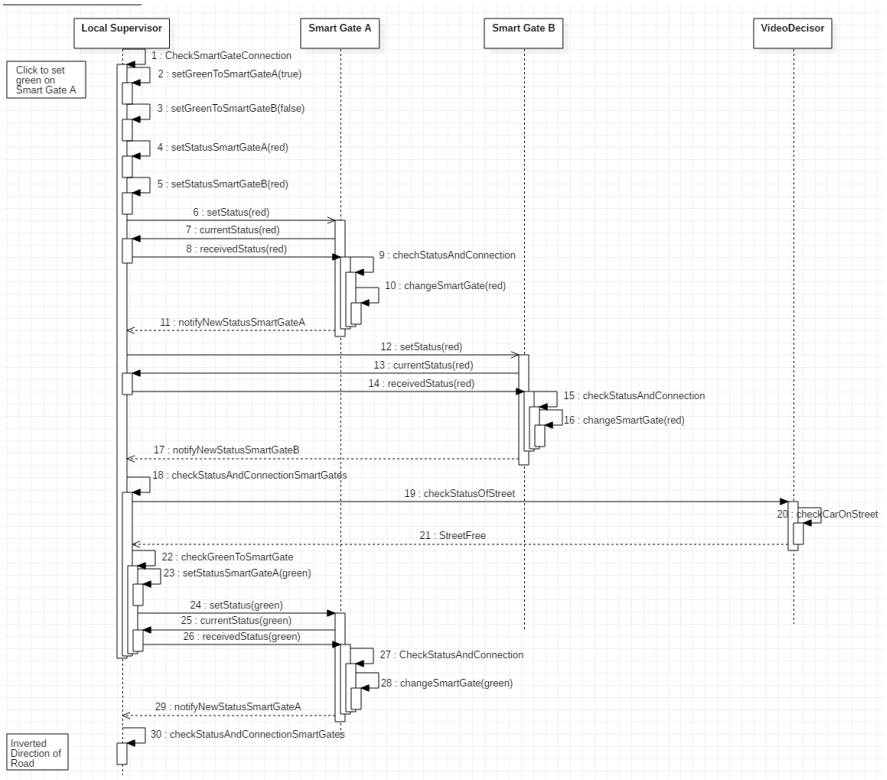


Figure 4.4: Sequence diagram of operations that are performed in the phase of reversal of the road

inconsistency of the Smart Gate respect to the Local Supervisor and the Smart Gate show "Prohibited Access" until they receive another command and the "Smart Gate Change" protocol starts again.

In addition, to increase the security of the reversal protocol, every two seconds the Smart Gate send their status to the Local Supervisor and if this does not arrive, the City Operator is informed that the Smart Gate that did not send the status is currently, and it is not online anymore, to the Local Supervisor.

To avoid that the status is temporally misaligned with what was present at that time on the Smart Gate a timestamp is also added to the message. If

the timestamp has a misalignment of more than 5 seconds, then the Smart Gate status may be inconsistent with what is present on the Local Supervisor and a message is shown to the City Operator.

The protocol described above has been implemented in the Smart Gates and in the Local Supervisor through the Node-Red platform via a Node-Red flow (Figure 4.5) which can be divided and explained in sub-flows (Figures 4.6 for box (a), 4.7 for box (b) and 4.8 for box (c)) for better clarity.

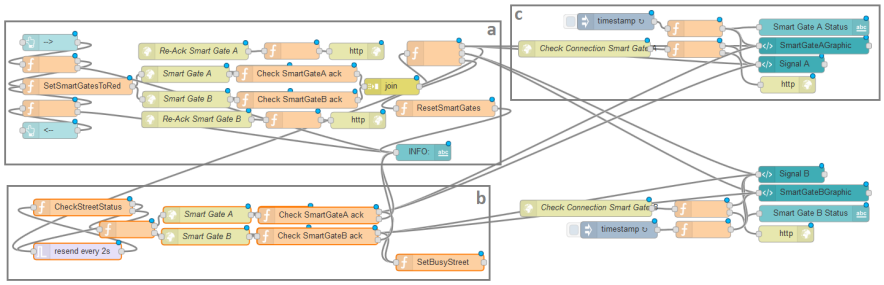


Figure 4.5: Complete flow of Node-RED that allows to realize the "Smart Gate Change" protocol

The first sub-flow segment (see Figure 4.6) allows the Local Supervisor to inhibit access from both sides of the road before allowing it on the chosen Smart Gate.

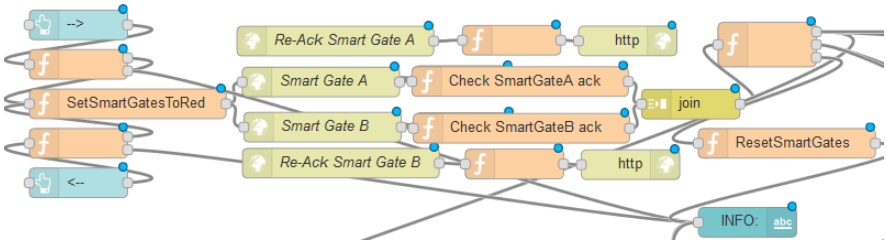


Figure 4.6: First sub-flow that allows the City Operator to set on both Smart Gate a "Prohibited Access" message. The entire flow is shown in Figure 4.5

The flow starts with the two blue nodes on the left that correspond to

the buttons (present in the interface of the Local Supervisor) that allow the City Operator to reverse the direction of travel. Depending on which of the two buttons is pressed, the first operation carried out by the following two orange nodes is to check that both Smart Gates have sent their signal within 5 seconds from when the button was pressed (1). If this is not the case, a message will be displayed indicating "Connection Problems" to the Smart Gates. If there are no problems the information, about which of the two Smart Gates will allow access after being inhibited by both Smart Gates, is saved (2,3). The following node sends the "Prohibited Access" signal to both Smart Gates, waiting in response for the ACKs containing the status that was sent (4-17). The last block (18) checks that both ACKs have returned, otherwise it signals connection problems between the Local Supervisor and the Smart Gates. If the ACKs are returned but not correct, an inconsistency error is reported, and the Smart Gates remain in the "Prohibited Access" state waiting for a check or for a new signal to be sent that can be successful.

If the ACKs have been returned and are both correct then the flow continues, going to check if the state of the road is free (19). The check is carried out every 2 seconds and if it is successful and the road is empty (20,21), the status, that indicates which of the two Smart Gates should allow access, is recovered (22). A signal is sent to the right Smart Gate with the status of "Access Allowed" and is checked again that the ACK returns and that it is correct, if this happens the message that the reversal is successful is shown (23-30). If this is not the case, a message notifying the likely inconsistency between the Local Supervisor and the Smart Gates is displayed. This inconsistency should be cleared by periodically sending the statuses running the Smart Gates to the Local Supervisor every two seconds.

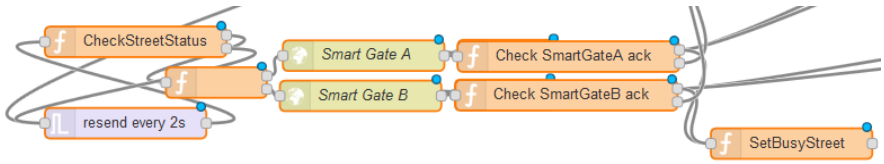


Figure 4.7: Second sub-flow that controls whether the road is marked as free and sends the "Access Allowed" message to the correct Smart Gate. The entire flow is shown in Figure 4.5

In this second sub-flow segment (see Figure 4.7), it is possible to notice

the nodes that have a “Re-Ack” at the beginning of their name, through these nodes it is possible to receive the ACK and to resend the known status and the timestamp of the Local Supervisor to the Smart Gate (receivedStatus of the Sequence Diagram shown in Figure 4.4) that will answer with the last ACK (the one checked in the nodes whose name starts with “Check if”, notifyNewStatusSmartGateA in the Sequence Diagram shown in Figure 4.4).

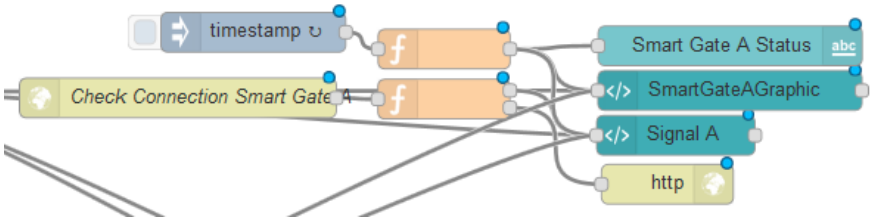


Figure 4.8: Third sub-flow that controls the consistency of status between the Smart Gates and the Local Supervisor. The entire flow is shown in Figure 5

The third flow segment (see Figure 4.8) implements the part that maintains the consistency between Local Supervisor and Smart Gates (here is shown just a synthesis of the flow for Smart Gate A, but the flow for Smart Gate B is identical). Every two seconds the Smart Gate A notifies its status to the Local Supervisor through a direct call that arrives on the node “Check Connection Smart Gate A”. The data that arrives is the status of the Smart Gate and the timestamp of when the message was created: the status is sent to the dashboard to maintain or recreate consistency and the timestamp is saved. Periodically, the lower part of the flow checks that the timestamp sent by the traffic light and the one saved is no older than a predefined time. If this time is exceeded, the Local Supervisor interface (Figure 4.9) will notify the City Operator that no longer a connection with the Smart Gates exists and it will remove the signals related to the latter because we cannot know if they are any more consistent.

The last flow segment (see Figure 4.10) is used on the Smart Gates for the logic complementary to the one written above is as follows. From the first node at the top the calls are made by the Local Supervisor when the status of the Smart Gate must be changed. When a status change message

arrives, the new status is saved and is sent to the Local Supervisor to make sure that the message has arrived correctly. If the return ACK contains the same new status as the one already sent in the first message, then the Smart Gate changes its status. The bottom flow checks every two seconds if the connection with the Local Supervisor is active. If not, the status is set to "Prohibited Access" until City Operator intervention or resolution of the connection problem.

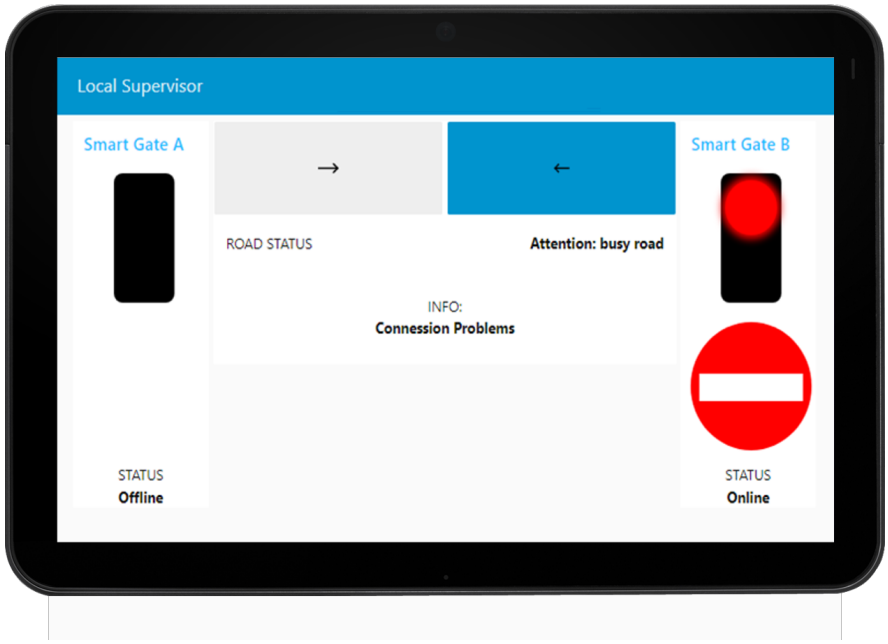


Figure 4.9: This figure shows the Local Supervisor interface in case of inconsistency and following a connection problem with the Smart Gate A

The system described above allows to control the travel's direction of the road being in front of the Local Supervisor and does not allow any remote control. Using the tools developed and made available by the Sii-Mobility architecture, it is possible to create a simple and intuitive dashboard as shown in Figure 4.11. Through this dashboard it is possible to remotely control the reversal of the considered road, having much more information at our disposal than if in front of the Local Supervisor. It is possible to

have the real-time images available from the Video Decisors to make sure that the road is free during the reversal, it is possible to modify the signals shown in the Smart Gates and it is possible to interact directly with the Local Supervisor to launch the command that starts the reversal protocol described above.

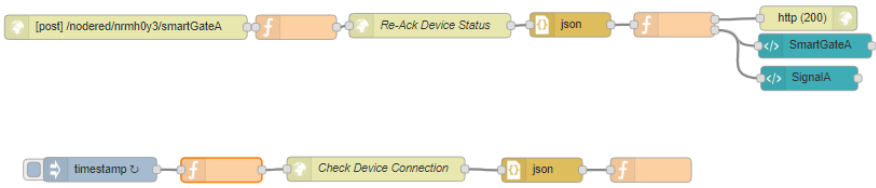


Figure 4.10: Flow that realizes the logic of the Smart Gate. The upper part is the one that receives the status and changes it if the ACK exchange is successful. The upper part is the one that checks if the connection is always active between the Local Supervisor and the Smart Gate

On the left of the image in Figure 4.11 we can see how it is possible to retrieve information from all the static and real-time data saved by the Sii-Mobility ecosystem. A map can be inserted with points of interest that can be added or removed dynamically by pressing the buttons on the left of the map. For example, in the case of the study presented here, the operator in the Control Room can show the position and the real-time values of the traffic sensors in the entire area affected by the change in gear of the road with the system installed. Moreover, for each single sensor it is possible to obtain, in addition to the real-time value of the sensor, also the historical data up to one month before the request. Once the choice has been selected, the trend of that sensor in the selected period is shown in the bottom panel. The use of this Widget is done through a Dashboard Wizard (Figure 4.12), that utilizes the services provided by a Dashboard Builder module and allows the user who is creating the Dashboard to decide which are the services he wants to see inside the map and on where the area should be centered.

In the top right corner of Figure 4.11 it shows the signal received by the camera part of the Video Decisor subsystem and based on which it is possible for the operator to decide if the road is free or not. Next to the video there are two buttons that allow the operator to send the signal to the Local Supervisor whose interface is shown in the lower part of the Control Rooms.

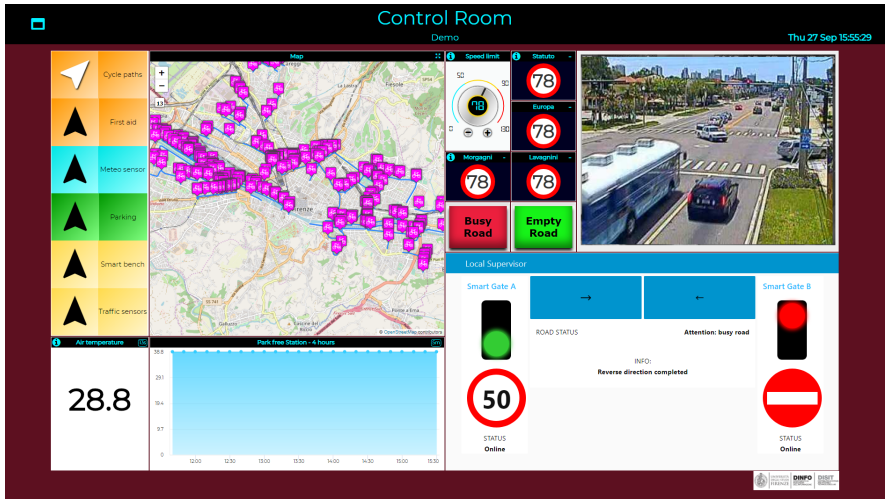


Figure 4.11: The Control Room shows a large amount of information and actions that the City Operator can do through this Dashboard. It allows to have static and real-time information (including historical data) of the POIs that are in the part of the map visible on the left. It allows to send data to the logic developed via Node-red through direct calls (Local Supervisor view), through web sockets (buttons "Empty Road" and "Busy Road") or through IoT brokers (Dimmers and road signs)

The connection with the interface is sent to an Android device located in the box on the side of the road segment and the operator in the Control Room can act as if he was in front of the Local Supervisor. In this way, if the connection is lost, it is not possible to send signals to the Smart Gates as they cannot access the interface.

The "Empty Road" and "Busy Road" buttons are part of the Video Decisor subsystem. The importance of these buttons is to make a dashboard in the Control Room interactive: it allows the City Operator to send commands that can be read and, basing on the value that these commands have sent, to execute the linked logic. These nodes were added to the dashboard directly by creating the flow on node-red. In fact, using the blue nodes that can be seen on the left of Figure 4.13, the user can decide in which of his dashboards add buttons that, if pressed, will send a message in the flow of

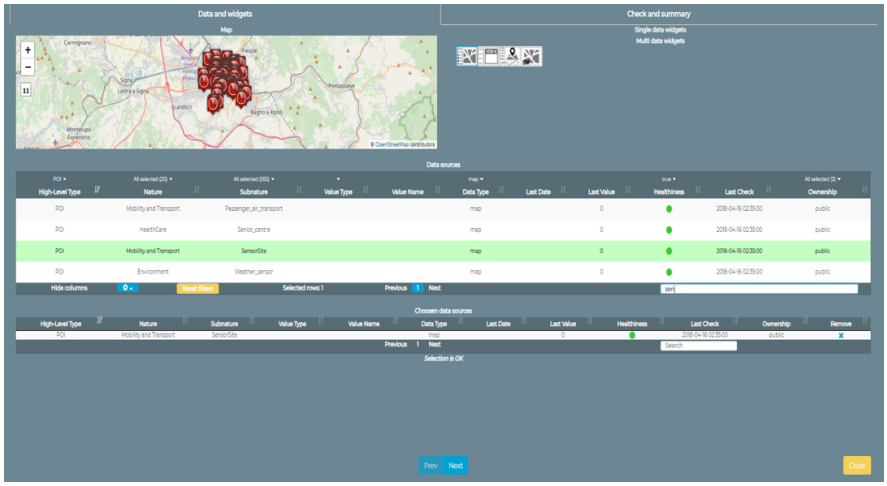


Figure 4.12: Creation of a widget (the map with the selectors of the various POIs) through the Wizard. In the example, traffic sensors are selected as POIs

node-red outgoing from the respective blue nodes. When the flow is executed the button appears on the selected dashboard and sends data to the node inside the flow.

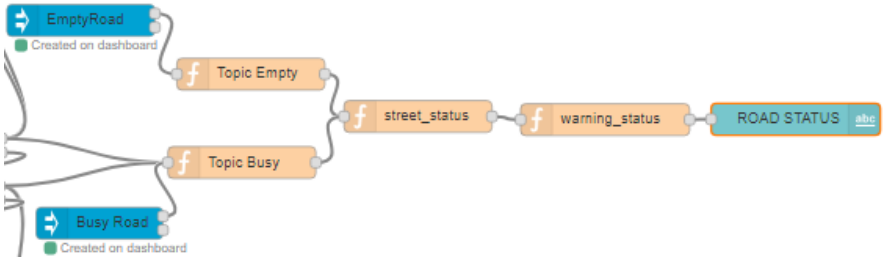


Figure 4.13: Forth sub-flow that allows the interaction of the dashboard with Node-red. Through this flow, data can be sent from the buttons in the Control Room to the logic that resides in the Local Supervisor. The entire flow is shown in Figure 4.5

In the development of the subsystem of the Video Decisor the flow, that follows the reception of data from the buttons, controls which of the two buttons has been pressed and saves the value of free/busy road. It will be used to allow the change of state of a chosen Smart Gate.

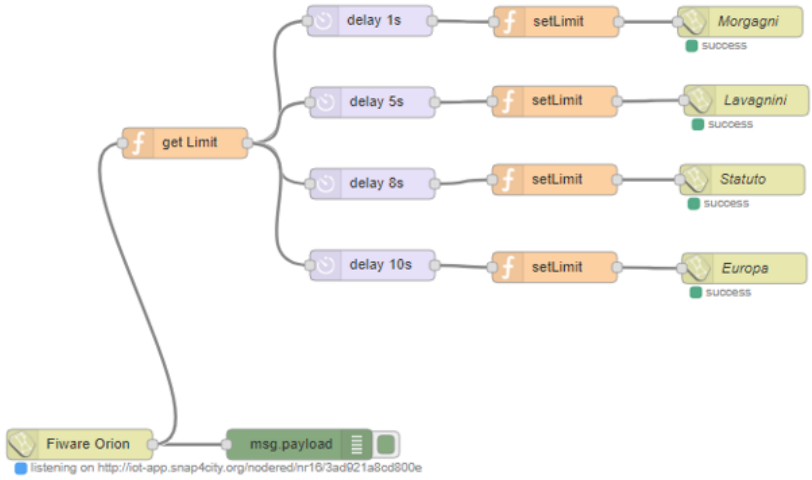


Figure 4.14: Flow that allows City Operator to read from an entity connected to the Dimmer and write these values to four other entities connected to the road signs in the Control Room. These entities reside within an IoT Broker

Moreover, in addition to the impulsive buttons, other actuators can be inserted in the dashboard for example a Dimmer: an example is shown in Figure 4.11 that present a button that is between the 4 road signs and the map. This Dimmer is connected to an entity on a Context Broker and when it is turned around by the users, the value of the connected entity changes. The detailed flow in Figure 4.14 allows to read the value of the entity associated with the dimmer in Event-Driven mode and write the 4 entities associated with road signs in the dashboard that will change their value based on that set by the Dimmer.

4.4 Considerations

The way the user moves around the city dramatically changed in the last

twenty years and the increase of integration between the city infrastructure with the sensors/actuators that are able to act-on/catch its status is enabling nowadays a wide range of new application to be built on top of new IoT architectures and platforms. A set of requirements, these architectures have to be satisfied in terms of smart mobility, has been identified and a brief description of the logical separation between local computation (off-grid solutions) and cloud elaboration has been presented. The major building block of our IoT solution has been explained and a brief description of the main functionalities of the tools have been highlighted. A complete and real problem has been used to validate our IoT solution and its modularity: a road segment in which the direction of travel can be reversed at any time according to the needs of the City Operator that manage that section. We described all the problems related to its automation and how it's possible to employ our IoT building blocks to enable a secure workflow, integrating a local computation with a cloud solution. Some snaps of the logic of the IoT application has been presented, to highlight the simplicity of the programming environment supported.

Chapter 5

Smart City IoT Platform Respecting GDPR Privacy and Security Aspects

This chapter describes several aspects of the Snap4City platform regarding its privacy and security support respecting the GDPR of European Commission. Due to the potentially very sensible nature of the managed data in an Smart City platform based on a IoT architecture, privacy and security aspects have to be taken into account by design and by default. In addition, an end-to-end secure solution has to guarantee a secure environment at the final users for their personal data, in transit and storage, which have to remain under their full control. [1] [2]

[1] Part of the work presented in this chapter has been published as "Privacy and security aspects on a Smart City IoT Platform" in *19th IEEE International Conference on Scalable Computing and Communication (SCALCOM)* [57] and has been submitted and is currently under review as "Smart City IoT Platform Respecting GDPR Privacy and Security Aspects" for *Journal IEEE Access* .

[2] *Acknowledgments:* this work was partially supported by the European Union's Horizon 2020 research and innovation program, with the "Select4Cities" PCP project (within which the Snap4City framework has been supported) under grant agreement No 688196, the University of Florence and all the companies and partner involved in Snap4City

5.1 Introduction

IoT is becoming a disruptive technology, especially for city users of metropolitan areas. The pervasiveness of IoT Devices, integrated in common objects, is becoming increasingly deeper. The addresses' space for these devices would be enough to point any sensors of any devices at any moment without restrictions. Diffuse products that implement Low-Power Wide Area Networks (LPWAN) technologies for IoT introduced by SigFox and Semtech (LoRa, Long Range) have been gaining interest and have been under intense deployment campaigns worldwide [210]. At the same time, short range IoT devices (based on technologies such as IEEE 802.15.4 or Bluetooth Low Energy, BLE [114])) are sold in increasing quantities and are already able to support scenarios for smart homes, energy metering and industrial automation. On the other hand, the start of the diffusion of 5G devices and services is creating high expectations in networking IoT technologies, as the killer application of previous technologies in metropolitan areas. With a high peak data rate, high connection density and better network energy efficiency, the 5G standard also addresses ad hoc specification for the M2M (Machine 2 Machine) use cases. With the help of the IPv6 (Internet Protocol version 6) address schema, the 5G standard will be capable to deliver a complete integration of IoT Devices [138].

A wide variety of objects (e.g., smart bulbs, IP (Internet Protocol) cameras and alarm clocks) are currently part of the user's environment, relying on the modem and hub devices of the user homes to connect them over the Internet. The IoT infrastructures permit the realization of a more integrated scenario where real world and smart devices complete each other and permit management with less human intervention: open communication schema supporting different standard protocols (e.g., MQTT over TLS, Message Queuing Telemetry Transport over Transport Layer Security) enable devices to connect each other and to exploit cloud-fog infrastructures [65]. In terms of security and management of data privacy, the complexity is growing, not only for the needs of establishing secure connection but also for the data management and access rules [97].

In most cases, the communications from IoT Devices and IoT Brokers is established only by using simple and not mutual authentications. The IoT Brokers are gateways in which the IoT Devices are registered, and by which they can be managed (for example, for firmware updates, for sending massive reset commands, etc.). In addition, IoT Devices and IoT Brokers could

communicate with IoT Edge Devices that are entitled to perform some computation up to machine learning [137]. IoT Edge Devices are typically located on premise and have additional complexity in terms of secure communication with other tools on the cloud. In fact, IoT Edge solutions typically need to send/get data on/from the cloud, and directly on/from dashboards [61]. Moreover, IoT Edge devices could also be enabled to manage/process private data. Most of the IoT frameworks on the cloud provide structures and processes that may coordinate and control several different IoT elements, in the field and on the cloud as well. In certain cases, the IoT solution may be limited to on-premise solutions and tools without the need of having a cloud counterpart.

The IoT frameworks may present a set of modules, rules and algorithms that organize the ways in which data processing is performed and managed among involved entities (i.e., devices, edges, processes on cloud/containers, dashboards and, users). Even more, IoT Platforms may go beyond frameworks and may need the implementation of the so-called IoT Applications that permit definition of user-component-logic, at the same time, hiding the complexity of the infrastructure (of the IoT Frameworks) [42]. Furthermore, the IoT Application processes may manage open and private data and may produce results that are of a private nature of some user(s). Please note that IoT Applications may also be deployed on IoT Edge Devices (see Figure 5.1).

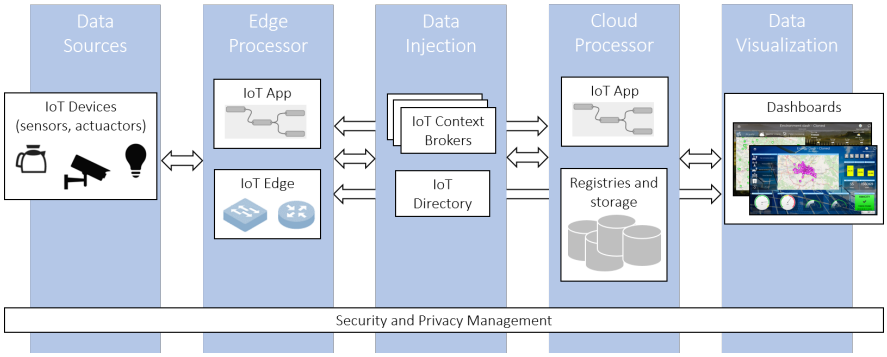


Figure 5.1: IoT Platform General Architecture

The information sent/received to/from the IoT Devices could be very sensible and private; therefore, protection and cryptography techniques are

diffusely implemented. All steps of a secure communication must be guaranteed as well as solutions to protect data during all of the life cycle, including communicating, consumption for data analytics, up to the visualization [149], and acting back on actuators. It has to be remarked that machine to machine communications among IoT elements have to be secure, without the need of having them personally authenticated/authorized by the owner since in most cases the user does not know that data are processed by some process in the cloud or on the IoT Edge. In most of the scenarios, a high level of security has to be assured since the users rely on the system to manage private data, which can be eventually exploited to help-inform-assist the data owner in daily activities and tasks [207].

The IoT Platform architectures have to offer a high level of security. They need to support a heterogeneous source of information (IoT Devices, sensors/actuators, mobile, and streams) that is accessed in a multitude of different manners, and by using various communication systems in scenarios where in some cases, it is difficult to predict when and where the data are generated and made available for the system/subsystems of the platform [216]. For these reasons, the architecture of secure IoT platforms must be carefully designed, due to the huge amount of data exchanged among parties, the complexity and the heterogeneity of the protocols and devices involved [130], and the level of security assured to the users [146].

Moreover, a layer of complexity has been added by the final adoption of the European Union General Data Protection Regulation 2016/679 (GDPR) [12]. The regulation was proposed in April 2016 and became operative in May 2018. It has been designed to improve the former European Union Data Protection Directive 95/46/EC (introduced in October 1995), with the main goals to (a) provide uniform guidelines about the protection of individuals with regard to the processing of personal data and on the free movement of such data in all the EU (European Union) state members and (b) provide adequate recommendations with respect to the technology enhancement that occurred over the past 20 years. According to the GDPR, specific mechanisms to save and manage authentications and data messages have to be taken into account by design and by default. It resulted in a strong impact on the implementation of complete IoT end-to-end stacks, as explained in this chapter. Please note that GDPR aspects are more relevant in IoT applicative domains in which several users are involved, as in the Smart City, Smart House, Smart Health, etc., whereas they are less relevant in Industry

4.0 solutions in which all of the data have to be kept private and are typically owned by a single owner/industry and are rarely in need of differentiated accesses among users/groups at a fine-grained level [164]. This may happen in the cases in which the control based on Industry 4.0 IoT is performed on some cloud implementing an "Industry 4.0 as a Service" solution. Moreover, the compliance of an IoT-based solution with GDPR is a very complex aspect to be assessed [170]. There are a large number of online tools for assessing the compliance to GDPR that are applicable to institutions since the GDPR includes guidelines for managing privacy and not only guidelines and features for solutions. There is not a unique recipe or checklist to assess that a web application is GDPR-compliant and neither that an IoT Application is compliant, which is a much more complex architecture.

In this chapter, the privacy and security issues related to the design of the Snap4City IoT Framework are addressed (`https://www.snap4city.org`). Snap4City is a solution produced in response to a research challenge launched by the Select4Cities PCP (Pre-Commercial Procurement) H2020 research and development project of the European Commission (`https://www.select4cities.eu`). Select4Cities identified a large number of functional (mainly Smart City IoT) and nonfunctional requirements (e.g., open source, scalability, security, GDPR compliance, modularity, usability, and working on the cloud and on premise) which are reported on their web site, and aimed at creating the best solution for modern Smart Cities supporting IoT/IoE in the hands of public administrations and supporting creation of collaborative Living Labs (thus stressing the aspects of GDPR and security). Most of the identified requirements have been taken from the large association of Living Lab ENOLL (European Network of Living Lab association, `https://enoll.org`), and from concertation of smart cities at level of European Commission as EIP-SCC (European Innovation Partnership on Smart Cities and Communities, `https://eu-smartcities.eu`).

Snap4City responded to the research challenge and with more than 14 months of work, developed and set an operative solution, which has been demonstrated to satisfy all Select4Cities requirements, including those regarding GDPR and security discussed in this chapter. Snap4City, in brief, enables the creation and management of communities of users that collaboratively realize the following: (i) create IoT solutions and are somehow connected to organizations (cities, regions, industries, group of users, and final users as well), (ii) exploit open and private data with IoT and IoE de-

vices respecting GDPR, and (iii) create/use processes and IoT Applications that could be on the IoT edge, mobile and cloud and they may interact with each other and with users via messages, dashboards and applications of different kinds. The produced Snap4City solution is GDPR-compliant and provides end-2-end secure connections on the IoT stack. The set of highly challenging requirements constrained the Snap4City team to address the abovementioned technical and scientific unaddressed problems, most of them related to security and privacy. These specific requirements, the complexities and the solutions put in place are reported and discussed in this paper, with special care regarding the satisfaction of GDPR and security aspects in the full IoT stack for smart cities and in some measure also for Industry 4.0.

This chapter is organized as follows. In Section 5.2, the major requirements identified by Snap4City from its challenge on Select4Cities and which are needed for secure and Smart City IoT platforms are listed. They have been mainly divided into functional and nonfunctional requirements, and a specific section is presented to identify security vulnerabilities to be avoided. In Section 5.3, the related works are presented including an analysis of the most prominent solutions, focusing on aspects of security in the overall user data management workflow. Section 5.4 presents the Snap4City Platform General Architecture addressing the aspects of security when the IoT solution is deployed on premise, on cloud and mixed. In the same section, the aspects of security are analyzed in the configurations on cloud, while the aspects of authentication and authorization are discussed in Section 5.4.4. In Section 5.4.3, a table reporting a matching of requirements vs the main Snap4City components and section is reported. In Section 5.5, more detailed architectural aspects regarding securing the IoT M2M (Machine 2 Machine) communications are presented, taking into account IoT Devices, IoT Edge Devices, IoT Brokers, and IoT Applications. Section 5.6 discusses the remaining technical solutions to address additional GDPR aspects that are indirectly connected to IoT and that are mandatory to complete the security shield of the infrastructure. In Section 5.7, the details regarding the validation of the Snap4City solution are reported. The validation included the verification of the criteria for GDPR compliance vs requirements, and the results obtained performing verification of vulnerability in Penetration Tests and the actions performed to solve the detected problematic areas. In Section 5.8, the conclusions are drawn.

5.2 Requirements analysis

This section presents and discusses the major requirements that a platform for the Smart City IoT/IoE domain has to satisfy (in terms of privacy, security and GDPR). In some cases, the relationships with Industry 4.0 aspects are also presented as evidence to highlight the similarities. The overlap between Smart City and Industry 4.0 is relevant, since when a city has a water plant to monitor, energy consumption to control, etc., the problematic issues are those of Industry 4.0 plants. Since GDPR formalizes the aspects related to privacy, it is very difficult to separate GDPR-driven requirements from those about security. In the sequel, the requirements are presented in logical order from R1 to R18. In Section 5.6, the verification of GDPR compliance of the platform also addresses the related aspects of GDPR with respect to the following requirements. Therefore, the ideal Smart City IoT Platform has to establish the following:

- R1. support different kinds of IoT Brokers and thus different IoT Devices and IoT Edge devices. "Different" in the sense that the Platform must be capable of supporting a set of IoT Brokers (on the cloud or remotely located on premise). They may provide different protocols and modalities to authenticate and establish secure connections with the Platform and Devices;

- R2. support IoT Discovery Abstraction for IoT Devices. This means that the Platform must support the classification and search of IoT Devices, abstracting from their IoT Broker and protocol. Thus, in the activities of searching and subscription to IoT Devices, it has to be possible to identify them by searching for IoT Devices details such as: Device ID, sensors ID, geo-information (e.g., close to a GPS point, along a path, into an area, ...), sensor kind (e.g., temperature, humidity), nature (environment, mobility, energy, etc.), value unit (Celsius degree, micrograms per cube meter), protocol (NGSI, AMQP, COAP (Constrained Applications Protocol), etc.), Broker, etc. The result of the IoT Discovery process can be a set of IoT Devices or sensors that could be accessed / subscribed independently on their IoT Brokers and protocols;

- R3. guarantee authenticated connections among IoT Devices, IoT Edge Devices, IoT Applications, storage on the cloud (so-called Data

Shadow, as in AWS, IoT Azure), Dashboards on secure channels. The IoT Devices can be in the field and on premise, while others can be on the cloud and mixed. The authenticated connections have to be established among each other, and in the best cases, by using mutual authentications. Moreover, as a fallback solution, by using less secure connection models, such as those based on keys and/or basic authentication (that can also be employed when the IoT devices are not compliant with most secure protocol approaches). Please note that some of the communications among entities are machine to machine, M2M, while others are human to machine, H2M;

- R4. inform users about the security level at which the solution may work according to the level of security taken (it may depend on the kind of sensitive data managed);

- R5. support developers in managing security. This means that the developers of IoT Applications should be supported in creating applications that exploit the security in a transparent manner as much as possible. For example, the developers of IoT Applications need to create connections with: Dashboards (for presenting data and collecting actions from users), storage (for accessing to historical data, or for saving additional data, results of some data analytics) and with IoT Brokers (for subscribing on the data drive, or sending/receiving messages), etc. IoT Applications may also invoke and implement Data Analytics processes exploiting a large amount of data storage, for example, by using machine learning approaches. Thus, the authentications to establish these M2M connections have to be automated. This means that the developers are not forced to use the credentials in the source code to establish authenticated connections (for example, with the IoT Brokers, Dashboards, Storage, etc.). Please note that this kind of security weakness happens in most of the IoT platforms in the state of the art, see for example, Azure IoT;

- R6. guarantee Secure Communications in all kinds of connections involving IoT Devices, IoT Brokers, IoT Applications, Dashboards and storage. In some cases, the communications can be in PULL and/or PUSH. The secure communications have to be guaranteed by means of an authentication approach during which certificate and/or access tokens are used, and then activate SSL/TLS connections supported by

mutual authentications in the best cases as described in the previous points. The communication in M2M does not have an allowance for some back-office authentications that are specific for each user;

- R7. support developers with open hardware and open source software for implementing secure IoT Devices and IoT Edge Devices. In this context, most of the platforms use proprietary solutions/devices to guarantee the secure connections, see for example AWS IoT, Azure IoT suite, etc. This requirement is connected to R3 since the developers have to be enabled to create their own secure devices to communicate with the rest of the IoT platform elements in a secure manner, with mutual authentications. The adoption of an open solution allows the adopters to develop their own devices that can be secure at the same level of native solutions provided by the platform provider;

- R8. support signed consent to authorize the usage, access and management of the different Data Types of the Platform. The concept of Data Type is derived from GDPR and can be regarded as Data Category. In the context of IoT solutions, the Data Types can be: IoT Devices/Edges, IoT Applications, Dashboards, as well as Data entities in the storage, time series, etc. According to GDPR, the authorization/delegation to manage personal data (Types) provided by a user to the IoT Platform management must be performed by using a Signed Consent and not via an informed consent as in the past. Any Data Type should have a specific signed consent, registered on the platform and the grant can be revoked any time by the user;

- R9. register and manage IoT Data Types providing, receiving, managing, storing and retrieving Data Types, for personal data and related access control. According to GDPR, IoT Data Types must start as private for the user, and only after the creation, the user may decide to avail them as public or accessible to specific users by delegation. The delegation in access to Data Types has to be at level of: (i) IoT Data and data sets (group of IoT Devices); (ii) single IoT Device; (iii) single sensor/actuator value of an IoT Device, (iv) IoT Applications, (v) Dashboards, (vi) storage values, etc. For example, the owner of an IoT Device, collecting a number of personal health parameters (pressure, glucose, temperature, etc.), would be interested to grant access to a partner only about the glucose level (only for one of the sensors of

the IoT Device), and keeping the others private. Therefore, only the owners or delegated users can access the data. To make a Data Type public must be the equivalent of publishing data anonymously;

- R10. manage IoT Data Type ownership (permitting the change of ownership) and the access delegations according to the GDPR. In the delegation management, it must be possible to list them (check the grants provided) and to revoke delegation;
- R11. support roles, organization, and groups to manage different kinds of user's categories/group. It has to be possible to provide access delegation at Data Types to user categories to avoid creating thousands of delegations every time a new user joins a group. For example, in the smart city context, certain sensors would be directly accessible for all the officers of the mobility area. Thus, when a member of a group is added/moved, the delegations to its entities do not need to be created/removed. This requirement is mainly derived from Living Lab aspects and for multitenancy support on the platform to allow the usage of multiple applications and groups on the same platform;
- R12. store personal data in an encrypted way according to GDPR to prevent, in the event of breach, the identification of any specific individual compromised data;
- R13. provide at the users the "right to be forgotten", according to GDPR. A user must be able to fully manage personal data and eventually requiring the erasure of them from the Platform. As a limiting case, the user must be entitled to request the deletion of all personal data including user profile data. Please note that the platform must permit access to collected data at law enforcement agencies for a limited time interval, for example, for two months, that is a type of waiting/investigation time;
- R14. support auditing for each user, to monitor who has accessed their personal Data Types. The user has to access at the auditing data, obtaining details about the accesses, such as: when, where, how, and who accessed data; this feature is requested explicitly by the GDPR;
- R15. support data breach detection: to implement automated methods to detect in a short time whenever some data and Data Type have been tampered or leaked; this feature is requested by the GDPR;

- R16. support accounting in terms of collecting/computing metrics/indicators about resource consumption. For example, counting the number of IoT Devices/Data, IoT Applications, Dashboards, etc., that the user has registered, created, and requested. The assessment is the first step to enforce eventual limitations according to the user's role and/or for billing. In IoT platforms, it is very frequent that an accounting is performed on the basis of the number of messages exchanged (both H2M and M2M), the amount of data stored, the number of IoT Applications created, etc. The accounting feature is only marginally connected to the aspects of security and GDPR, while the insertion of limitations on the number of resources that each user may request/-exploit to/from the platform is a form of security against denial of services and penetrations that in most cases, try to abuse the services for resource releases;

- R17. support data protection (privacy and security) by design and by default, requiring controls built into products and services from the earliest stage;

- R18. ensure a level of security by providing technical and organizational measures appropriate to the risks, including, but not limited to, pseudo-anonymization, confidentiality and integrity, and performing a periodic penetration test. This requirement has to be satisfied by the Platform since many stakeholders may provide several kinds of data with different kind of licenses that range from highly sensitive data to fully open data.

In addition to the security aspects, it is important to keep in mind a few more nonfunctional requirements, which are valid for Smart City IoT Platforms and may also be for applications in the IoT smart home, Industry 4.0, etc. They are indirectly connected but relevant to privacy and security aspects. In fact, a modern IoT platform must:

- provide technical and organization measures to ensure scalability and cloud services for IoT Brokers, IoT Devices, and IoT Applications. This means that in most cases, they have to be managed as Virtual Machines or Containers in an architecture supporting vertical and horizontal scaling;

- provide technical and organization measures to ensure availability, resilience, disaster recovery, periodic stress testing, pentest and workload. This requirement has been identified in the context of Smart City IoT/IoE as a nonfunctional requirement related to the reliability and robustness of the platform to guarantee a high availability;
- provide support for Cloud-Fog data routing and local (on premise) IoT computation on the IoT Edge, on which the security must be guaranteed as well. This also means that the solution should be installable on the cloud and on premise, and mixed solutions may be viable as well;
- provide support for building Dashboards and data presentations, business intelligence, visual analytics with simple visual tools that can be used by nonprogrammers. Please note that the Dashboards are the front end of IoT Applications, and thus the connection to the rest of the IoT stack has to be performed by using authentications on secure connections, such as via HTTPS or Secure WS;
- provide support for registering and managing heterogeneous in/out sensors and actuators on IoT Devices and virtually on Dashboards such as: (i) virtual sensors as buttons, dimers, sliders, switches, etc. (which are elements in which the user acts creating data for the platform); (ii) virtual actuators for showing of real-time data such as: graphic representations of a bulb or of an engine, gauge, single content, speedometer, level bars, time trends, etc. (which are elements to show data on the user interface).

5.2.1 Specific Requirements on Security

As described in Section 5.7, the verification of most of the above described requirements has to be performed by using test cases, user interface inspection, and for some of them by using code inspection. In addition, in order to verify the aspects of security, the usage of the so-called Penetration tests is quite diffuse. In the context of the Smart City IoT, the main vulnerabilities to be verified can be summarized as:

- SQL Injection. Injections flaws occur when unvalidated data are sent as part of a command or query to the interpreter. The infected data are executed by the interpreter by running unexpected commands or accessing data without permission.

- Broken authentication. This refers to situations when procedures for authentication and session management are implemented incorrectly. Thus, the attackers may obtain passwords, keys, session tokens, or exploit implementing weaknesses to assume the identity of other users.
- Sensitive Data Exposure. When web apps and APIs do not protect sensitive data, attackers can obtain the data for fraud, identity theft or other crimes.
- XML External Entities (XXE). Old or badly configured XML processors evaluate references to external entities in XML documents. External entities can be used to identify internal files using URIs, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
- Broken Access Control. This refers to bad controls about what authenticated users can do. Attackers can take advantage of the deficiencies to access unauthorized features and/or data, such as accessing users' accounts, changing permission and roles, etc.
- Security Misconfiguration. Good security requires proper configuration of tools and web servers. In most cases, the default configurations are not always secure. This also implies keeping software and libraries up to date.
- Cross-Site Scripting. This type of flaw occurs when a web application receives data from untrusted sources and sends them to a browser without proper validation and/or "escaping". This kind of attack may allow running malicious scripts on targets' browsers; such scripts can hijack the user's session, deface the website or redirect the user to a malicious site.
- Insecure Deserialization. Unsafe deserialization may lead to code execution in a remote manner. In addition, these kinds of attacks can be used to perform privilege escalation attacks.
- Using Components with Known Vulnerabilities. Components, such as libraries, frameworks, and other software modules, work with the same privileges as the application. Once identifying the version, it could be possible to exploit their well-known vulnerability to perform the attacks.

- Insufficient Logging & Monitoring. Insufficient logging and monitoring, associated with a missing or ineffective integration with incident response, allow attackers to further attack systems, maintain persistence, rotate across multiple systems, and tamper, extract, or destroy data. Typically, the attacks are detected several days after the effective attack.

5.3 Related works

A Smart City IoT Platform should be usually accessible and used by a number of developers and operators and by final users that could have very low (or maybe nothing at all) knowledge and understanding of computer programming and communication protocols. The hiding of the IoT technical aspects and complexity, and at the same time supporting GDPR and security of personal/processed data is a very challenging task. There are some technical specific solutions frequently adopted by security practitioners that can drive the design of many basic aspects of security that people can use trustfully. On the other hand, there is still an open issue of how to create a full system that is satisfactory for all the mentioned security and privacy requirements in a user-friendly transparent manner for the different kinds of users.

In the literature and markets, several IoT platforms have already been proposed to the audience, even if just few of them have been tested on a large scale and on wide international geographical areas [147]. There are several interesting products that have been deployed for industrial scenarios and some of them that clearly still need vast improvements, mainly in data pipeline management and on the usability side. The platforms that support more flexibility usually became more difficult to use/configure, even by a user that is knowledgeable about IoT concepts and computer programming.

In the context of related work about IoT security, it is worthwhile to take into account the technological approaches that have been used to support the privacy and protection of data in distributed environments (IoT deployment). Quite comprehensive surveys about the different enabling technologies can be found in [191] and [40].

In Table 5.1, a comparison among Smart City IoT Platforms based on the identified requirements including GDPR is reported. The requirements listed in the table are a synthesis of those listed and derived from Select4Cities and

ENOL as described in the Introduction. Table 5.1 reports the assessment with respect to the Snap4City platform that is presented in this chapter. In the following paragraphs, we are presenting a review of the main technologies, research-oriented solutions and industrial platforms in the context of the Smart City IoT. For the industrial platforms, we have considered AWS IoT by Amazon [4], MS Azure IoT [24], and Google IoT [17], since we think that only these large platforms can be comparable in terms of security/privacy requirements coverage with respect to the Snap4City solution and architecture. The Snap4City general architecture has also been compared with other solutions in [49].

Among specific IoT security solutions for the ARM (Advanced RISC Machine) microcontroller, the ARM mbed IoT has the advantage [55] over the other platforms to provide an operating system for development, while the support for user privacy and devices' healthiness automatic check is overlooked. Calvin IoT platform by Ericsson [165] presents a five-layer architecture deployed on top of devices and has been substituted with the new Kappa solution. The HomeKit solution by Apple mainly focuses on features related to the smart-home scenario [18]. All of them, in addition to Calvin, offer authentication of smart devices via the X.509 certificate and an internal ad hoc solution for authorization. The TLS/SSL (Secure Sockets Layer) channel is always involved in overall communication. Minor solutions are the AirVantage stack which is an ecosystem of IoT projects provided by the Eclipse consortium [21], and SmartThings from Samsung [33] that focuses on the home-connected scenario and permits authentication via OAuth2.0 protocols over SSL/TLS channels and authorization via an ad hoc capability-model.

Many techniques for secure communications have received an increment of interest in the last period due to the direct application in the IoT context [133]. Some of them are directly employed in the definition on privacy models for reciprocal trust. For example, Data Tagging permits associating labels to the dataflow in order to grant trusted entities access [99] mainly in a social-bookmarking context [120] or in platforms for photo sharing [197]. Zero Knowledge Proof allows managing user's privacy between two entities without the need of the prover to reveal any information to the verifier [113], [41]. Elliptic curve-based zero knowledge proofs has been used in devices with strong constrained resources to establish trust between two parties [74]. Group-key management technologies enable cipher support in a

group communication, and thus also in distributed networks [178] with novel approaches for the IoT scenario [203]. Finally, the K-anonymity privacy model permits disclosing entity-specific information such that the released data cannot be reliably linked back [184], [177].

Other solutions for IoT security have been based on the Blockchain paradigm. Some steps toward a complete IoT platform based on Blockchain concepts have been made and look promising. For example, the usage of Blockchain for the management of authentication and authorization via Smart Contracts [80] is such an application. It relies on an external (central) repository or a key-management system and not on a distributed and decentralized approach [93]. In addition, some problems related to the scalability have also been detected, regarding the management of policies for authorizations and the risks of race attacks [131]. In this chapter, we are not addressing IoT Blockchain solutions since, for the moment, they do not cover the entire security stack and do not satisfy GDPR.

Moreover, there are a number of proposals that aim at increasing the control of privacy in the IoT context. These solutions can contribute to end-to-end communications of the platforms. The most relevant are the obfuscation systems that try to provide the anonymization of location-based services through spatial and temporal cloaking [115], [96] or over noisy channels [88]. Other specific methods about privacy and protection have been applied in medical data field [127], using pseudo-anonymization techniques that focus on decupling data from the owner to enable a secondary use of data with respect to data privacy [155] or to enable trusted third parties via privacy-enhancing techniques (PETs) in the context of data collection [81]. Some other researchers focused their attention on the specific problem of enabling trust between different components in a distributed IoT architecture [75], [182].

Among the most interesting widespread solutions, we can surely place Amazon AWS IoT (Amazon Web Services) [4]. This platform lets smart devices connect and interact each other and with the AWS Cloud. AWS IoT makes wide use of proprietary services such as Amazon Dynamo DB (database), Amazon S3 (simple storage service), Amazon Machine Learning and others. It consists of a cloud solution that provides a data shadowing system to support intermitting connectivity via the communication protocols MQTT, HTTP1.1 (Hypertext Transfer Protocol) and WebSocket. A Registry Unit is used to collect any information about the IoT Devices and

to track their behavior. The main supported security features are:

- authentication at the device level via X.509 certificate and via AWS authentication;
- authorization and access control by mapping the X.509 certificate to a policies-based system through a set of rules processed by a so-called Rules Engine;
- secure communication of traffic is ensured to be made over SSL/TLS protocol to ensure the confidentiality of the application protocols, only for their supported protocol and devices.

The AWS IoT ecosystem is composed of a complex stack of Amazon's services that enable the developers to define the business logic via programming the data flow among IoT Devices' toward their visualization in proprietary dashboards. The variety of tools for IoT programming is wide and relies on several AWS components (AWS Batch jobs, AWS Step functions, Amazon MQ (Message Queue, with an IoT Broker as Active MQ) and many more). Moreover, the IoT Devices can trigger AWS Lambda Functions that may be written in Java, Python and C#. The Lambda Functions are used to implement business logic or to trigger actions in the AWS Cloud or on premises with some limitations. The data visualization is rich and flexible, and it can be exploited for data inspection and reporting. The AWS IoT Device Defender module can be configured to create alerts on a data breach on a singular device. Accounting on the activity of the platform is rich and detailed: it makes strong use of the IAM users (Identify and Access Management) module to specify groups and roles and to cluster/aggregate data and information. Limitations of this platform are discussed in the sequel.

Another well-known platform is the Microsoft Azure IoT Suite [24]. It enables the usage of smart devices to the Azure cloud via the modules called Cloud Gateway and IoT Hub. It manages the identity and authentication of registered devices with an identity registry and natively supports communication protocols such as AMQP (Advanced Message Queuing Protocol), MQTT and HTTP. On top of the communication layer, there is the IoT Solution Backend layer that presents a set of Azure services (as machine learning and analytics) and the Presentation Layer that is involved in the visual presentation of the data. The main security features of the Azure solution are:

- authentication is mutual using the SAS Token (Shared Access Signature) and X.509 certificate;
- authorization and access control are realized via Azure Active Directory using a policy-based model to support near-instant revocation;
- secure communication using SSL/TLS protocol; the identity registry implements the secure storage manager.

Azure IoT provides a development environment to build business logic. The integration toward mobile applications is made by using the Xamarin platform in C# and .NET programming languages. In addition to the mobile environment, it is possible to also write applications in Java, Node.js and Python. Security is provided by the IoT Hub. For the creation of complex dashboards to visualize the data, the developers can use the Microsoft Business intelligence tool and a detailed role-based user access control is available to provide a rich description of the user characteristics and rights (Owners, Contribution, Reader, User Access Administrator).

Finally, the Google Cloud IoT is a set of tools to connect and process smart devices' data in the cloud and at the edge [17]. It is composed of cloud services that permit native support of MQTT, HTTP and AMQP protocols over TLS/SSL via a two-fold subsystem called Brillo and Weave. The offered security features are:

- authentication via Oauth2.0 (Open Authentication) protocol along with a digital certificate;
- authorization and access control involve the SELinux module to manage the access control security policies. The enforcement is made above the right to read, execute and write for any users or group of users;
- secure communication provided through SSL/TLS protocol.

Google IoT platform makes strong use of the access token (to enable authenticated access to the data) in the form of JWT (JSON (JavaScript Object Notation) Web Tokens), to enable access between the components. On the other hand, it is not direct for the IoT developers to inject credentials in the Application Logic to exploit them. Tokens are created when a new IoT Device is registered, and credentials may need to be specified in clear text in the application code. The documentation for the platform is not of

immediate usage for experts. A detailed set of APIs (Application Program Interfaces) is provided, and only simple cases are presented that require knowledge of Apache Beam. Several programming languages can be used to program the data flows from the device to dashboards, such as Java, Node.js, Python, Go, Ruby, PHP, C#.

Amazon AWS IoT and Microsoft Azure IoT Suite solutions strongly rely on proprietary and closed sources for the user identification and access control enforcement (AWS IAM, Azure Active Directory, respectively). In meantime, the Google IoT platform supports integration with the open OpenID Connect protocol. All these platforms have strong limitations on the supported number of protocols (mainly just HTTP and MQTT), and on the number and IoT Brokers types that are supported (very limited support on managing multiple brokers). For the security connection, the modalities in which the transport level security is guaranteed are very similar: TLS/SSL and X.509 certificate. In most cases, from their proprietary certified IoT Devices and toward their IoT Brokers only. In most cases, when the user passes to develop IoT Applications and attempts to exploit communication with custom devices, the security is not natively guaranteed. Moreover, from the documentation, it is not evident how to implement custom scenarios and secure devices. In these cases, the definition and generation of security credentials (access tokens) is not supported at runtime, and it is used only when the IoT Device is registered to the platforms. It is difficult to find specific ways to enforce the access control in real time. Specific tools to label and classify the type of data that flow in the platform by default were not found in the analyzed solutions and none of them permits specifying the way that data authentication is provided. In those platforms, a common approach is taken for programming the data flow: programming languages can be used to implement a public interface so that the platform can use the written source code to manage the data. None of them provide a visual editor for creating IoT Application flows that can be used without any software programming skills. In addition, these solutions often require mandatory cloud services and are not suitable for deploying on an autonomous on-premise scenario. As a limiting case, they may accept having some processes on the IoT Edge on premise, with reduced capabilities connected with the cloud solution.

It is important to notice that, in the context of IoT, different solutions may need to satisfy different use case scenarios. For example, in the healthcare domain the personal data must be protected in a stronger way and

real-time requirements must be strictly satisfied with respect to what could be needed in the entertainment domain. A complete study of the classification of different scenarios and their required security requirements has been made in [174]. For example, among the considerations are clarifying differences among aspects such as object security, end-to-end security, cyber-physical-social security, hierarchical security, lightweight security and defense. For this reason, in the next section, we describe the specific requirements on privacy and security that have been identified in the context of the Select4Cities Smart City IoT/IoE challenge, with some additions that have been applied to address the aspects of the Industry 4.0 domain that are needed in city plants/installations with the Snap4City architecture design.

Other solutions such as Kaa [23], CISCO [7], CarrIoTs [6], Thingsboard [35] support IoT networking, without visual programming, semantic management, and limited control and integration with smart city data. A relevant lack in the IoT management of devices for IoT Discovery is registered to the platform as IoT Eclipse.org [21], IoT Ignite [22]. Most of them lack semantic search support, such as Bosh IoT [5] and PCT ThingWorkx platforms [36]. For these last two platforms, the effective capabilities in support of GDPR are not very clear, especially for supporting multiple Data Types as those needed in Smart Cities, which are more suitable for industry applications.

5.4 Snap4City architecture

The Snap4City architecture has been designed and implemented to satisfy the above described requirements that have been introduced in principles described in this section. Thus, Snap4City allows being deployed on the cloud and on premise, as a well as in mixed scenarios, and supports IoT Edge Devices with IoT Applications, and IoT Applications on the cloud as well. The Snap4City architecture is flexible enough to satisfy the above requirements which are valid for a wide range of IoT domains with local computation on premise, on cloud and mixed. In Snap4City, a set of tools can guarantee the privacy and protection of the data managed by the system respecting the GDPR, requirements of ENOLL and those of Select4Cities. Snap4City was the winning solution of the Select4Cities PCP and challenge.

The Snap4City.org is the cloud-based backbone of the solution. It acts as the boilerplate where different city organizations and City Operators configure, via a set of graphical tools, the processes for data aggregation, data

Table 5.1: Comparison of the mentioned IOT Platform on the basis of the main requirements identified. The table is divided in two part: Open source and proprietary solutions. "( )" are used to mark partial, M means MQTT, C means COAP, H means HTTP, L means LIMITED. The assessment refers features accessible at the date of submission

	IOT Discovery Abstraction	Authentication, Authorization	Security end-2-end, secure on IOT and Dashboards	Open HW and Open SW	Integrated Community management	Data Types: IOT Devices, IOT App, Dashboard, Data	Data Type: Publish/share, Delegation, Consent and change	Data Type: Download and Delete	Auditing on Data Type Access	Open Source end-to-end	Scalability IOT	Visual Programming end-to-end applications	Advanced Smart City API, MicroServices	Multi Domain Semantic Platform	Standard based Modules and IOT, Open Devices	Resource Sharing	Data Analytics integrated	Dashboard H24/7, protected connection	Multi-protocol on IOT
Snap4City	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y
KAA [23]	y	y	y	y	y	y	n	y	y	y	y	n	y	n	(y)	n	n	y	y
Thingsboard [35]	y	y	y	y	n	y	n	y	y	y	y	n	n	n	n	n	n	y	M, C, H
IOT eclipse.org [21]	n	n	n	(y)	n	y	n	n	n	y	y	n	n	n	y	n	n	n	y
IOT IGnITE [22]	n	y	n	y	n	y	n	y	y	y	y	y	n	n	n	n	n	y	M
FIWARE [14]	n	y	n	y	n	n	n	y	n	y	(y)	(n)	y	n	y	n	n	y	y
ARM mbed IoT [55]	y	y	y	y	y	n	(n)	n	y	y	y	n	n	n	y	n	n	y	L
Airvantage [1]	y	y	y	y	n	y	n	y	y	y	y	n	n	n	n	n	n	y	M, H
AWS [4]	y	y	y	y	n	y	(n)	y	y	n	y	n	n	n	y	y	(y)	y	L
Azure IOT [24]	y	y	y	y	y	y	y	y	y	n	y	n	n	n	y	y	(y)	y	L
PTC Thing-Workx [36]	n	y	y	y	y	y	n	n	y	n	y	y	n	n	y	n	n	y	y
Bosch IoT Suite [5]	y	y	y	y	y	(y)	(n)	y	y	n	y	y	y	n	y	n	y	y	y
CISCO Jasper [7]	y	y	y	y	n	(y)	(n)	n	y	n	y	n	n	n	n	–	(y)	y	L
Siemens Mind-Sphere [32]	y	y	y	(y)	n	y	(n)	y	y	n	y	y	n	n	y	n	y	y	y
Carriots [6]	y	y	y	(y)	n	y	n	n	y	n	y	n	n	n	–	n	n	y	M
Google IOT [17]	y	y	y	y	y	y	n	y	y	n	y	n	n	n	n	n	(y)	(y)	M, H
Homekit Apple [18]	y	y	y	y	n	y	n	n	y	n	(y)	n	n	n	n	y	n	y	L
Smarthing Samsung [33]	y	y	y	y	y	y	(y)	y	y	n	(y)	n	n	n	n	n	n	y	L

analytics, and data rendering via dashboards. The data generated by IoT Applications can be injected back in the storage and in the Knowledge Base by making them accessible for MicroServices and Dashboards, and thus in connection with the IoT installations in the field. Therefore, in the following paragraphs, the different scenarios are presented, keeping in mind that all of them persist at the same time in the actual deployments.

5.4.1 General architecture

The Snap4City architecture highlighting the security aspects is reported in Figure 5.2. From the upper left part, the system permits accessing the platform at the users with a set of devices/terminals via a Web User Interface and/or mobile applications. The users may access the platform functionalities in an authenticated manner. The user's identity is verified and propagated by the Single Sign-On (SSO) Login module toward any Snap4City module that needs to access some user's data. The SSO allows performing a centralized authentication management to all the modules and tools of the infrastructure. The users' personal data (mandatory and optional information about the user) are kept in a separate storage called User Registry.

The platform is interfaced to the real world, in which a multitude of IoT Devices are located and supported (hosting several sensors and actuators). A simple IoT Device can be a microcontroller enriched with sensors/actuators capable of send/receive messages to/from a specific gateway (Lora, SigFox, OneM2M, etc.) or directly with some IoT Brokers that can be external or internal to the Snap4City platform.

Since most of the IoT Brokers do not support native authentication and channel protection (such as IoT Orion Broker of Fi-Ware), they can be protected by an IoT Firewall (a shield) that allows performing only authenticated connections on the basis of the SSO and thus allows or denies access according to the requests. The IoT Firewall performs the access control establishing a mutual authentication and a secure connection over HTTPS, exploiting OMA NGSI Ver.1 and Ver.2 protocols (Open Mobile Appliance, Next Generation Service Interfaces). The IoT Firewall supports other authentication mechanisms such as those based on: (i) access token, (ii) a couple of keys (as in SigFox), and (iii) basic authentication over HTTPS. It can be easily adopted for wrapping other protocols and IoT Brokers.

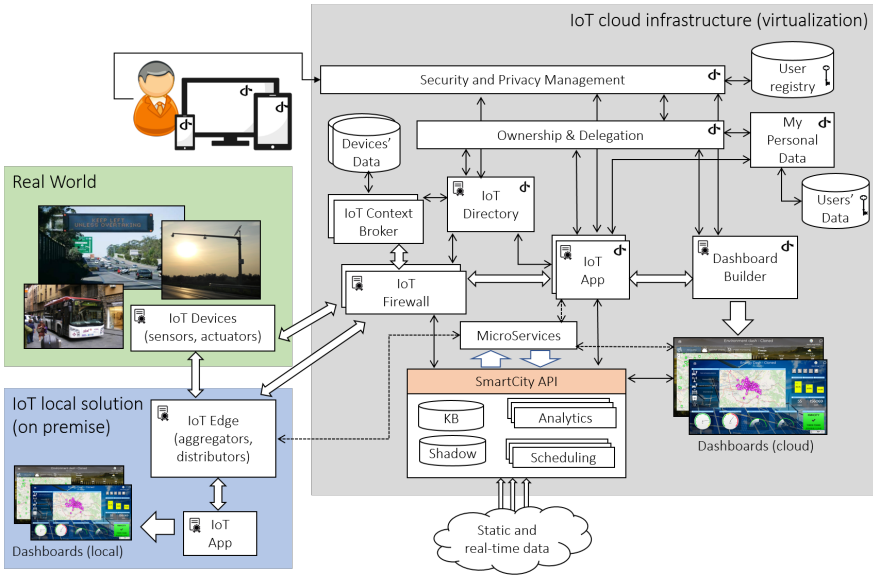


Figure 5.2: Snap4City architecture, main security aspects

5.4.2 The role of IoT Edge on premise vs security

IoT Edge Devices are a powerful class of IoT Devices that is capable of (i) managing sensors/actuators IoT Devices, send/receive messages to/from some IoT Gateway, and (ii) executing IoT Applications (processes with some local logic, for example, for data aggregation, data mining, machine learning, artificial intelligence, AI, etc.). In Snap4City, IoT Edge Devices can be based on Raspberry pi, Linux, Windows, Docker and/or Android. Thus, powerful computer-based devices can be delegated to play the role of IoT Edge and not only embedded systems. IoT Edge Devices may have capabilities to connect on multiple communication channels including local wireless networks. These kinds of devices may play the role of a message router toward some IoT Brokers (for example from ModBUS to Snap4City). More complex handheld personal computers (e.g., Android on smartphones or similar devices) can also be used with the Snap4City platform. They are IoT Edge Devices with the embedded strong network capabilities toward the Internet via 4G, 5G connections to directly communicate with the infrastructure (for example, to

access the historical data for machine learning). In all these configurations, the solution has to guarantee secure connections and authentications.

As a remark, any supported IoT Device can communicate directly to the platform via an Internet communication reaching some IoT Broker on the cloud. Otherwise, they can be configured to be connected with Snap4City using an aggregator/distributor device (IoT Edge) connected, for example, via a local wireless network interface toward an access point (in IEEE 802.x), or other private/patented technologies (LoRa, SigFox, RS485, ModBUS, BLE (Bluetooth low energy)), thus playing the role of an IoT Gateway. The IoT Edge in turn would be capable to communicate with the platform on premise or on the cloud via the Internet.

Moreover, Snap4City IoT Edge Devices may have IoT Applications exploiting Snap4City services on the cloud. IoT Applications in Snap4City are node.js processes that can be formalized by using the visual syntax and model of Node-RED [27]. The exploitation of Snap4City services is performed by installing the Snap4City suites of MicroServices (see Figure 5.3), directly from the Node-RED Library`https://flows.nodered.org/?term=snap4city` of the JS Foundation. The libraries allow easily invoking MicroServices on the cloud using them as nodes and they establish secure connections. They include services such as: IoT Brokers, IoT Discovery, Big Data storage, Data Shadow, Data Analytic (artificial intelligence and machine learning), External Services and Dashboards, plus a large number of specific MicroServices for Smart City services, and Industry 4.0. Examples of Smart City MicroServices are: search for paths of buses and time schedules, parking status, routing, parking predictions, anomaly detection, traffic flow reconstruction, environmental data values at any point of the city, etc. [52]. The Snap4City MicroServices exploit the Advanced Smart City APIs [152], which in turn are largely based on the semantic engine of the Km4City Knowledge Base (KB) for smart city data and service management.

IoT Edge Devices may need to provide results via local Dashboards (accessible on an intranet, on premise, for example, by using Node-RED classic Dashboards). More sophisticated Dashboards can be produced exploiting Snap4City cloud data and capabilities, via secure communications, such on WS (Web Socket), to guarantee the real-time stream with virtual sensors/actuators on Dashboards. They are also called Virtual IoT Devices. Thus, a set of predefined sensors and actuators, such as buttons, dimers/knobs/sliders, lights, pumps, fans, traffic light, etc., and custom widgets, can be created.

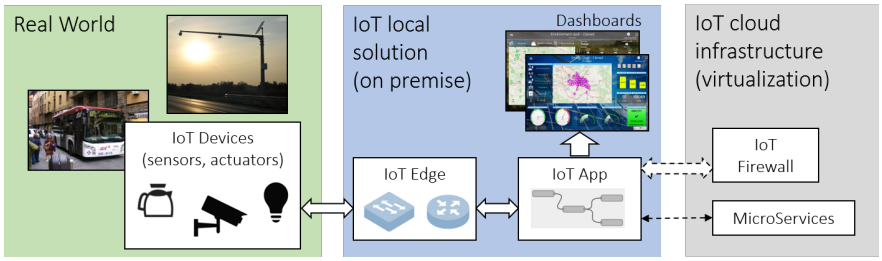


Figure 5.3: Snap4City configuration for local solution (with not mandatory use of IoT cloud infrastructure)

5.4.3 IoT Cloud scenario vs security, general view

When an IoT solution on the cloud is chosen (as shown in Figure 5.4), the data flows from/to the IoT infrastructure via a set of Context Brokers (Mosquito MQTT, Fi-Ware Orion Broker NGSI, etc.) without the usage of IoT Edges. The IoT data streams arriving on the cloud are automatically saved on Data Shadow for creating the historical IoT data.

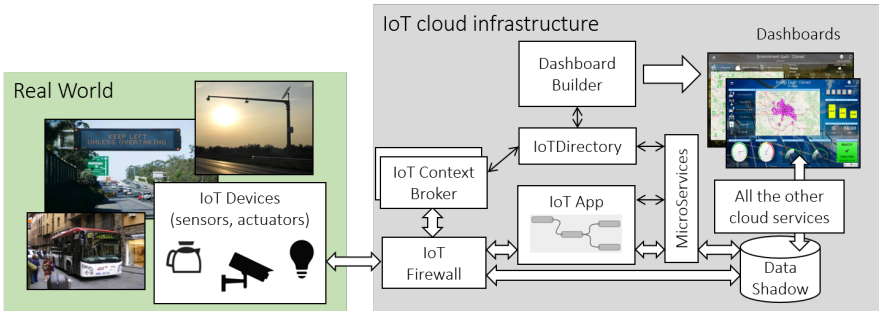


Figure 5.4: Snap4City configuration for a complete cloud solution

The data collected in the Data Shadow is made available to any process of the Snap4City solution, mainly to the IoT Applications deployed on cloud via the MicroServices. The IoT Applications (based on Node-RED plus the Snap4City library of nodes/MicroServices for the Smart City) allow the user to design, in a visual manner, personal data flow IoT Applications for data

transformation, management, and data processing. The MicroServices are the means to invoke Data Analytic processes on the cloud to access storage, and domain specific services such as those described in the previous section. The IoT Applications can integrate visual elements of Dashboards (called widgets), which can be improved and integrated with smart city data by using the Dashboard Builder [61]. The Dashboard MicroServices (Snap4City nodes of Node-RED flows) allow users to compose in visual and intuitive manners the data presentations as Dashboard Widgets for full user interaction and include virtual sensors and actuators, custom widgets, exploiting smart city API, maps, etc. The Security of the Smart City API is taken for granted in this chapter since it is the Rest Call as described and discussed in [152].

Since the IoT Platform needs to connect multiple IoT Brokers and Devices, an IoT Directory listing them and providing general abstraction services is needed. From the security point of view, the IoT Directory must manage the IoT Device's registry, including references to the credentials that IoT Devices have to provide at the IoT Firewall to establish a secure connection on the IoT Platform. In this regard, different levels/models of authentication are available (key credentials, certificate credentials) and are specified whenever a new device is registered by the user in the Snap4City ecosystem. The IoT Device messages can also flow directly to the IoT Applications, where, in the form of aggregated data, they can be part of data analytics and data refinement. This means that the IoT Application has to be capable of subscribing to the IoT Device, via the IoT Broker to receive data in Push.

Any IoT entity, such as data, IoT Devices (with their sensors and actuators), IoT Applications and Dashboards, whenever they are instantiated/created (on the cloud or on premise), have to be associated at their creation with a user (the owner).

The IoT entity owner is the only one authorized to manipulate those entities, and thus entitled to provide delegation in access to those elements. The rights of the entities (both ownership and delegation) are managed by the Ownership module which can be queried only in an authenticated manner. Dedicated tools for the entity and Data Type are used to grant/deny access to a user/group/ organization. For example, the IoT Directory is the only tool for changing the ownership and creating delegation on IoT Devices. The users can also make public their own IoT entities, which are implemented

by creating a "Delegation to Anyone" in an anonymous form according to GDPR.

The ownership and delegation data, in addition to any other personal data, are put in a private and secure storage managed by the MyPersonalData module, in conformance to the GDPR. MyPersonalData are labeled using a classification based on motivation, variable name, variable value, and variable unit. Once defined, the developers can exploit the MyPersonalData safe storage via MicroServices or APIs and may also render these data on Dashboard Widgets. In addition to the personal data, the so-called MyKPI (Key Performance Indicator manager) is also available. They can be used to manage time series with variable GPS locations. They are typically used for storing mobile sensors, values collected from mobile phone movements (personal paths), on board unit of vehicle data, etc.

5.4.4 Security of User/Machine access and auditing

Please note that in the IoT stack (as depicted in Figure 5.1), a number of entities associated with the users may need to preserve privacy and security, such as: IoT Devices, IoT Applications, IoT Edges, Dashboards, and data storage. Therefore, before entering into the IoT aspects of security, it is mandatory to describe the mechanism for user/machine authentication and authorization.

The Snap4City solution uses several mechanisms to provide the authentication and authorization enforcement for users/entities to access the platform resources (see Figure 5.5). The User Registry is partially managed by a distributed directory information service (based on LDAP, Lightweight Directory Access Protocol) and partially maintained by a CRM (customer-relationship management) based on Drupal (which are maintained synchronized). The LDAP module is reachable only via a private internal subnetwork such that it is more easily protected from attacks. On the CRM, the users can review and update their information and eventually request to completely remove their account data according to the GDPR. Mandatory information is saved in LDAP and includes: usernames, hashed version of the user's password, email, roles and organizations/groups affiliations. The username is the primary key used to identify the users in the overall Snap4City platform. The roles (Manager, AreaManager, ToolAdmin and RootAdmin) are used for users' classification regarding their trust level in the framework and to grant access to Snap4City functionalities (e.g., enable different views

of the interface, permit more or less data investigation). Moreover, the public/anonymous users are also authorized to play with some tools, even if with limited features. The organization/group information is used as a multitenancy key to organize the users in terms of location or purpose and affiliation (for organization, e.g., Firenze, Helsinki, Antwerp, DISIT). For each organization, a number of groups can be set up and each organization may define its own names for them (for example: Developers, ICT officials, third-party developers, citizens, decision makers).

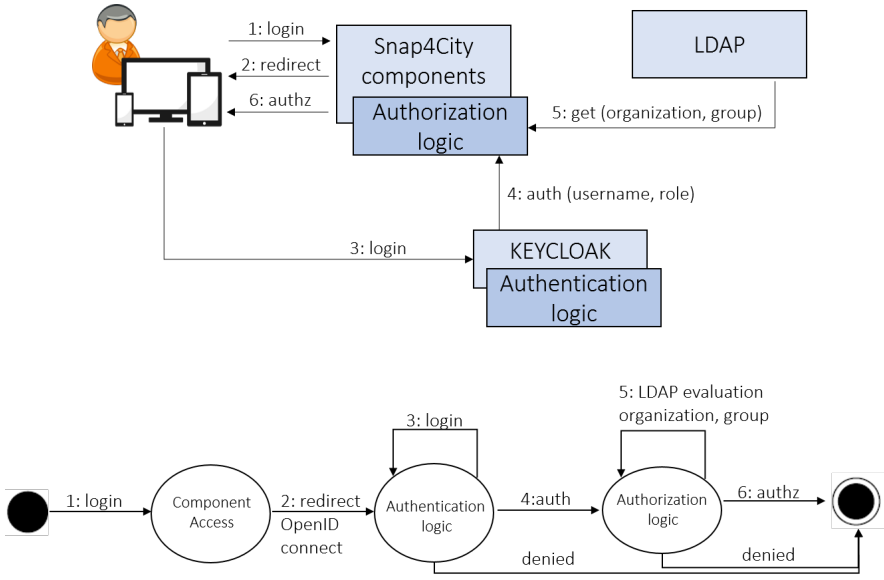


Figure 5.5: Snap4City communication and state-transition diagram about the management of users' authentication/ authorization

The user authentication is enforced by using a Single Sign-On (SSO) module with Identity Management. It exploits the OpenIDConnect protocol to provide a users' authentication system for all Snap4City modules (which include the CRM module based on Drupal, and Keycloak for the OpenID-Connect). Since OpenIDConnect is based on OAuth 2.0, the latter is used for the authentication enriched with the user's identification part. According to the flow numbers reported in Figure 5.5, every time a user tries to access a resource exposed by a Snap4City module (1), the request is forwarded to

the SSO Server to verify if the user is already logged (2). In the case in which the user is not already logged, the SSO Server requires the user to specify user's credentials (3). Whenever there is a match to the credentials stored in the LDAP users' register, the user is authenticated (4). Then, it has to be eventually authorized (6) to access the resource exposed by the Snap4City module (depending on ad hoc additional enforcement provided by the modules (5)). The user's role specified in LDAP is mapped in the SSO rules' registry. Thus, the system administrator can specify, in particular, the access rule (5) to enable or deny access (specification of grants) to a specific user's role to specific Snap4City modules. During the authentication, Keycloak passes to the contacted Snap4City modules the usernames and their roles. Whenever the user is granted to access a specific module, the module can implement another finer ad hoc authorization's rule to eventually authorize/deny the requested resource (6). The module could also contact the LDAP to retrieve the organizations and groups the user belongs to.

Any communication between a couple of Snap4City modules is made on top of the SSL/TLS protocol. Thus, transmission is kept confidential and a system of temporary shared secrets, represented in the form of an access token, JWT, that permits the SSO system to work among the different modules is used. As described above, when the user accesses in an interactive manner, if a JWT is not present or not valid (elapsed, tampered or not correctly digitally signed), it is redirected to the login Web page (3). When a M2M connection needs to be established; for example, an entity (such as an IoT Edge Device or an IoT Application) needs to access to the Snap4City MicroService module, a formal user still needs to specify credentials, at least for the first time. Then, an offline-access token is released (refresh token), and the machine does not need to request the user to login anymore and can use the refresh token to request a normal access token as specified above. The refresh offline token has a long lifetime, so that human intervention is limited, and it is maintained in a safe. Once the authentication and identification of the user is successfully completed (a valid access token is returned), the Snap4City modules are contacted attaching the JWT as their credentials. They are used to verify if the user/machine has enough rights to access the requested data/resource.

The OpenIDConnect protocol enables the identification of entry points and of communications among different Snap4City modules to authenticate the users/machines. This configuration permits realizing a good user expe-

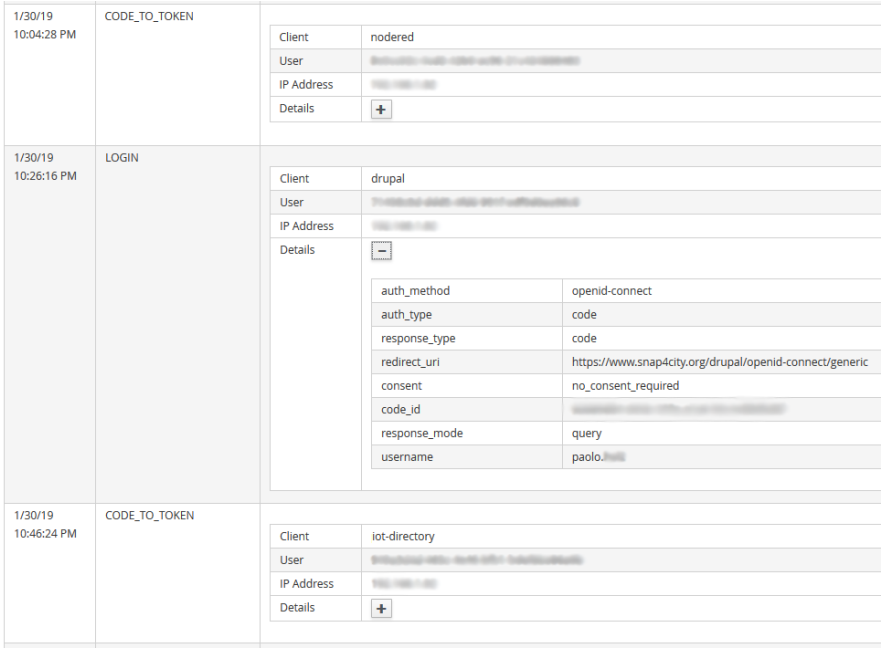


Figure 5.6: Auditing of module's activities (some text has been intentionally obfuscated)

rience; and at the same time, setting up mechanisms for detailed auditing of user accesses (see Figure 5.6), which is a demanded requirement for infrastructure monitoring. Moreover, the secrets that are used by the different Snap4City modules to access the SSO Server for user's authentication are specific for any module; thus, in the case of leaking or malicious intrusions, the problem can be isolated, and there is no need of a complete system reconfiguration.

5.4.5 Architectural considerations

As a summary, in Table 5.2, the main requirements and their mapping on architectural components are reported. The mapping also provides evidence regarding which are the main sections and subsections in which those aspects are addressed in this chapter. Please note that a systematic visualization

of all the aspects would take too much space; therefore, this chapter has been optimized to show only the most relevant and innovative aspects and presents them by scenarios. Moreover, the Snap4City platform is compliant with several protocols, documented and has several test cases `https://www.snap4city.org/283`. In addition, all the source code is on GitHub and is 100% open source including the security aspects from code of IoT devices to that of Dashboards.

5.5 IoT M2M secure connections

In this section, the architectural mechanisms for establishing secure M2M communications are discussed and presented. In detail, the security aspects addressed are those related to: IoT Device communications on premise and on the cloud; IoT Applications communications with respect to devices on premise and on the cloud; and communication of IoT Devices and IoT Applications with respect to Dashboards.

5.5.1 IoT Network, Devices vs security

Any IoT Device has to be registered in the IoT Directory to enable the communication from/to an IoT Device with the infrastructure (stating also the security model adopted, if any, and obtaining an IoT Broker). For IoT Device registration, the user has to specify a unique identifier, type/manufacturer, and location on a map. The IoT Broker can be internal, and thus it can be chosen/assigned on the basis of the protocol. As an alternative, the user may refer to an external IoT Broker, which can be registered on the platform as well. When the number of devices to be registered is massive, the user can use a procedure to automatically register a large set of devices starting from a detailed CSV (comma separated value format) file containing all the needed information (i.e., so-called Bulk Registration). According to the requirements, several different authentication schemas for different kinds of IoT Devices and protection levels have to be supported and have been implemented as described in the following. In addition, all the IoT Devices at their registration start as private in terms of the user that registered them, that is: private as default according to GDPR.

In many proprietary solutions, the interaction with the IoT Devices and thus also their authentication is completely demanded to be proprietary to

Table 5.2: Requirements vs main sections of the paper and main modules of the architecture

Requirement description	Security and Privacy Management	Ownership & Delegation	Personal Data	IoT Context brokers	IoT Directory	IoT Firewall	IoT Apps and management	Dashboard Builder and management	MicroServices	Smart City API	IoT Devices	IoT Edge, IoT App on premise	Data Shadow and Storage	Knowledge Base Km4City	Snap4City Platform Support Living Lab
R1. IOT Brokers				x	x	x									
R2. IOT Discovery Abstraction					x					x				x	
R3. Authentication, Authorization	x	x	x							x					x
R4. Inform User of Security Level															x
R5. Developing Secure Applications	x	x				x	x		x						x
R6. Secure Communications	x	x													
R7. Open HW and Open SW									x		x	x		x	x
R8. Signed Content Vs Data Types	x	x													x
R9. Managing IoT Data Types	x	x	x	x	x	x	x	x	x	x	x	x	x		x
R10. Managing Ownership and Delegation	x	x	x												x
R11. Support Roles, Org. and Groups		x	x		x	x								x	x
R12. Encrypted Personal Data	x		x												x
R13. User Profile Management, forgotten			x												x
R14. Auditing Data Types	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
R15. Data Breach Detection	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
R16. Accounting	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
R17. Data Protection by Design	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
R18. Evidence of Level of Security	x	x													x

the gateway/server infrastructure and provide specific credentials. In those cases, the IoT Devices exchange messages/data only with their own infras-

tructure servers (e.g., for SigFox configuration, the SigFox server provides two keys: K1, K2, as credentials). As a consequence, in Snap4City, as in other interoperable IoT Platforms, it is possible to obtain data access in pull or receive them in push (after subscription). Once a data message arrives in the Snap4City platform, it is associated with the owner of the IoT Device disregarding if the message has been pulled or received in push. In the case in which an IoT Application is developed for data collection, the application itself (for example, a coded program) is forced to include in the code body the credentials and this could be regarded as a breach of security since one accessing the IoT Application code would also access the credentials. In Snap4City, in order to avoid this weakness, the MyPersonalData safe storage is used for automatically storing of the IoT Device credentials which are used for subscription (push) or for each server access in pull. Thus, IoT Device credentials are used only at the execution time on the basis of the passed refresh token and are not present in the code. The whole mechanism is automated if the user registers on the platform some SigFox or other proprietary authorization schemas.

For Snap4City Open IoT Devices (devices that are provided as Open Hardware/Software, or derived solutions), the platform can rely on several communication protocols, such as NGSI, MQTT, COAP and AMQ, and provides a Snap4City protection system. If NGSI is chosen, the Orion Context Broker of Fi-Ware is used with the IoT Firewall. Examples of Open IoT Devices of Snap4City are: IoT Button ESP32, Arduino based IoT Device, Raspberry pi IoT Edge (open SW), etc., of which the source code can be downloaded from the Snap4City Portal.

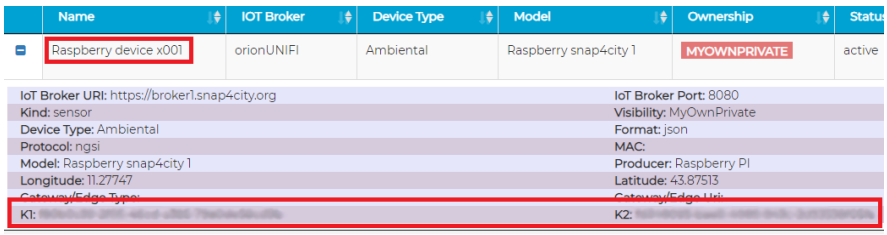


Figure 5.7: Details of a registered IoT Devices

A simple authentication modality makes use of a pair of keys (K1, K2) as credentials. In this case, they are automatically generated by the Snap4City

when the IoT Device is registered in the IoT Directory and are provided to the device owner for setting them into the IoT Device (see Figure 5.7). When an IoT Device needs to communicate in push with the Snap4City framework (e.g., a value of a sensor is updated), it has to establish the communication using K1, K2 credentials and its own IoT Device identifier.

Moreover, in Snap4City, according to GDPR, an additional set of keys is generated when the IoT Device owner delegates in access (read-only) data to another user, or group of users/organizations. The delegation can be performed at level of single sensor or for the whole IoT Device. When the delegation is removed, the K1 and K2 are erased and will no longer be valid. When an IoT Device is declared "Public", an Anonymous delegation is generated; in this later case, the K1, K2 can be omitted, and any access operations will be always permitted. In any case, only the owner may modify the internal settings/values of the device.

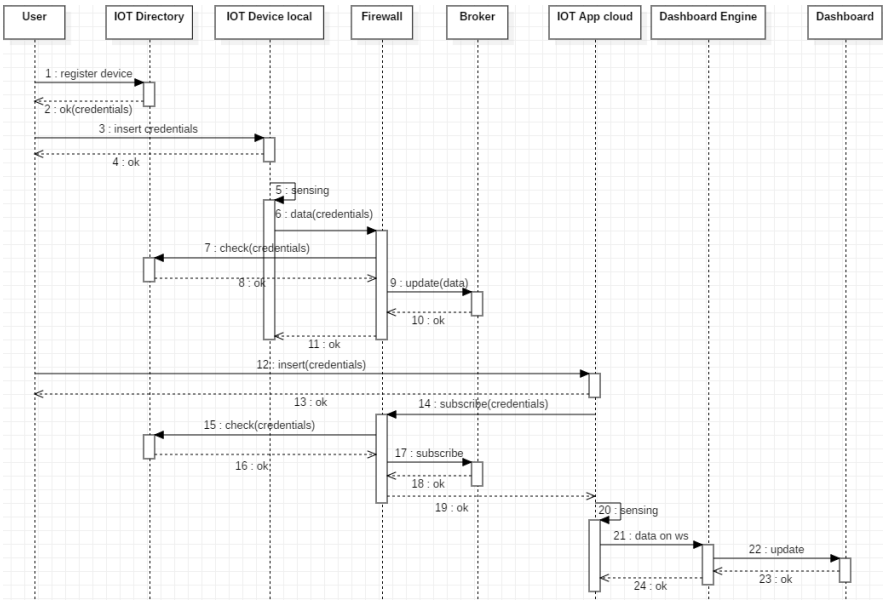


Figure 5.8: Sequence diagram of X509 certificate security enforcement

In Snap4City, a higher level of security can be chosen by the IoT Device owner in terms of the security barriers' authentication solution, which

is based on the X.509 certificate and mutual authentication. This technique involves the exchange of a signed digital certificate, based on a private-public key cryptography solution. The Snap4City IoT Directory acts as a security and identity unit using a self-signed Certification Authority. The exchanged certificates are SSL/TLS-based, to ensure secure authentication: the HTTPS mutual authentication schema between the device and the Snap4City framework is established. When a M2M connection from the IoT Device is performed, it is hard to retrieve the private keys of the IoT Devices, since they are stored in a key-secure storage (SIM card) commonly protected by an additional device password. In detail, the complete flow (see Figure 5.8) implies the creation of a private secret when the device is registered in the IoT Directory. Later, a digital certificate related to the private secret is digitally signed by the Snap4City framework. This signed certificate, with the private secret and the Snap4City certificate, are injected in the IoT Device in order to establish a mutual authenticated communication between the IoT Device and the platform, which is the IoT Firewall in this case.

In Snap4City, a mixed authentication model with a moderated level of security is available, when the IoT Device has very low resources and cannot support the complete SSL/TLS stack. It relies on the usage of the K1, K2 authentication system described above, plus the ability to verify by the IoT Device of the Snap4City endpoint by using the thumbprint (SHA, SHA3, Secure Hash Algorithms) of its public certificate to establish a secure HTTPS communication. This information can be recovered from the certificate metadata or can be easily estimated having the certificate only.

5.5.2 IoT Applications vs security

As discussed in the above mentioned requirements and in the general architecture, IoT Applications exploit data access from storage, and connections with IoT Devices, Data Analytics, and general MicroServices. Thus, the Snap4City users can create IoT Applications with business logic using the Node-RED environment enriched by a large set of Snap4City nodes/MicroServices [52]. The IoT Applications can be executed on the cloud or on premise as IoT Edge Devices. Therefore, different security approaches have to be enforced to support all cases and scenarios.

When an IoT Application is executed on the cloud, it runs in a Docker container on an Apache Mesos & Marathon cluster. Web access to the Node-RED user interface is guaranteed via a reverse proxy configuration

that follows the possible reconfigurations of the container cluster. An ad hoc authentication and authorization module has been developed to interoperate with the security management provided by Node-RED. A so-called Strategy has been written to plug the intelligence of the user access control to rely on the Snap4City infrastructure. Some adaptation of the Node-RED code was made to enable access to users with different kinds of roles. Any time a user accesses an IoT Application, the login is made using the same approach as any other Snap4City modules, retrieving from the SSO server a refresh token that is exchanged with an access token any time the IoT Application needs to communicate with any other Snap4City module. To avoid the user capability to login at any time, and thus, to cope with M2M communication, the refresh token elapses and it is refreshed automatically by the platform in any 8-hour period. Since the IoT Application runs on a cluster and over time it can migrate between different nodes, the refresh token is stored in a local secure storage with an exclusive access of the specific IoT Application. Despite the flexibility, for security reasons, the users cannot load additional nodes autonomously from the Node-RED Library; they have to ask at the administrator to validate and load them. The cluster of Docker containers is also continuously monitored by the AMMA (Application and MicroService Monitor and Analyzer) tool for assessing the volume and messages exchanged on it [59].

In case the IoT Application is executed on premise/on IoT Edge Device, the User Interface is usually available only in a direct cable connection by the IoT Edge owner. In this case, if the business logic (IoT Application) written by the user programmers needs to access some functionality of the Snap4City on the cloud (i.e., MicroServices), the credentials of the user can be inserted manually and directly into the Node-RED flow, and the platform will take care to exchange them in favor of a valid refresh token and follow the same scenario previously described.

5.5.3 Security of IoT Network vs dashboards

In the context of IoT, the communication modalities with Dashboards may be very complex since they are usually capable of recovering and sending data via multiple channels and sources. For example, they establish direct connections with: storage, IoT Brokers/Devices, IoT Applications (on the cloud and/or on IoT Edge), and some external service via Rest API, etc. These M2M connections present different modalities to establish authenti-

cated and secure communications. In the context of IoT, the most interesting are those with IoT Devices/Brokers and IoT Applications since the others can be established as already described in the previous sections. Please note that both communications, namely, (i) IoT Devices/Brokers – Dashboards and (ii) IoT Applications - Dashboards, are all bidirectional, real-time, and data-driven modalities.

In Case (i), the secure or unsecure communication could be established using device/broker protocol and it may be difficult to acquire live updates on the user browser, for example, using MQTT, NGSI, etc., over TLS (the only solution would be to perform from Web client a polling on IoT Devices/Brokers). Thus, a solution that has been adopted has been to connect to an intermediate server that provides connection with Dashboards on the user browser via WebSocket secure, (WSs), in push. This solution has also been adopted for Case (ii), where IoT Applications/Node-RED presents Dashboard node MicroServices of Snap4City. WS protocol that allows data driven communications among the Internet browsers with the intermediation of a WebSocket Server. The secure communication is achieved by using a cluster of web socket-based applications that forward the messages to the proper connection. For example, a message produced by an IoT Application for a Dashboard's widget is taken and then forwarded to all the browsers that are currently viewing that particular widget (a sort of WS Broker). Similarly, an input widget on a Dashboard produces a message that is forwarded to the IoT Application managing the widget. A JSON-based protocol has been developed to achieve the bidirectional communication with access control, where the access tokens are used to guarantee user identity and to check if the user is able to perform the requested action (unless it is a public dashboard).

5.6 GDPR and delegation system

At the moment of the registration, the user has to be informed about which Data Type (personal data of the person) will be collected. In addition to this notification, a signed consent is requested. All the collected Data Types need to provide a user interface for review, download and delete. The user may decide to delete them, while, for the law enforcement agency, the data would not be immediately deleted; they are moved and definitively deleted only after 30 days. In addition, the Snap4City platform permits the users

to delegate access to their own data to any other user or anyone belonging to a specific group/organization, as already specified above regarding IoT Devices and their sensor's values. The delegated users have read-only access to each specific delegated data. A delegation can be any time reviewed and revoked, and it is not passed in cases of data change of ownership or cloning. A delegation can be created on top of the IoT Directory (IoT Device and sensors/actuators), Dashboard Builder (view of personal data) and IoT Application (user generated data).

A pseudonymization system management and the encryption of recorder data are designed by default to prevent identification of any specific individuals from compromised data in case of breach events. The Snap4City solution relies on storage where the links between the user's personal information and its data are stored completely decoupled via the use of a user's identifier shared between the MyPersonalData storage and the LDAP/CRM/Keycloak Snap4City's modules (Pseudonymization). In some cases, due to the large amount of data that can be generated, different storage systems could be involved to record the different user's data. The data storages are located in a set of proper servers, not directly accessible from any external access except from a set of authenticated MyPersonalData APIs. The Database is protected from external access via Tablespace Encryption, where any tables included in the MyPersonalData Database are protected by a set of keys recorded in the database itself. These keys are protected by an external Master key, memorized out of the database and accessible just by the system administrator (a superuser not registered in the Snap4City framework that is kept completely out the scope). Several techniques can be enabled on top of this separation (Master key rotation) to provide even more required security (Encryption). As already stated before, the in-transit encryption is guaranteed using always the SSL/TLS stack for network connections. Any modules of the Snap4City framework that need in some way to access some data for a specific use would need to use the authenticated APIs of the MyPersonalData module.

A valid access token (in JWT format) has to be presented, that specifies the credentials of the user who requests the specific resource. The MyPersonalData eventually authorizes the access, if the module matches the ownership or the delegation of the specific requested data (Access Control). The detailed chain of trust of the Snap4City platform is highlighted in Figure 5.9.

In event of a data breach, the Snap4City framework is able, in a timely

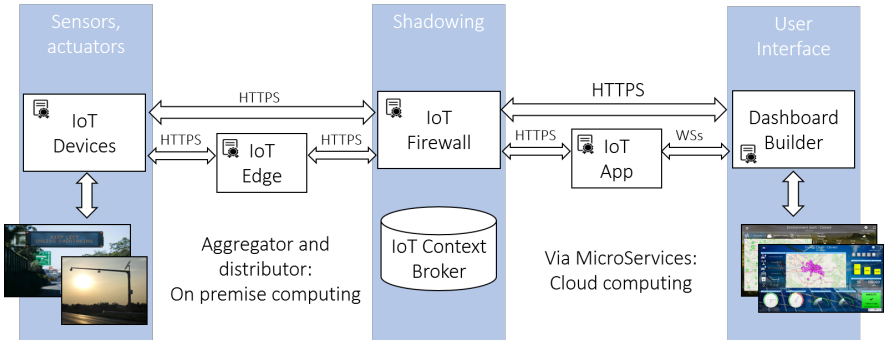


Figure 5.9: Chain of trust in Snap4City architecture

fashion, to detect and report on the issue and generate a set of records of what activities had been performed against the data. Monitoring in real time on different levels of detail is possible in the Snap4City framework by the system's administrator, via a system of notifications that mainly employ the sending of detailed emails. A set of managements tools are also able to provide constant monitoring of the different modules, services and databases' behaviors to proactively mitigate the risks. A set of thresholds and personal notification's messages are configured on the different analyzing's tools (Monitoring and reporting). Moreover, any Snap4City module is accompanied by an auditing user interface, where the activities performed on the modules and the requested service are graphically displayed for an easy and quick forensic analysis requested by the controller. Some other views on the activities on the modules are added for specific purposes, for example, a complete trace logging of the violation (request refused by the module) invoked on the MyPersonalData module (Auditing) (see Figure 5.10).

5.7 Experimental results

The validation of the Snap4City platform has been performed by several teams testing and stressing the platform against more than 150 different test cases and scenarios that are listed on `https://www.snap4city.org/108`. The PCP took approximately 18 months for the final validation (third Phase of the PCP), including different kinds of validations for developers with

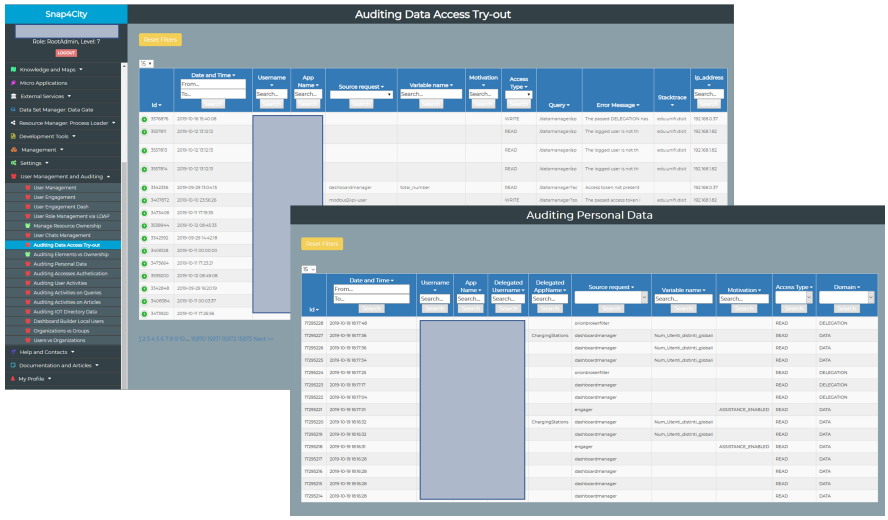


Figure 5.10: Auditing violations on the access to MyPersonalData, and Try-Out. (some details have been obscured for privacy)

Hackathon, with ICT officials for functional and nonfunctional requirements, for City Operators with dashboards and operative cases (see public Dashboards for Antwerp and Helsinki on Snap4City.org), for final users with mobile Apps and thousands of users, and included stress tests on the cloud and diffused PENetration tests. Thus, Snap4City competed in the PCP with more than 25 different smart city platforms. Two mixed teams of the cities of Antwerp and Helsinki, with several people (including IoT experts, usability experts, ICT experts, and legal experts) have verified the aspects reported in the previous Table 5.2 point by point for the requirements, and in Table 5.3 for GDPR vs Requirements. Snap4City resulted in being the winning finalist of the many Smart City IoT Platforms, as described in web page: `https://www.snap4city.org/558` and in which you can see the award, videos, and the link to the list of requirements that were overcome and validated, and in the web pages of Select4Cities. Moreover, the competition has been against requirements and their effective implementation and validation via test cases, all of which are accessible on www.snap4city.org.

5.7.1 GDPR Compliance vs Requirements

The demonstration that a platform is GDPR-compliant is a very complex task, and there is lack of official tools for the verifications. On the web and market there are a number of checklists that can be adopted for the verification that a data management process is GDPR-compliant and only a few that may help the developers to test if their WEB solution is GDPR-compliant. It should be noted that, some of the GDPR aspects may be evident from the user interface, UI, and thus they can be verified by external testers, while others (such as encryption, security level, etc.) can only be tested by code inspection and/or performing the penetration test (pentest) as described in Section 5.7.2. Interesting proposals have been suggested as a partial checklist on [32] and [16]. The list reported in Table 5.3 reports the approach adopted for verification of GDPR compliance of Snap4City. In particular, it takes into account the UI aspects, source code access and demonstrations, as it was requested by Select4City to assess the platform.

Moreover, in Table 5.3, each major feature to be compliant with GDPR has been related to the proposed requirements of Section 5.3. The detailed verification report would likely encompass some hundreds of pages and thus could be accommodated to the space constraints of this article.

Table 5.3: Criteria for GDPR Compliance verification features vs verification approach (Verif.) and main requirements of Section 5.2 (Reqs.).

GDPR Compliance Verification Feature	Verif.	Reqs.
Signed consent	UI	R8
User profile management and control	UI	R13
Data Type private as default	UI	R8
Rights to access per element	UI	R9
Rights to transfer per element	UI	R10
Rights to erase per element and total	UI	R13
Rights to revoke/change per Data Type	UI	R10
An interface for Right management for Data Type	UI	R9
Clear Terms of Use and Privacy Policy	UI	–
Auditing Tools for Data Type	UI	R14
Publish as Anonymous	UI	R9
Encrypt personal users' data	Code	R12
Secure Authentication and Authorization	Code	R3
Data protection by Design	Code	R17
Secure connection	Code	R6
Security Control, data breach control, anonymization, etc.	PEN Test	R15, R16, R18

The set of GDPR features is not strongly related to the application domain of the smart city platform in terms of mobility, energy, home, etc., while it is mainly related to the usage of the platform, so that it refers to

the applications. In all of the smart city domains mentioned, the user may be involved or not. An application on mobility for managing traffic without providing personalized services would not need to collect personal data; neither would it need to have citizens registered on the platform. The same can be stated for energy: the reporting of building energy consumption in an aggregated manner is not personal data. Therefore, the mapping of GDPR aspects vs domains would be very difficult to be realized without describing the applications. In Snap4City, the applications may all involve final users, in any domain. This implies that personal data have to be treated as described in the paper, disregarding the application domain. It is also very restrictive to think that the application would be domain-oriented for the data. For example, parking predictions are based on historical data of parking but also on traffic and environmental data, and events of people. This is the strong point of Big Data.

5.7.2 Security and Privacy assessment

To assess the security and privacy of the solution, a number of penetration test's activities have been performed in the period April-July 2019 by professional companies of the sector. During the period, the Snap4City platform had approximately 1200 operative users on the Web Portal, 1.8 million of new data per day, 6 organizations that were operative (Florence, Helsinki, Antwerp, DISIT, Sardegna, and Garda), more than 4500 data ingestion processes executed per day, more than 320 IoT Applications running on the cloud and on IoT Edge Devices, more than 50 processes of Data Analytics, approximately 840 Dashboards of which 200 public, more than 300 MicroApplications of HTML5, 15 IoT Brokers, 5 Mobile Applications connected on Smarty City APIs, with approximately 3500 distinct active users on Mobile Apps daily, etc.

Any detected issues have been analyzed and for each of them a set of counter measurements have been taken to make the platform more robust to external intrusion and attacks. An approach of incremental tests has been chosen to spot the largest number of weaknesses and to enable a progressive evaluation-and-patching process. Each phase has been carried out by different actors such as:

- first phase: two groups of internal developers and security experts that know very well the Snap4City infrastructure;

- second phase: two groups of external testers that know nothing about the Snap4City infrastructure and that had very low interaction with the Snap4City developers;
- third phase: two third-party companies that executed the test without any interaction with the Snap4City developers; one of them has a high ranking status and reputation on security issues and professionally works in the penetration test field of study.

During the period in which the penetration tests have been performed, an activity of a stress test was also carried out. Even if the focus of the pentest was not about analyzing how the platform responded to a high workload of requests, some feedback on the platform's performance has been collected to also improve this aspect of the platform and thus resilience on the workload and DoS (denial of service) attacks on APIs.

The penetration tests were performed following the following steps:

- Intelligence gathering activities against a target: in this step, information about the target has been acquired using the OSINT (Open Source Intelligence) public accessible sources, and the information was collected using different tools such as: SpiderFoot (`https://www.spiderfoot.net`), Maltego (`https://www.maltego.com`), Shodan (`https://www.shodan.io`).
- Service detection and identification: in this step, the target has been analyzed to find the services exposed and the versions of the used tools. This activity has been performed using tools such as: Necfraft (`https://www.netcraft.com`) on the main Snap4City domain, ZenMap to identify its open target ports (`https://nmap.org/zenmap` such as 80/tcp 443/tcp 5060/tcp 8080/tcp), Google Dork (`https://securitytrails.com/blog/google-hacking-techniques`), Nikto (`https://cirt.net/Nikto2`), DirBuster (`www.owasp.org/index.php/Category:OWASP_DirBuster_Project`). The theHarvester script has been used (`https://github.com/laramies/theHarvester`) to retrieve any user's profiles leaked by the platform; no profiles were found, except the only one publicly presented as a contact point of the platform for external enquiries.
- Vulnerabilities detection, verification and analysis: in this step, the target has been analyzed and deeply scanned for the different vulner-

abilities using tools such as OWASP ZAP (`https://www.owasp.org`), Pentest-Tools and Burpsuite (`https://portswigger.net/burp`) and verified using sqlmap (`http://www.sqlmap.org`) and xenotix6 (`www.owasp.org/index.php/OWASP_Xenotix_XSS_Exploit_Framework`).

For example, the complete crawling of the identified contexts performed via the OWASP ZAP and w3af tools was performed from the point of view of (i) a user not registered and (ii) a logged user. Considering that any external links pointing to services out of the context have been excluded, a list of 73 ANAME records has been identified for a total number of approximately 8500 unique URLs that identified the attack surface where the following deeper analysis was performed.

A detailed set of attacks has been performed on domain and subdomains using the OSWAP ZAP and Arachni (`https://www.arachni-scanner.com`) tools. Even proceeding separately by subdomains, any complete analysis required several hours of computation. An analysis of the false positive alerts has been carried out to highlight just the true risks. For the main domain, 3095 URLs have been identified and a complete attack required more than 750 thousand requests for a total of approximately 39 hours of computation.

The pentests found some vulnerabilities that have been immediately solved, as described in the following and in particular:

- few "SQL Injection" vulnerabilities were found by using the OSWAP ZAP, Sqlmap (`http://www.sqlmap.org`) and Gobuster (`https://github.com/OJ/gobuster`) tools. These problems were found mainly in the Dashboard Builder and management modules. The problems identified were solved adding checks on the API parameters provided in the HTTP GET or POST requests; a specific list of 18 URLs (over a total of 244 URLs with high risk) has been identified and simulated attacks have been carried out manually and solved. The most relevant vulnerabilities were related to SQL injection of malicious code during the editing of information related to Dashboards and to Remote OS commands script injection permitted in a specific form of a User Interface exposed by the platform.

- a "Broken authentication" vulnerability was found as it was possible to do a password-guessing attack; to prevent this, it is set to not accept a login for 2 minutes after providing 5 wrong passwords.

- few “Sensitive Data Exposure” regarding test users and passwords and a test private key present on GitHub were removed and changed; moreover, in some cases, the listing of directories was enabled;
- one “Broken Access Control” for a possible local file inclusion was found that was present in the CKAN tool used for managing open data. Moreover, in the process of fixing the SQL injection problems, also the potential broken access controls were identified that allowed a user to access data of another user by manipulating the API parameters; in this case, stronger controls were added to prevent this situation;
- few “Security Misconfiguration” vulnerabilities were found, and, in particular, were identified as a web server supporting the TLS1.0 and TLS1.1 protocols;
- some “Cross-Site Scripting” vulnerabilities were found, mainly in the Dashboard Builder and management module. The identified problems were solved escaping the input parameters using html entities when the input is not a html content; otherwise, only the script tag is sanitized;
- regarding “Using components with known vulnerabilities”, the vulnerable versions of nginx and Apache2 web servers were detected and upgraded; moreover, a vulnerable version of Drupal 7 was found, which was at the basis of the “Snap4City Platform Support Living Lab” module, and it was upgraded to the latest version available.

For the other types of vulnerabilities, as mentioned in Section II.A, such as “XML External Entities (XXE)”, “Insecure Deserialization”, and “Insufficient Logging & Monitoring”, nothing was found.

5.8 Considerations

The shift of paradigm from completely central elaboration in the scenario of Big Data and the Smart City IoT toward a more distributed computation with edge computing has required different approaches in designing a platform that supports security by design. The Snap4City Smart City IoT architecture is composed of several modules that elaborate MicroServices that can be accessible for IoT Applications on the cloud and on premise, which is on the IoT edge. In those cases, security has to be addressed by

design and by default about the privacy and protection of the managed data since in most cases, data can be very sensitive, due to their nature of city or personal data of the Living Lab users. Different levels of protections have to be offered on the basis of the sensitivity of the carried data and different requirements have to be addressed and satisfied in different use case scenarios (e.g., medical assistance, engineering and architecture, entertainment, city risk assessment, and city resilience). In this paper, Snap4City architecture and security solutions respecting the GDPR of the European Commission have been presented. The solutions addressed the full stack from IoT Devices, IoT Edge Devices on premise, IoT Applications on the cloud and on premise, Data Analytics, and Dashboards. Snap4City has been produced in response to the challenge launched by Select4Cities H2020 of the European Commission. Select4Cities identified a very large number of requirements for modern Smart Cities supporting IoT/IoE (Internet of Things/Everything) in hands of public administrations and Living Labs. In that challenge, Snap4City demonstrated to have satisfied all the requirements proposed. Moreover, innovative solutions have been proposed, and new specific requirements that were not identified since the beginning of the project have been outlined (and are reported in this paper). Therefore, as claimed by the evaluator, the Snap4City solution has gone beyond the expectations, namely, in the satisfaction of requirements including GDPR, on the innovations and with respect to the state of the art during the validation performed in the PCP with specific pilots and stress tests in Antwerp and Helsinki in Europe (they are top-level smart cities). The stress security assessment has been performed in a piloting period with more than 1200 registered users, thousands of processes per day, users on mobile Apps, and more than 1.8 million of complex data ingested per day. The Snap4City architecture and solution described in this paper comply with the high security level and satisfy the GDPR of the EU. In the validation, Snap4City has also been stressed and tested by using several Penetration tests that allowed identifying a few vulnerabilities that have been solved in the current validated version. Thus, the solution guarantees an IoT end-2-end high level of trust for the current supported technologies in terms of security and privacy aspects, addressing security in the stack that includes: IoT Devices, IoT Edge Devices, IoT Applications (on the cloud and on IoT Edges), Data Analytics, Dashboards, and Smart City APIs for Mobile Apps in which the security level is also supported. Snap4City is completly open source and license/patent free, and it

also currently in use in several cities in Tuscany (central Italy including Florence), in Antwerp, Helsinki and Lonato del Garda. Snap4City is a solution produced in response to a research challenge launched by the Select4Cities H2020 research and development project of the European Commission. Select4Cities identified a large number of requirements for modern Smart Cities supporting IoT/IoE (Internet of Things/Everything) in the hands of public administrations and Living Labs, and selected a number of solutions. Therefore, at the end of the process that took 3 years of work, Snap4City has been identified as the winning solution.

Chapter 6

MicroServices Suite for Smart City Applications

This chapter presents a suite of MicroServices for the Snap4City platform that enable the creation of a wide range of IoT Applications for Smart Cities. They involve the use of several component present in the IoT architecture, like Dashboards, IoT Devices, Data Analytic, IoT Discovery, etc. A complete list of requirements that focus on IoT Application in Smart City context has been collected and the solution proposed has been validated against a large number of IoT Applications for the cities of Florence, Antwerp and Helsinki. In addition, three applications have been presented for validation, putting in evidence the relationships among MicroServices, Dashboard and Data. [1] [2]

6.1 Introduction

The concept of Smart City is becoming increasingly important and pervasive, making it a major area for research, business and policy makers [116].

[1] Part of the work presented in this chapter has been submitted and is currently under review as "MicroServices Suite for Smart City Applications" for *Journal Sensors* .

[2] *Acknowledgments:* this work was partially supported by the European Union's Horizon 2020 research and innovation program, with the "Select4Cities" PCP project (within which the Snap4City framework has been supported) under grant agreement No 688196, the University of Florence and all the companies and partner involved in Snap4City

Smart Cities are currently required to provide flexible architectures, in order to satisfy new functional and non-functional requirements that arise and constantly evolve in many different contexts and environments. There is also the necessity of scaling up to handle the increasingly number of users, devices, and services. However, the current common approach is still to develop ad-hoc applications for specific domains, which leads to the production of non-interoperable services [86]. Smart Cities are becoming every day more focused on activities related to IoT and IoE (Internet of Things/Internet of Everything). This has provoked a switch of technology and approach in field data collection. In the past, most of the city generated data have been collected from domain specific vertical applications, by sending data to dedicated concentrators (for example, the collection of data from traffic lights, traffic flow sensors, parking, etc.). Each and every vertical application was using its own solution for data collection either via radio, or any other means. On the other hand, the MicroServices architecture paradigm has arisen from wide-scale industries requirements of building and maintaining large-scale distributed systems [134], supporting an adequate availability, scalability, modularity, as well as flexibility [102]. The new push of IoT is stimulating cities to adopt the same gateways to collect data of multiple services. This means that the infrastructure of the sensor network has to be properly designed and planned by the city operator. LoraWan can be an option [114], while in many cities 5G solutions are coming and would replace former technologies in short time, at least within the large cities [138].

However, the arrival of a more aggregated IoT streams via 5G is not a problem for smart city data aggregators serving the Smart City Control Room [61]. Presently, they collect data from several heterogeneous sources, and to have a reduction in the number of data concentrators to be aggregated would be a benefit not a problem. On the other hand, the IoT/IoE paradigm is also strongly impacting on the infrastructure management [65], [137], since the systematic adoption of IoT has also led to the adoption of PUSH approach (event driven protocols) and it has brought forth the possibility of acting on actuators and not only sensing the city, not to mention the possibility of creating Event Driven Applications. They are also called IoT Applications which have to be capable to process messages and produce reactions in real time [195]. For many aspects Smart City application are one of the most complex cyber physical systems, CPS, due to their complexity in terms of data, data analytics, and interfaces with the real world, physical

and digital on the user interface.

In the state of the art, several different definitions and architectures have been proposed for IoT frameworks [189]. Most of the IoT platforms adopt a code-based development for the design of IoT Apps. Some of them provide development tools as: Eclipse, or JavaScript. Others provide simple tools as IFTTT [20], relying on rule-based scripts [47]. Other platforms adopt solutions similar to ETL (Extract Transform Load) processes [30] capable of working for event driven processes, or just surrogating it with high rate polling. One of the most promising tools for creating IoT App is Node-RED [27], based on Node.JS, which has been proposed by JS Foundation as fully open source. The Node-RED approach is a mix of visual composition of nodes/blocks to compose the so-called flows that are concurrently executed by a Node.JS engine. It is quite diffused, being also directly provided into official releases of IoT devices as the Raspberry Pi family. It is provided with a minimum set of functionalities (the building blocks/nodes), while other blocks can be easily added loading them from a large library made available by the JS Foundation. The classical nodes provided in the standard version can be classified as: input, output, function, social, storage, analysis, advanced, and dashboard; such nodes are not sufficient for creating Smart City IoT Applications. Actually, for the use in the context of Smart City it was not powerful enough to cope with the basic requirements of such domain, even though its use in the field is quite widespread. In the next Section, many other solutions have been considered and compared, in order to select the starting point of our research and innovation action, i.e. the Snap4City platform funded by European Commission, as described below.

This chapter is structured as follows. Section 6.2 details the related works. In Section 6.3, requirements for a MicroServices based programming solution for smart city applications are analysed and reported. Section 6.4 presents the Snap4City Architecture with a stress on data flow, IoT and MicroService aspects. In Section 6.5, the Snap4City MicroServices Library is presented and motivated. Section 6.6 introduces some applications exploiting the MicroServices in the context of emergency management, personal mobility, and crowd of people monitoring with mobile PAX Counters. In Section 6.7, the Snap4City development life cycle has been formalized. Section 6.8 presents a validation of the solution performed with city officials in developing IoT Applications. Conclusions are reported in Section 6.9.

6.2 Related works

Several Smart City use cases have been presented so far in literature, transforming the collection of huge quantities of data related to urban environments into useful content for various stakeholders (citizens, tourists, local government and companies) [140]. Among them, some of the most notable use cases concerns the cities of Santander [91], Manchester, Barcelona [39], Singapore [71], Seoul, San Francisco [135], Rio de Janeiro [132] and Florence. Most presented solutions however still focus on specific domains, aiming at resolving particular problems, with little software reuse [185].

In order to manage the high variety of devices and applications, including IoT, mobile, web and service-oriented framework, the MicroServices architecture is being increasingly adopted in recent IoT based solutions for Smart Cities. Actually, Microservices architecture focus on the development of simple and loosely coupled services [219], enhancing scalability and availability, facilitating maintenance and fast isolated testing. These aspects aim at simplifying the complexity of traditional service-oriented architecture (SOA) [94].

The programming of IoT Applications is performed in several different manners by different tools [212]. As described in Chapter 5, in AWS IoT ecosystem a complex stack of Amazon's services is proposed, which enables developers to define any business logic by programming the flow of IoT Devices' data towards their visualization in proprietary dashboard. The IoT Devices can activate AWS functions written in: Java, Python and C#. These functions can implement business logic or trigger actions in the IoT world or Cloud. Data rendering is flexible and can be used for data inspection and reporting. Azure IoT provides a development environment to build business logics. The integration with mobile applications is performed by using the Xamarin platform in C# and .NET. In addition, it is possible to write application also in Java, Node.js and Python. For the creation of dashboards, developers can use MS Business intelligence tool. In Google IoT platform, several programming languages can be exploited to program the data flows from device to dashboards, like Java, Node.js, Python, Go, Ruby, PHP, C#. In such cases, data flows are implemented by using programming languages. Not any of them provide a visual editor for IoT Application flow definition to be used even without any software programming skills. In addition, these solutions often require the mandatory use of cloud services, and they are not suitable for any autonomous deploy on-premise scenario. As a limit case,

they may permit to have some modules configured on premise IoT Edge with reduced capabilities connected with the cloud solution.

In [173], a survey on Visual Programming Languages (VPL) for IoT has been proposed. In the survey 13 different VPLs have been considered. The analysis has been mainly focused on comparing them on the basis of the programming environment, licensing, project repository, and supported platforms. Some of them are Open Source platforms, while others are proprietary. Among the Open Source platforms (Node-RED, NETLab, Ardublock, Scratch, Modkit, miniBloq, NooDL), only some of them can be programmed using a Web interface and can be also executed on some dockers or virtual machines on cloud. In [83], the usage of haptic interfaces in the context of IoT Applications for smart manufacturing has been reviewed. The analysis has produced a classification and the identification of the main elements needed to the exploitation of haptic devices without entering the problems of the microservices and VPL for programming IoT Applications.

In the assessment of Open Source VPL for IoT, several important aspects should be taken into account, such as:

- performance in terms of number of messages processed per second, per minute;
- diffusion of the tool, namely the size of the communities supporting the tool, both as developers and users developing applications and additional features;
- openness in terms of flexibility to create new blocks to extend the basic functionalities, for example, exploiting external services, developing data analytics, etc. For example, in the domain of smart city or industry 4.0, you could add visual modules representing the IoT Device elements, as well as Data Analytics;
- Cloud/docker support for their use on cloud and/or support for any execution on embedded IoT, thus executability on several operating systems. Some of these VPLs produce code only for specific embedded devices. On cloud the automatic resource provisioning and the elastic scaling of resources are needed to manage large amount of IoT applications;
- level of expressivity: an IoT VPL can be oriented on defining functional or data driven/flow aspects. The first are typically less expres-

sive, since such constructs are related to the programming language: section, assignment, condition, iteration, etc. Data Flow models are more expressive and may be easily extended with new blocks/functions and this lack of expressivity may be solved if the right blocks are identified. Therefore, this can involve as a result to produce simple data flows even for complex problems. In fact, the complexity of the graphs produced for solving problems in the applicative domain can be measured in terms of number of blocks/modules and connections used to solve such problems;

- usability, including visual editing of data flow via web browser or through specific client tools, etc. The fundamental features, such as: inputs, outputs, functions definition, social gathering, storage load-/save, data analytic and dashboard (user interface), may be not enough for creating complex Smart City IoT Applications with low effort. On the other hand, one may have a well-designed VPL, and the effort may be vanished due to the lack of expressive and domain suitable MicroServices;

- managing resources: the possibility of developing IoT applications (as well as portions of them) as data analytics and share them in a marketplace. In the context of VPL for IoT, the development of Data Analytics is quite complex, since processes may have a lot of dependencies and the programmers' skill may imply the knowledge of the data storage and models, in addition to the knowledge of algorithms for data analysing. This approach may require the involvement of experts of sensors, algorithms, infrastructure, programming, knowledge modelling, domain etc.

Some solutions that can be suitably used to address such aspects and overcome the problems described above are, for example: Node-RED (it can be executed on multiple operating systems, it has a visual programming via Web interface and the source code can be loaded on several different embedded devices, etc., [27]), NETLab is an open source development environment for embedded system [26]; Ardublock is an Open source block programming for Arduino, mainly functional approach and limited to specific devices [3]; Scratch is a visual tool of MIT, adopted for code generation for IoT, very low level programming model, not event driven, functional; Modkit is strongly oriented to Embedded IoT devices; miniBloq is a block

functional programming at very low level; NooDL allows the fast prototyping of the user interface and it is less suitable for data analytics; VISUINO is an open source tool to develop code for embedded IoT, but providing limited capabilities to be used on cloud; Kura is an Open Source Java visual tool for IoT; FLOGO is an Open Source tool for data flow in Go, natively works on AWS Lambda, [28] or Wyliodrin a development environment for IoT edge embedded devices, deploy and update.

Most of the above solutions are strongly focused on producing code for embedded devices with functional model and not data driven/flow, such as: Scratch, Modkit, miniBloq, etc. Therefore, they are not suitable for creating IoT Applications that can be put in execution on IoT Edge, as well as on cloud docker containers, addressing the complexity of Data Driven applications and flows; exploiting Data Analytic. Solutions such as AWS, IoT Azure, and IoT Google are more oriented on traditional programming. The VPL tools based on data driven, open to extensions, supported by large communities, that can be put in execution on both edges, could be only a few like: Node-RED and maybe the emerging FLOGO. In [179], Node-RED, flow editor has been used for creating smart city IoT applications for projects such as VITAL. The suite of nodes implemented is not focused on functional aspects of IoT Applications, so that the resulting flows are still too much focused on technical aspects and not user centred. In fact, as depicted in [179], even the creation of simple activities as the identification of traffic flow sensors results into the development of complex workflows with a high number of nodes/blocks. Specialized workflow formalization models have been also proposed, such as in [73]. In [89], a Model Driven Development (MDD) process has been proposed consisting of (i) a tool for creating semantic algorithms, and (ii) a workflow generator on the basis of the first results. The user interface is a subset of Node-RED environment.

There is a few IoT development environments where algorithm repositories are provided for their exchange in the community, such as in [160], [176]. This feature should be mandatory in cloud based IoT development environments such as: Axeda [31], BlueMix [19], ThingWorx [37]. On this last aspect, there are two levels in IoT valorisation environments, such as:

- IoT platforms marketplaces, such as BeeSmart City, Fi-WARE, EOSC, etc., which are typically constrained to a class of solutions or open to all. Their target is the promotion and marketing of IoT or Cloud solutions;

- IoT Applications and solutions resources sharing and marketplaces, such as: Node-RED library, which are typically focussed on specific IoT kind of development processes, and on promoting the interchange.

In this chapter, the issues related to Snap4City IoT development environment and framework are addressed. The experience described herein refers to the design and implementation of Snap4City platform, `https://www.snap4city.org` [48], which is based on Km4City [49]. Snap4City is the solution produced in response to a research challenge launched by Select4Cities PCP (Pre Commercial Procurement) H2020 research and development project of the European Commission (`https://www.select4cities.eu`). Select4Cities has identified a large number of functional (mainly Smart City IoT) and non-functional requirements (open source, scalability, security, GDPR compliance, working on cloud and on premise, etc.) which are fully described in their web site and aimed at creating the best solution for the modern Smart Cities supporting IoT/IoE, in hands of public administrations and Living Labs. Most of the identified requirements have been taken from the large association of Living Lab ENOLL (European Network of Living Lab association, `https://enoll.org`), and upon consultation of smart cities at level of European Commission as EIP-SCC (European Innovation Partnership on Smart Cities and Communities, `https://eu-smartcities.eu`).

Snap4City has responded to the research challenge and could prove to satisfy all the Select4Cities requirements. Snap4City allows the creation and management of users communities, which collaboratively: (i) may create IoT Solutions, (ii) exploit open and private data with IoT/IoE respecting GDPR, and (iii) create/use processes and IoT Applications that could run on Edge, Mobile and cloud, with the capability of interacting one another and with users via messages, Dashboards and Applications. A specific attention has been given to the creation of an integrated development environment for IoT App (cloud and edge), based on VPL, with dashboards, and supporting data analytics and resource sharing, thus minimizing any required technical expertise as programmers. To this end, a large number of visual elements have been identified and developed as MicroServices for Smart City IoT. The starting point has been Node-RED, of which we could identify many breaches to cover the Smart City domain. In addition, the developed platform has a full support to give user assistance during the development life cycle, from data gathering to dashboard production, as well as in IoT App development and sharing of results with other developers and in the community as self-

assessment, together with any security support as described in [57].

6.3 Requirements for MicroService-based Smart City Applications

As above described, the aim of the research reported in this chapter has been to design and implement a visual programming environment where city operators may develop Smart City Applications with IoT by means of visual constructs. In this Section, the identified requirements of the challenge are listed.

As a first step, by analysing the state of the art we realized that a VPL for developing IoT Applications has to provide generic requirements, so as to support:

- data communications sending/receiving messages/ data, providing both push and pull modalities. This means that the language has to be capable to perform requests using many different protocols among the available ones [92], such as pull protocols to get data (e.g.: Rest Call, Web services, FTP, HTTP/HTTPS, etc.) and push protocols to receive data via data driven subscriptions (WS, MQTT, NGSI, COAP, AMQP, etc.). In the context of IoT solution, this feature is quite common. For example, in Node-RED library, you can find a large collection of nodes covering tens of protocols, while large platforms as AWS and IoT Azure are typically limited and address just few protocols. In some cases, such limitations concern the supported security and authentication schemas, since in most cases, username and password (or authentication keys) have to be included directly into the flow in clear text, thus making not all the approaches on IoT security satisfactory [57]. For example, in Node-RED each single IoT Device can be connected to a flow using specific credentials directly included in the code. On the other hand, a generic service for IoT Discovering and Registration is missing.

- data save and retrieval to/from some bigdata storage. For example, the capability to store data and exploit them to implement algorithms for predictions, decision making processes, value trends overviews, etc.

- data transformation and processing via the algorithm formalization

and the possibility to start, pause, recall their execution periodically or sporadically, according to the arrival of some event or other firing conditions.

- calls of external services in order to enable the instrument of delegation to external solutions, when it comes to the computing and processing, for instance the computation of some complex Data Analytic algorithms, or the exploitation of external services such as the Social Media Analysis via REST Call. For this purpose, the classic Node-RED provides access to Watson IBM for machine learning and artificial intelligence reasoning. Desirable applications could be, for example, computing sentiment analysis, performing a clustering classification and recognition, computing some predictions on the basis of historical data of some City IoT values.

- the presentation of data via some Dashboards that would be the user interface of the application. Data visualization should be expressive enough, including values in real time, time trends, histograms, pie charts, etc.

- the use of the Dashboard as a user actuator interface, enabling the user to act: (i) on some internal variables, (ii) to send messages/ commands to IoT Devices and flows.

The VPL IoT platforms should present a number of non-functional requirements such as demonstrating capabilities of: robustness (in terms of availability and fault tolerance), scalability (to be capable to serve from small to very large businesses with corresponding volumes of processes per second), security (authentication, authorization, secure connection, etc.), full privacy respect (compliance with data privacy according to the GDPR, General Data Protection Regulation European guidelines), interoperability (e.g., communicating with any kind of protocols, devices, services), and openness in terms of open source and in terms of possibility of adding new modules and functionalities, etc. On the other hand, most IoT development environments are not web-based development platforms, and are focussed only on generating code for specific embedded as Arduino, see the above mentioned: VISUINO, Ardublock , ModKIT, miniBloq, etc.

In addition, by analysing the requirements identified by Select4Cities, ENOLL, EIP, etc., we realized that a large number of smart city IoT specific

requirements have to be satisfied by a VPL to enable the easy development of IoT Applications in the Smart City context, at the service of City Operators and advanced City Users (among citizens, stakeholders, third party developers of SME, researchers on data science, etc.). For example, the VPL should provide specific features to address functionalities to develop applications in domains such as: mobility, environment, parking, culture, health, tourism, energy, and general city services, with no need of knowing technical details about the provided services or service supplier identity. Actually, in almost all cities, different services on mobility, environment, tourism, energy, education, etc., are provided by different operators. Moreover, some of these services are also related one another; for instance, the number of free parking lots and the events in the area, the number of weather forecasts and the number of people in city gardens, the traffic flow and the environmental data regarding NOX, etc. That is, when it is supposed to rain, the number of traffic car increase, and thus the number of free parking slots along certain hours of traffic decreases.

On such basis, a number of specific functionalities have been identified to develop smart city applications via VPLs, which should be available in terms of MicroServices and building blocks, in order to:

- Access to Smart City data entities on the basis of their functional semantic descriptors. For example, status or prediction of "parking square Carlo IV" and not status of variable "Status" of device "sensor45", etc. ;
- Discover and exploit city entities' data regardless of their technical details, such as their push/pull protocols and gathering model, messages format, provider, etc., while focusing the search on functional aspects and ignoring all the details about the query syntax used in the specific storage (e.g.: SPARQL, SQL, ElasticSearch). For example, to provide answers to the following questions: "let me access to all temperature values in this area, path or close to a given GPS point"; or: "give me the sensors measuring PM10 (or any other environmental variables) close to my position or along a path", regardless of the actual connection of sensor devices which would be connected to the platform by using different IoT brokers, different protocols, etc.;
- Create Graphic User Interface for web and mobiles, with visual representation of complex data in dashboards, allowing also to input data

into IoT Applications and the whole system. Node-RED provides a number of basic nodes to create GUI for IoT App (Dashboard nodes), whereas the offered solution presents strong limitations as to its usage in the context of smart city. They are not: presenting maps (with pins, paths, areas, etc.), heatmaps, trajectories, origin destination matrices, comparing trends, Kiviats, multi-pie charts, etc. Furthermore, they are not protected by authentication and secure connection. In smart city domains, advanced data types need to be managed and shown, for instance: maps, paths on maps, heatmaps, collective trajectories, area shapes on maps, areas, ortho-maps, origin destination matrices, etc. Most smart city data types are related to sequences of GPS points;

- Access to Data Analytics and their production. The real need in the context of Smart City is to grant data analysts with the possibility of creating some data analytic processes (it may be in R, Tensor Flow, Python, Java, etc.) and using them into the data flows as MicroServices, with no need for a programmer to step in, nor for an administrator to take action every time. This approach may create a strong flexibility into the IoT Applications. For example, classical Data Analysis algorithms for smart city could include solutions to:
  - Compute the routing from point A to point B according to certain criteria, for example: using car, public transportation, bike or pedestrian; shortest or fastest; avoiding specific zones due to road works or other kinds of emergencies. This implies the production of a sequence of points connected by street segments.
  - Collect and compute metrics on social media data from Twitter (such as: counting Tweets, retweets, as well as extracting Natural Language Processing features [154], i.e. verbs, adjectives, citations, hashtags, and performing sentiment analysis on them). For example, computing metrics related to Tweets collected on the basis of a set of keys, Twitter usernames, hashtags, etc.
- Register new IoT Devices, regardless of their protocol, which are provided via some external or internal IoT Broker. This feature is strongly relevant to the registration of personal devices. For example, glucose meter, security devices for home monitoring, etc. The data collected from those devices have to be kept strictly private, and according to the GDPR guidelines [57], [198].

- Save and retrieve personal data, for example data coming from private devices, as those mentioned in the previous points, and thus according to GDPR. ;
- Load and publish Data Sets, aiming at ingesting open data and/or creating new open data sets, share them with other cities and networks. A data set is typically a file in some open format containing data description. For example, the position of benches in the city, the position and description of all the restaurants in the city, etc.
- Provide different kinds of events within the city. They can be (i) notifications of consolidated events (such as road accidents, fire brigade and/or police calls), (ii) entertainment and sport events, or (iii) potential emergency calls, not yet consolidated events which may be produced by police cops or citizens in the street just to communicate facts (for example, potholes in the street, a balcony with some problems), etc. They need to be profiled with a number of metadata, and in case of emergency events they should be compliant with CAP standard (Common Alerting Protocol) [10], in order to provide tools and instruments to enhance the city resilience capabilities, and be interoperable among several operators: fire brigade, local and national police, civil protection, etc.

6.4 Snap4City enanched architecture

In this Section, the general architecture of Snap4City is presented to contextualize the successive Sections where VPL aspects are discussed. The Snap4City platform has a Lambda architecture, as reported in Figure 6.1. Data are received in push and/or pull, and almost every data can be considered IoT data sources. They may come from IoT Brokers, social media, web server, web sockets, streams, etc.

In Snap4City, most data provided from External Services or operators are collected in pull via REST Calls by using scheduled processes written in ETL, through the Penthao Kettle tool. The ETL processes are executed (periodically or sporadically/on demand) on a cluster of Virtual Machines by a scalable and distributed scheduler called DISCES [49]. The collected data are regularised/reconciled according to the Km4City Knowledge Base [49], and then pushed into the Big Data Cluster storage. The reconciliation

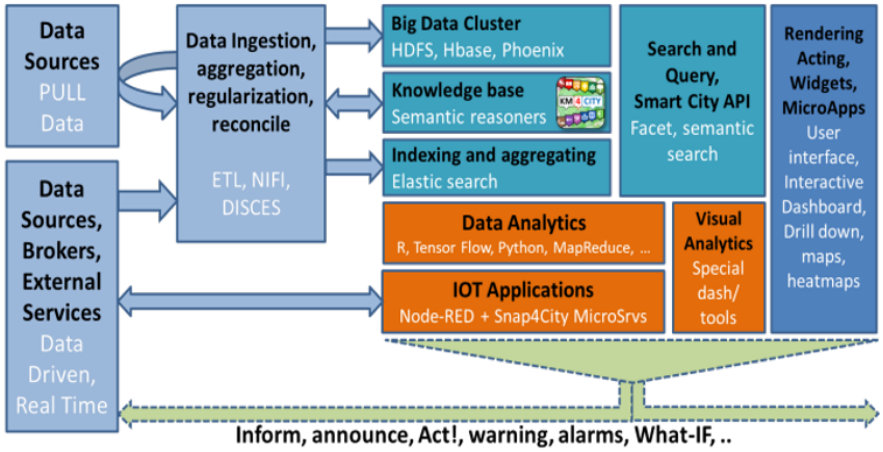


Figure 6.1: Snap4City Functional Architecture, cloud view in which the IoT Applications are only on cloud

process allows the reconnection of data to city entities already in place; for example, by connecting POI (Points of Interests) to civic numbers, traffic flow to street segments, etc. The aim is creating a fully connected knowledge base for the city, where semantic queries can be performed by exploiting physical and conceptual relationships.

On the other hand, data arriving in Push (data driven) are typically real time data streams, managed by a number of IoT Brokers which an Apache NIFI distributed cluster is subscribed to. The NIFI process is in charge of performing a regularization/reconciliation task on data, according to the Km4City Knowledge Base and then it pushes the resulting data into the Big Data Cluster storage, while indexing and thus creating the Data Shadow. The above described parallel solution tends to be normalized in a single approach based on NIFI [34] when the platform is small. Otherwise, when the platform has large data volumes, a distributed data warehouse for data ingestion based on ETL is more effective. The Big Data cluster storage includes a HDFS Hbase, a Phoenix storage, an RDF store for semantic reasoning based on Km4City (which is operatively implemented on Virtuoso store), and an ElasticSearch Index. The whole solution presents textual indexes on all fields, semantic indexes, and elastic aggregations to perform

advanced "faceted" filtering queries. The indexing and query supports are exploited by Snap4City Smart City APIs [49]. When small installations are performed, the HDFS cluster can be avoided.

The operative IoT processes (IoT Applications) can be executed on cloud or on premise. In Figure 6.1, the cloud case (private or public) is depicted. When IoT Applications are executed on IoT Edge that are located on the field, they may directly communicate to the IoT Applications or Dashboards on cloud or by means of IoT Brokers (on which all the other ca be subscribed). On such grounds, without the risk of losing generality, in Snap4City, the IoT Applications for Smart Cities are obtained as:

IoT App = Node-RED + Snap4City MicroServices

The IoT App exploits basic nodes of Node-RED Node.JS plus Snap4City MicroServices that are suitable for smart city data transformation and processing.

The Node-RED platform is based on two components: (1) a web-based visual editor to design flows, and (2) a runtime environment that may execute flows. In Snap4City, the Node-RED editor has been improved to:

- communicate with the so-called Resource Manager;
- save and load IoT Applications and manage SSO (Single Sign On) with Snap4City;
- manage a large set of MicroServices, namely the Snap4City Libraries of Nodes that are accessible from Node-RED official library.

Moreover, the runtime engine of Node-RED has been also improved to (i) manage the security according to SSO and Snap4City model, (ii) execute it on Docker on the elastic management solution of Snap4City.

On the other hand, the changes performed on the Node-RED have been released open and are functional only for use on cloud for large scale. While, Snap4City library can be used on the installation of IoT Edge with standard Node-RED tools, without any restriction.

The Snap4City solution has formalized and implemented a large set of MicroServices satisfying the above discussed requirements. The MicroServices provide an easy and formalized access to all the Smart City services which are available in cloud from the platform (including the ones to control a part of the platform itself). They are made available into the Node-RED Node.JS environment to create IoT Applications as VPL. Among the MicroServices, the IoT applications need also to access such services allowing

exploiting Data Analytics, Visual Analytics and Dashboards. The latter two aspects can be employed to create the Graphic User Interface (GUI) of the IoT Applications. These tools, orchestrated by the IoT Application flows, may automatically inform, announce, act and produce alerts and warnings on IoT Devices, networks, user interface, external services, etc., and provide support to close the loop towards the user acting/reacting on the GUI and/or Devices, including notifications.

In the deployments of Smart City IoT as a Service, SCIaaS, such as Snap4City, and in large smart city applications, a relevant number of IoT Applications may need to be deployed on the basis of the on-line requests made by users/organizations. Snap4City has been recently accepted by the EOSC (European Open Science Cloud) marketplace of the European Commission and therefore the IoT Applications need to be managed and allocated on demand, on the basis of the users requesting them, as well as to be put in execution on cloud. To this end, an elastic (vertical and horizontal) infrastructure has been created to manage in scalable manner the IoT Applications in containers [59], and mechanisms to guarantee end-to-end security [57]. In Figure 6.2, the user interface allows the user to manage its own IoT Applications, irrespective of whether they are IoT App on cloud or on IoT Edges/field, whether they are child processes/containers for Data Analytics or WebScraping, etc. Please note that different kinds of IoT Applications are represented by different icons.

The Snap4City MicroServices abstract low-level details to the programmer using a visual environment, hiding the complexity of sophisticate algorithms and tools. This is useful and suitable, for example, to provide routing, spatio-temporal search and discovery, for data analytic, dashboards, networking among IoT devices, IoT data abstraction, etc. The Snap4City MicroServices are distributed into two official libraries of Node-RED nodes by the JS Foundation portal (`https://flows.nodered.org/?term=snap4city`). The two libraries are dedicated to final users (basic), and to developers (advanced). The version dedicated to Users provides outputs of Node-RED nodes easy to be exploited for non-skilled user on JSON. In fact, most of the output produces single variables and not complex JSON structured messages. On the other hand, the version for Developers (to be installed on top of the basic version for final users), presents a number of nodes/blocks that can accept complex input JSON messages to create strongly dynamic IoT Applications.

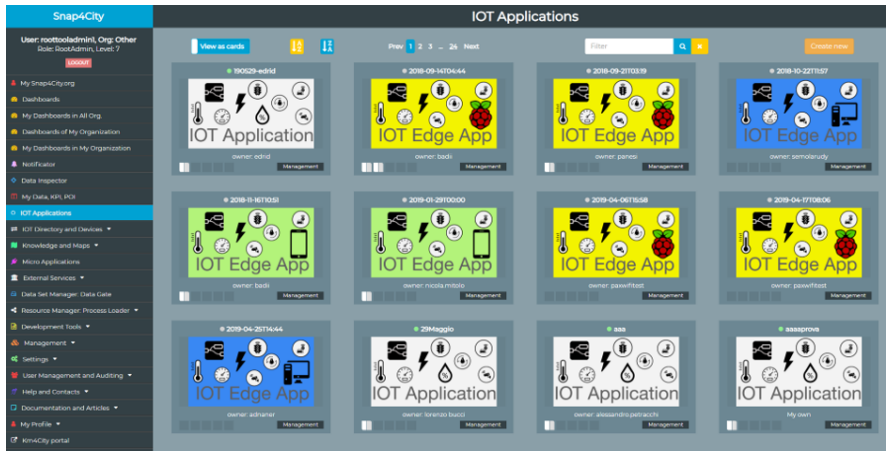


Figure 6.2: Snap4City: the IoT Applications manager as seen by a user

Both Libraries of Snap4City Nodes can be directly installed in any Node-RED tool of any operating system: Linux, Raspberry pi, Windows, etc. In addition, we have also developed an Android App executing Node.JS/Node-RED and our libraries to provide use of them on IoT Edge, also exploiting mobile device sensors on the above-mentioned operating systems. In [51], we have demonstrated how Snap4City approach may work for mobility and transport applications, where safety critical communications and solutions have to be set up, involving IoT networks with IoT Devices and IoT Edge, IoT Apps and Dashboards.

In this chapter, a deep view of the analysis and design of Snap4City MicroServices for smart city applications is presented. The analysis has been also supported by the evidence about how the MicroServices can be successfully used for IoT Applications implementation, so that they can be used by City Operators and Final Users.

6.5 Snap4City library of MicroServices of Smart City

In order to satisfy the smart city requirements reported and discussed in Section 6.3, in Snap4City a collection of more than 150 MicroServices has

been developed as Nodes for Node-RED programming environment. The Node-RED philosophy of visual programming allows the creation of event driven data flow applications where the exchanged messages are in JSON format. On the other hand, also periodic processes can be developed by scheduling one or more internal timers. This means that users can develop IoT Applications as Node-RED flows, exploiting both Push and Pull data protocols, with the same visual programming environment. In the context of smart cities, both protocols are needed, while IoT Applications have to be capable to create flows exploiting a large number of features which are typically not available in Node-RED open library, neither by installing a number of libraries from different providers. Moreover, the Snap4City MicroServices are at a level which can grant an easy development of IoT Applications for smart city even for non-expert users, as assessed in Section 6.6.

The most relevant families of nodes/MicroServices for smart city are listed below and they perform different kinds of activities, useful in the IoT App construction.

Access to Smart City Entities, that have different data models, and thus, different MicroServices may be required. Some Entities may have simple sets of Metadata for examples the ones describing the POI, e.g.: title, description, web, email, GPS location, images, opening time, etc.; others may have complex information, and/or also specific data types and structures, for example:

- Status of the first aid: number of people under observation for each colour in the triage, waiting time, etc.;
- Bus stops: all the bus lines including geometry, their planned time schedule, the real time delays, etc.
- Sensors with their values, measure units, types, healthiness criteria, etc.;
- Weather forecast associated with an area/region, which consists of a large set of values: temperature, humidity, pressure, wind, etc., for many different time slots in advance;
- Shape of cycling paths, gardens, parks, difficulties, restrictions, etc.;
- Parking areas, with the number of free spaces, predictions, typical daily trend of free spaces, costs, etc.;

- Events of: (i) entertainment, with their description, photo, start date, end date, etc.; (ii) from police officers on the street; (iii) Events of emergency, such as Civil protection early warning in CAP standard, etc.

In order to simplify this complexity, MicroServices like "Service Info" and/or "Service Info Dev" are provided for Final Users and Developers, respectively. In the IoT Application, a search/Discovery has to be performed as described in the next paragraph, otherwise the developer needs to know the so called ServiceURI, which is the unique identifier of all the city entities in the Km4City Smart City Knowledge Base. In the Snap4City development environment, the ServiceURIs can be recovered directly from the graphic configuration of the MicroService or searched by using the ServiceMap visual tool which provides a GUI for query. This means that IoT App programming is 100% visual, even if a single service is used for accessing a single element.

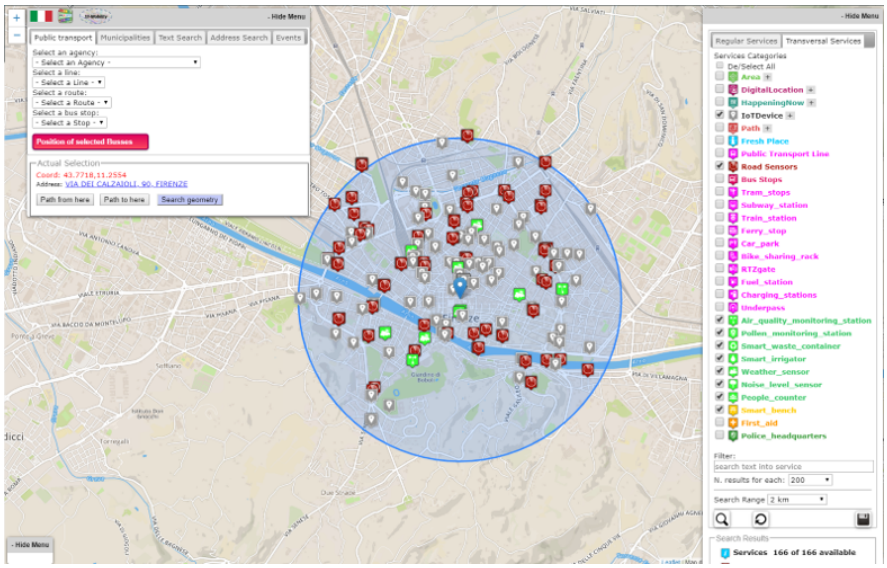


Figure 6.3: ServiceMap for ServiceURI identification

Search/Discovery of City Entities and their relationships. The search of city data has to allow users to discover data/device values by a

semantic search, using a composition of the available query types described in the following:

- Semantic classification. In Snap4City, all the POI and Services are classified with more than 20 classes (mobility, energy, banking, environment, cultural activities etc.), including a total of more than 520 subclasses (see for example, the menu of the ServiceMap reported in Figure 6.3);
- Geo spatial references: close to a point, max distance from a given point, along a path, into an area/polyline;
- Textual keywords substrings: for example, on the basis of the title and descriptions of city entities;
- Value Types: for example, all sensors measuring temperature, all bus paths, etc.;
- Historical time slot: for example, data values for the last 7 days;
- Prediction time slot: for example, data regarding predicted parking slots for the next 15 minute, 30 minutes, 1 hour;
- etc.

The results of this kind of searches can be a single element with its description as well as a JSON containing a list of entities. In this latter case, the list can be split in single messages using Node-RED Split node. Once the list of ServiceURIs is accessible, their detailed description can be obtained by using the above-mentioned "Service Info" node. The search facility for Final Users is provided via the Node interface, simply performed by employing a user interface for setting parameters. On the contrary, as to the nodes for Developers, search parameters can be also prepared and sent to the search Node in JSON. In both cases, the user does not need to know any query language (e.g., SQL, MySQL, or others), neither if the data are coming from a complex set of queries on SPARQL for RDF store, on ElasticSearch and Phoenix/HBASE. Therefore, the developer of IoT App does not need to know all the tiny details about the large variety of adopted storages. Snap4City provides more than 70 different nodes for searching different Smart City entities, providing results in different data types: POIs, time trends, values, events, schedule of busses, bus lines, recommendations,

addresses, routes, etc. This approach simplifies a lot the creation of Smart City Applications.

Discovering and Exploiting IoT Entities (sensors and actuators) should not be different than discovering any smart city entity. In Snap4City IoT Devices and data values can be accessed, searched and discovered by the above presented MicroServices/nodes. On the other hand, when the user/developer would like to create a data driven IoT Application exploiting specific IoT Devices, some specific MicroServices can be used so as to discover the desired devices regardless of the IoT Broker, IoT protocol or IoT Device. To this end, the IoT Directory MicroService exploits the services of the Km4City Knowledge Base for searching, managing and discovering all the available IoT sensors/actuators [49]. In Snap4City, developers can register on IoT Directory IoT Devices and IoT Brokers supporting a large number of different protocols and authentication models [198], [59]. The IoT Directory automatically registers new Devices into the Knowledge Base, and each new IoT element receives a ServiceURI, thus becoming a City Entity and POI [198]. A number of IoT Brokers are provided by Snap4City, while others are managed by third parties. For example, to provide answer to queries such as: "give me all temperature sensors close to my house" (regardless of the data providers, protocol, source, etc.).

Creation of advanced Graphic User Interface including graphics widgets which are: Dashboards, Virtual Devices to act on the IoT Applications and IoT Devices, advanced tools, etc. This means to give developers the possibility to design the IoT App user interface of the IoT App. The user interface has to show and also to get interactions from users to create messages for the IoT Network including the IoT App that may implement the logic of the user interface. The user interface is built by composing a number of graphic widgets in a connected Dashboard for presenting and collecting data. In Snap4City, the IoT Apps may be connected with Dashboards by means of widgets for:

- rendering data: single content, time trend, bars, histogram, map, pie, semaphore, dynamic signals, buttons, knobs, clock, bars, etc.

- collecting data as a switch, a knob or a keypad, which are interactive elements on Dashboards and are represented into IoT Applications as Input Nodes for the flow, able to provoke events into the IoT App according to the data driven approach.

- showing MicroApplications and External Services in a generic iframe widget.

In addition, other Dashboard Widgets may not have a counterpart into the IoT App and can be directly added to the Dashboard by using the Dashboard Builder editor [61]; which can:

- include Visual Analytics tools such as: External Services, MicroApplications, object Tracking, Origin Destination tools, heatmaps, traffic flows, maps of any kind, etc.
- visualize data from other sources: IoT Brokers/ Devices, data stores, API, etc.
- include Virtual IoT Devices that can send data directly to IoT Brokers, as any IoT Device but coming from the user interface.

Data Analytic, which would mean to provide MicroServices/nodes exposing a number of data analytic services to be exploited into the IoT Applications. In Snap4City, it is possible to: (a) exploit computing of data analytic processes already in place, (b) develop new data analytic processes, (c) calling External Services as rest call. In fact, Snap4City suite provides MicroServices implementing:

- Data Analytic for computing: routing, auto-ARIMA predictions, anomaly detection, descriptive statistics, machine learning predictions, heatmaps, providing social media data from Twitter Vigilance [72], etc.
- A generic Data Analytic MicroService where Data Analytic algorithms developed in R-Studio can be put in execution, according to specific guidelines, as shown in Figure 6.4;
- A REST call invoking External Services. For example, to get access to Twitter Vigilance API, The Weather Channel API, The Things Network API, etc.

Save and Retrieve Personal Data that could be time series for motion tracking, values of personal devices, clicks on mobiles Applications, POI, shapes, KPI (key performance index), Keys to access IoT Devices services, etc. The possibility to save and retrieve data from a safe storage (with the possibility of assigning delegations according to GDPR) enables a large

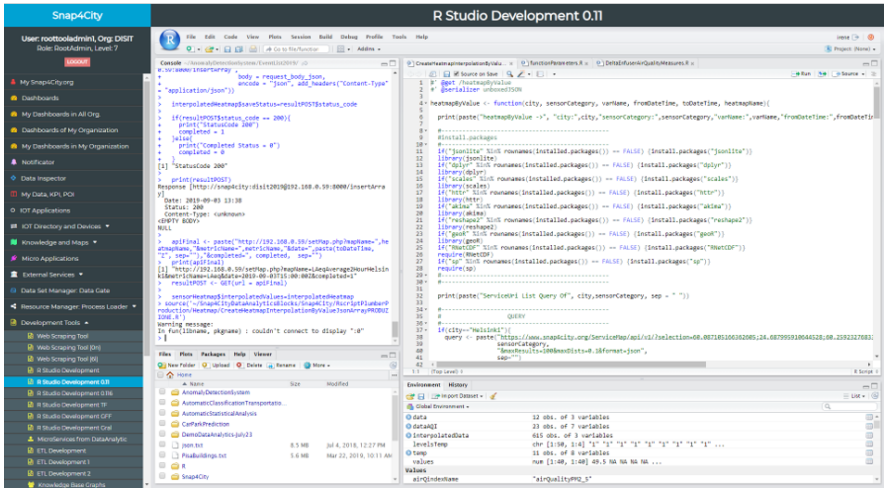


Figure 6.4: Developing Data Analytic MicroServices via R Studio to exploit them from IoT Apps in Node-RED

variety of smart scenarios for the final users and operators. For example, the production of IoT applications saving personal data from personal health devices (e.g., monitoring glucose), from home status, from the location of mobile phones and thus belonging to a specific person. In addition, the Snap4City platform automatically collects all the personal data gathered from mobile Apps accessed on the Snap4City login (provided that the user authorizes their collection with a signed consent, according to GDPR).

Save, retrieve, and publish Data Sets, as those managed in the open data portals. Most public administrations are used to publish their data sets in open data, and also share their data via federated networks, see for example the harvesting mechanisms of CKAN [9]. Having the possibility of creating data sets from the IoT Applications flow means that city operators can automatize their ingestion/update processes and their production/publication.

In the Snap4City suite of MicroServices, there is also a number of tools for managing DISCES backoffice scheduler of processes, as well as for saving LOGS regarding the data access and flows. The former are accessible only by administrators, while the latter can be useful for both developers and

administrators. Moreover, a number of so-called GEO-utilities have been identified and implemented in Snap4City such as MicroServices/nodes to:

- Calculate the distance from two GPS locations.
- Verify, once a GPS point and a closed shape/area are given, if the point is inside the area or not, so as for instance to verify the match with administrative areas, to understand whether your dog is in the garden or the monitored bear in the forest, etc.
- Get the most probable value of a variable represented in a Heatmap from any GPS point irrespective of the presence or absence of sensor in that point. This feature is very useful to decide among different routes: taking the quietest one, taking the less polluted one, taking the less crowded by the traffic, taking the busiest road to meeting people, etc.
- Get the closest civic number and street from a GPS point or vice-versa. This is very useful for geo reversing.
- Get the closest road map segment (typically called node in Open Street Map speaking) from a GPS point or vice-versa. It can be useful for routing and computing precise distances. This is very useful for geo reversing.

This means that the IoT Applications developers do not need to solve the so-called direct or inverse geo-referencing problems, since they are provided by default.

A number of Node-RED nodes useful in smart city contexts are also available: they have not been implemented by us, as they are globally provided by third parties such as: interaction with Facebook social media, SMS sending, email sending, Telegram, etc.

6.6 Example of applications

In this Section, three examples are presented with the aim of depicting the capabilities and the expressiveness of both the IoT Apps and the Snap4City development environment. They are IoT Applications for: (i) emergency management, (ii) routing on the basis of environmental data, (iii) dynamic management of PAXCounters IoT devices for monitoring people flow at museum and events.

6.6.1 Alerting about critical events involving people in a specific area

In this Section, we are presenting a scenario where a public operator (Road Operator) on the field, like a policeman or a public transport driver, notifies to a control room operator (City Operator) a critical event in the city. The notification includes the reporting in real time of the event position, the number of involved people and the seriousness of the event.

The City Operator would like to: monitor events vs services in the city and receive critical event notifications from Road Operators and assess contextual conditions and services status. The control room receives the notifications and can explore the status of the services in the city, evaluate the gravity of the situation and, therefore, take the correct decisions to cope with the event. Especially if many events may arrive from the same area, a critical condition may be early detected and may be an Early Warning propagated to other cops. The Road Operator would like to monitor the status of traffic, parking, environmental variables, speed limits, services and send critical event notifications via coded descriptions. Figure 6.5 better explains this scenario.

Figure 6.6 provides an overview of the City Operator's Dashboard named "The City Operator" can see the City Map (1) and select one or more services on the selector on the left (2) to see them on map. By a click on the elements forming part of the list on the left (2), the services belonging to the selected category are displayed (or hidden) from the map. Multiple categories can be selected at the same time to have a broader overview of what is the situation like in the area where the Road Operator is. By clicking a POI on the map (1), a pop up is shown representing the real time data of the selected POI (if available). The data shown in the pop up can be also represented as real time values in a dedicated widget (3), as well as time trend values in another specific widget (4).

When the Road Operator sends the message related to a critical event, the City Operator receives the notification in the widget (5) that lists all the notifications received. The minimap (6) is automatically centred on the coordinates sent by the Road Operator.

The creation of the City Operator dashboard has been performed by using a wizard guiding the user in the choice of the variables he wants to monitor and insert into the dashboard, as well as the widgets where to display the values produced by such variables [61]. Please note that Widget (5) is a data

Figure 6.5: Connection among City Operator dashboard in the control room and the one used by on Road Operators

driven visualization from data arriving from IoT Orion Broker in NGSI.

Please note that the two Dashboards shown in Figure 6.5 have been created in two different manners, starting from the:

- Dashboard Wizard which permits, through simple guided steps, to select the data sources, then decide which widgets to be used to visualize them , finally arrive to the dashboard creation and of the selected widgets in few clicks.
- IoT Application editor (Node-RED), selecting Dashboard MicroServices, and connecting real time data, to see them automatically appearing in the dashboard as Widgets.

Please note that both approaches are valid and can be mixed up.

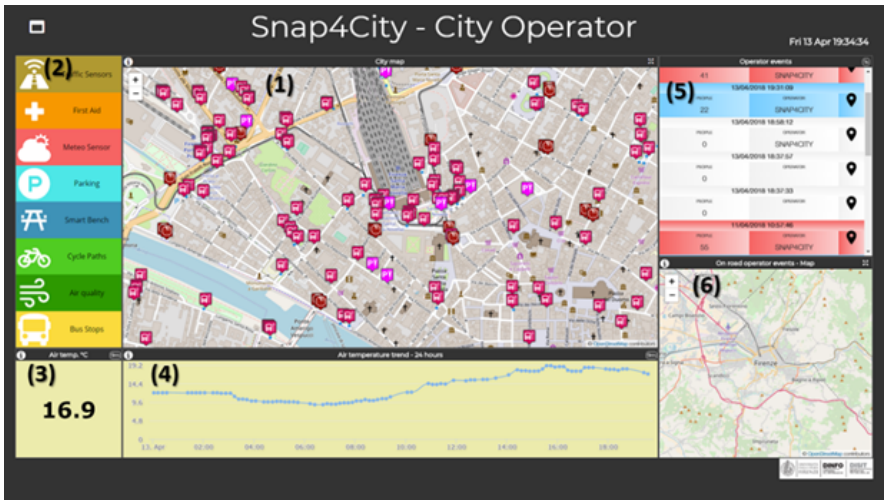


Figure 6.6: City Operator Dashboard

When the Dashboard Wizard is used, as in Figure 6.7, the user is requested to choose one of the available templates, and then the structure which the new dashboard has to be based on. If the user chooses a particular template, it is also possible to add more widgets in a second moment manually, or by calling again the wizard within the newly created dashboard. In the Wizard, the user can choose data sources to use inside the dashboard. The types of data sources are various: Complex Event, IoTApp, External Service, Heatmap, MicroApplication, My Personal Data, MyData, KPI, MyKPI, POI, MyPOI, Sensor, Actuator, Special Widget, WFS. Multiple data sources can be selected to be displayed inside the same dashboard. The data generated by the selected sources can be very different from one another, and the user must choose how to represent them within the dashboard. In the Wizard, for each selected data the user may choose among the possible widgets which can represent and display that specific data type. Therefore, the possible graphic widgets are dynamically shown. To make the choice easier for the user, when he has selected a certain data source, widgets are automatically filtered, and vice versa: the user could also firstly select the widget model for the dashboard, and then the wizard will filter the data sources showing only those that may be visualized on the selected

widget type.

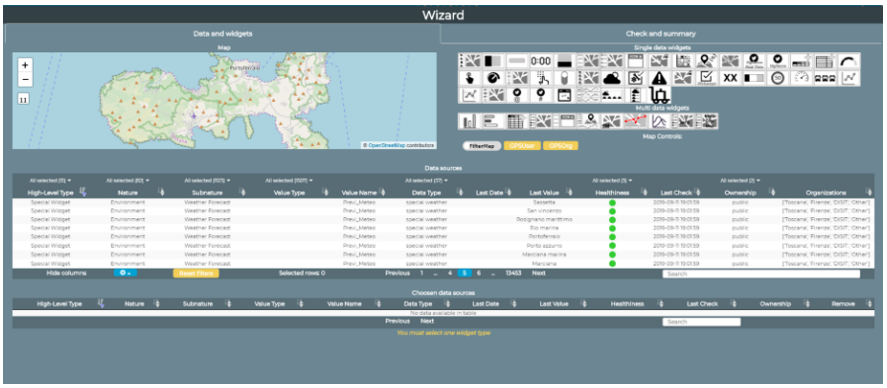


Figure 6.7: Dashboard Wizard

Once the new dashboard is created, if the user wants to add more widgets, the dashboard can be edited through the wizard again and again. The adopted approach allows to easily create interactive dashboards without using any particular programming skills, but relying only on the help of the wizard. The creation of the Road Operator's dashboard has been performed by using the IoT Application editor, which is the second approach listed above. The Dashboard Wizard is not used to create this dashboard, actually it has been created in automatic mode from a Node-Red stream. To allow the IoT App flows to interact with the Dashboards, we have to create in the flow Nodes/Blocks representing Dashboard Widgets. Widgets in the flows can be INPUT widgets (providing data from the Dashboard to the IoT App) as well as OUTPUT widgets (graphically representing some data on the Dashboard). These elements can be regarded as Virtual IoT devices as well, respectively Virtual Sensors and Virtual Actuators.

The Snap4City Dashboard nodes allow the creation of 5 types of widgets on the dashboards, to represent the different metrics chosen by the user/developer/city operator, among a range of many available ones; these widgets are: Gauge Chart, Speedometer, Time Trend, Single Content and Web Content. The first 3 types accept numerical values as input: in the first two types, the last sent value of the represented metric is displayed, and in the third type the metric history (as a time graph of all the metric values sent

over time) is also shown. The Single Content can receive data in any text format and can also display HTML. The WebContent allows you to send a url and to display the resource that is located at that url within the widget.

The nodes creating Actuators on the Dashboards are of 5 different types: Impulse Button, Switch, Numeric Keyboard, Knob and Geolocation. The Impulse Button sends an "On" signal when pressed and an "Off" signal when released. The Switch allows you to set the button to "On" until its status is changed. The Numeric Keyboard allows you to send any numeric data from the dashboard to the stream showing the last value sent in the widget. The Knob allows you to dynamically change the sent value, depending on how the knob is turned. The Geolocation sends the position retrieved from the browser where the dashboard is shown.

In Figure 6.8 (a), the relationships between the User Interface in the hands of the Road Operator and the corresponding IoT Application are shown. The Node-RED flow is divided into three parts that are described below. In 6.8 (b), GPS Position: the buttons receive the GPS Position that has been set as "Helsinki", "Florence", "Antwerp", and "My position" in this example. By clicking the GPS position button, the map on the top sets the position accordingly. The flow is making the request to the Smart City API, and the map is shown in the right place with the list of services and PINs on map. In 6.8 (c), People Number: The numeric keypad Widget allows the Road Operator to insert the number of people involved in the critical event. The user enters the number and press the "Confirm" button. The confirmed number is shown in the "Last confirmed" box on the top-right of the numpad. In the IoT App, the number is stored in a temporary memory.

In Figure 6.9, Alert Colour: the coloured buttons allow to set the colour code of the emergency. If the user has set correctly the GPS position and has inserted the number of people involved, by pressing the colour code he can also send the message to the City Operator, by posting it on an IoT Virtual Device of the Dashboard.

In detail, when the Road Operator presses the colour button indicating the priority of the emergency, a JSON is created containing information about all the previously described aspects such as: the number of people, the coordinates and the colour of the emergency. This JSON is sent as a value of an entity created on a Fi-Ware Orion Broker through a node which is enhanced with the ability to add authentication keys, if compared to the base node provided in Node-Red. This ensures that writing such an entity

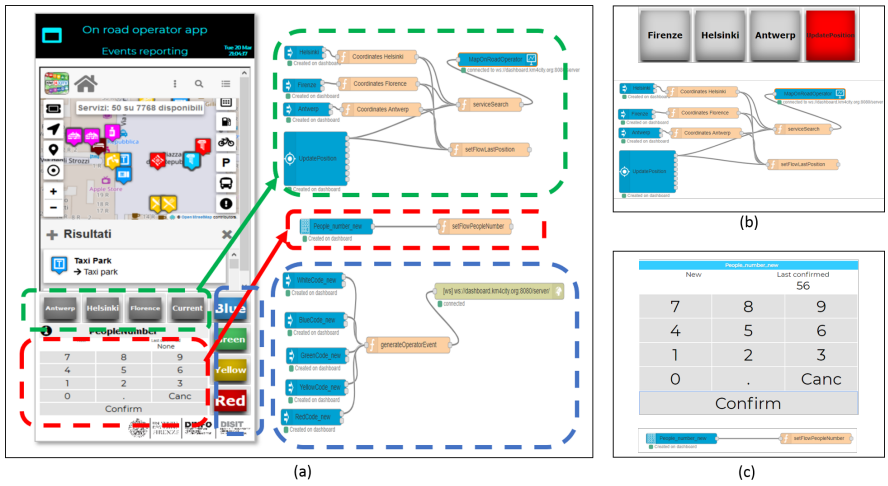


Figure 6.8: On Road Operator user interface and IoT Application managing the logic. Overview and first details

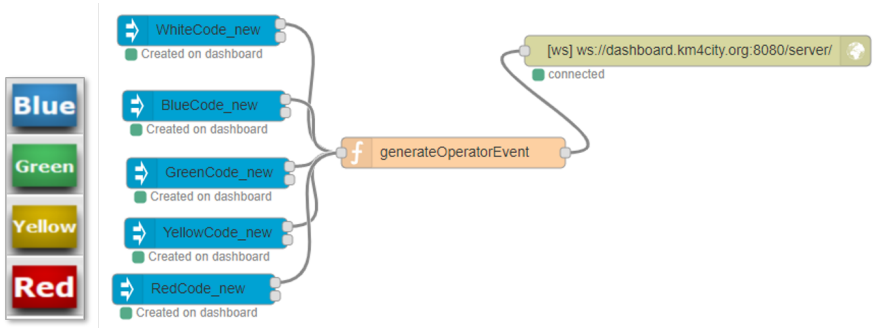


Figure 6.9: On Road Operator user interface, last part

can occur only if the IoT App, which is sending the data, belongs to the entity owner.

The entity registered on the IoT Orion Broker can be also chosen in the wizard, which allows you to create the widget that you see in the City Operator's dashboard at the top right (5). This widget adapts the displayed

value dynamically when a new JSON arrives, thus changing the state of the entity with no need on the City Operator's side to update anything on the dashboard in order to comply with the Event Driven messages. The flexibility of the dashboards allows to create mixed dashboards containing widgets created with the wizard and widgets automatically created by the Node-Red streams. If the dashboard is created automatically by the creation of widgets from a NodeRed stream, you can open the wizard and add additional widgets through the different modalities explained above. The messages exchanged between the Node-Red streams and the IoT Orion Broker use the NGSI protocol, which is the same protocol used when values are received from the widgets on the dashboards linked to the real IoT Devices of the IoT Orion Broker. The exchange of messages happening among the nodes dedicated to the dashboards (whether they are actuators or viewers), is made via Web Socket Secure protected by an Access Token depending on the owner of the same IoT Application.

The additional nodes created to enhance the functionality of Node-Red in Smart City can also be installed on local users' devices (such as Raspberry etc..). In some cases it is necessary to request and obtain an account on the Snap4city platform before exploiting such enhanced functionality. Without this account you cannot get the Access Token ensuring the security of communications among nodes and other components of the Snap4city platform [57].

6.6.2 Check which route is less polluted

The above examples have shown how it is possible to use Snap4City MicroServices for the development of systems useful for the security and control of emergencies in the public sector. The suite of MicroServices used in the above exercises have been mainly composed of dashboard elements. IoT App with dashboard can also be used by citizens within private applications. In the example developed in this Section, MicroServices retrieve information from the Smart City storage and info to create a dashboard which provides the user with pieces of information on which is the less polluted path at a precise moment to go jogging. If predictive data are available, it can work on predictions.

In Figure 6.10, the dashboard presents two possible paths to go jogging; the values of O3, PM 2.5 and PM 10 of the two paths are displayed in the centre.

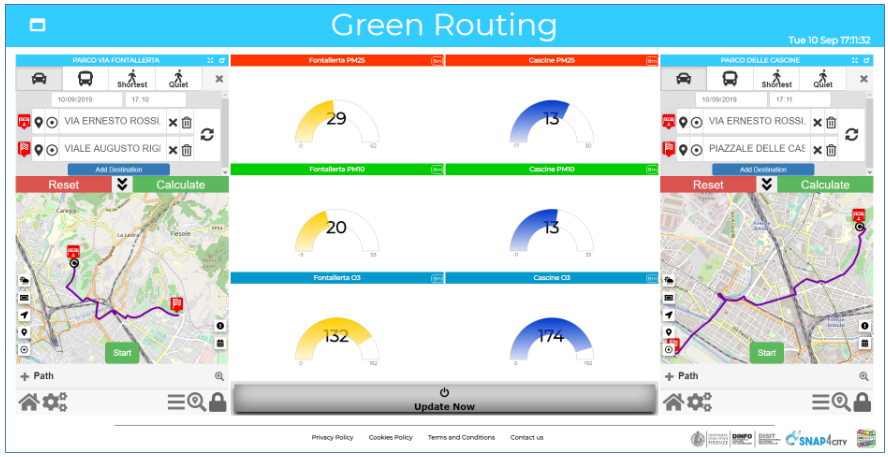


Figure 6.10: Personal Dashboard for deciding the less polluted path for jogging

The flow realizing the logic behind the dashboard is shown in Figure 6.11. This dashboard has been created automatically, when the Node-Red flow has been deployed the first time. In the flow, the outermost blue nodes on the right correspond to the two maps with the routes, the six central blue nodes on the right are related to the Gauge Charts reporting the values of the pollutants, and the only blue node on the left corresponds to the "Update" button that allows to recalculate the data when pressed. These nodes automatically create the widgets on the dashboard, if the dashboard exists; otherwise, they also create the dashboard, without any required intervention by the user, who has to edit only the name of the dashboard and of the Widgets, when configuring the nodes.

The orange nodes on the left, called "PARCO DELLE CASCINE" and "PARCO VIA FONTALLERTA", allow the user to calculate the shortest path between two geo-points. With a simple visual configuration, as shown in Figure 6.12, you can choose a starting point, an arrival point, the time you want to begin the trip and the transport means: the node will produce in output a JSON containing all the information about the calculated route. As you can see, the complexity of the route calculation is transparent to the user who has the result of his request in a simple and intuitive way.

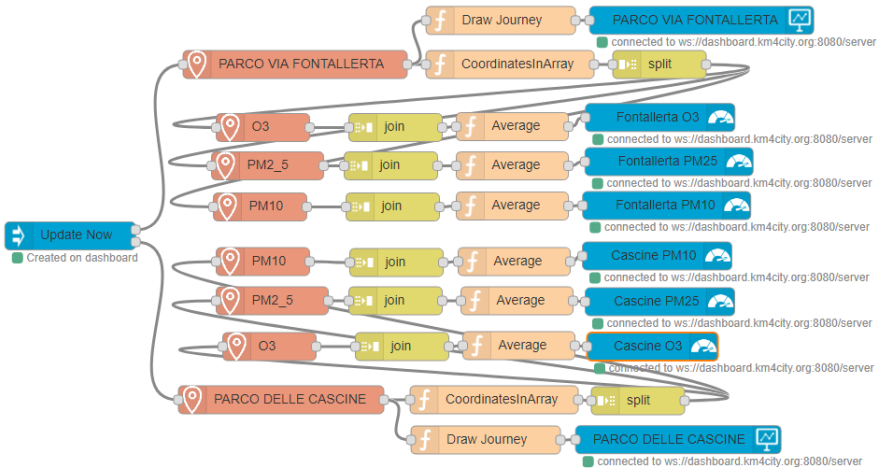


Figure 6.11: IoT Application with the logic of Dashboard of Figure 6.10

In this Node-Red flow, unlike the previous example, a minimum of programming ability is required: the JSON that is returned from the previous nodes also contains the complete path calculated by the system, and this must be divided into its main points exploiting the function nodes made available by Node-Red (where the user can interact with the variables by writing code in JavaScript).

The other six orange nodes on the left, internal to those just described and which have the names of the pollutants, allow you to get punctually the value of that substance in a given geographical point.

In this case too, the complexity lying behind this value is hidden to the user, who manages to have, in just few steps, that value in his hands, thus being able to use it as he pleases. The complexity behind the whole described scenario, which is transparent to the user, implies the retrieval of all the sensors data available in the area under consideration, the calculation of a heatmap (furthermore, interpolating values in the points where sensors are not located), its storage on a geo-server, and finally the recovery of timely data available from the same geo-server with the creation of APIs dedicated to this particular service. The calculation of the heatmap must be done periodically and, in turn, this periodicity has been done with a special Node-Red flow executing (on the pre-determined frequencies or scheduled time intervals) the R script in charge of calculating the heatmap. Please note that the sensors are typically located in some specific points, which may be very distant from the computed routing. A specific service in the smart city back office is dedicated to compute the heatmap and to provide a MicroServices which allows to get the value of pollutant, as well as of temperature, humidity, etc. in any point of the area.

In the flow depicted in Figure 6.11, the values of the polluting elements are calculated for the main points of the routes returned by the first two nodes, and the average is made so as to have an indicator of pollution for that route. The user who has created a dashboard made in this way can press the update button, before going for jogging and he will know where to go running to breathe cleaner air. The automated update can be programmed as well.

6.6.3 Controlling personal mobile PAX Counter

In the first example, we have seen how to create two interoperable dashboards through an IoT App, in order to ensure support for emergencies within a city. We have also demonstrated how this allows a smart interaction with Road Operators, who are located directly on the emergency scene, as well as with City Operators who are located inside the City Control Room. In the second example, we have shown how to create a dashboard retrieving data from city sensors to get real-time information on the state of air quality. We have also stressed how this can be useful for decision support, for instance when

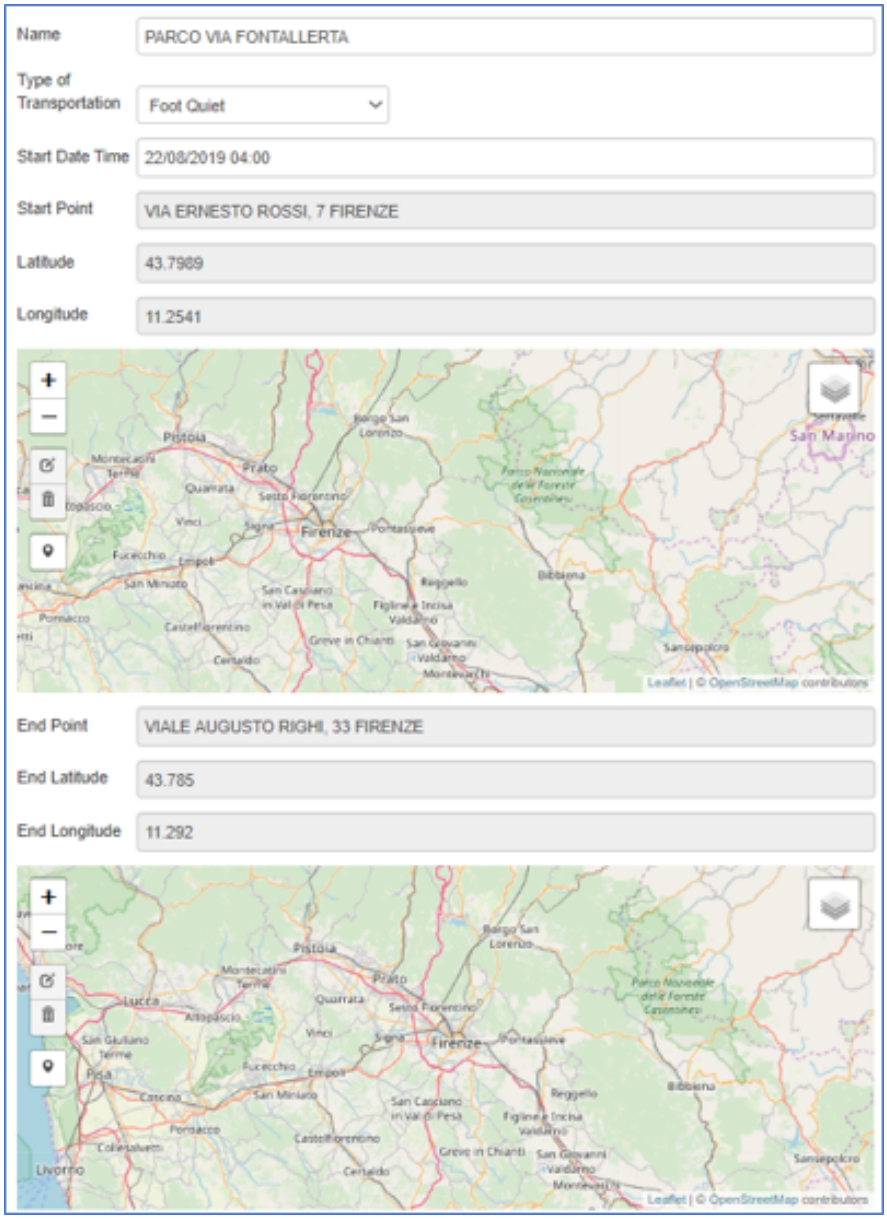


Figure 6.12: Setting for routing of IoT App in Figure 6.11 and of Dashboard in Figure 6.10

making choices based on pieces of information offered by the system and able to improve the user's health by suggesting the most virtuous behaviour. In both cases, everything was done through MicroServices and the user did not need any additional device to obtain the required info. All the temporary data for the processes has not been permanently stored.

This example depicts the interaction with IoT Devices counting people by using Wi-Fi and Bluetooth sniffing in its vicinity (according to GDPR [12]). In Figure 6.13, a mobile PAX Counter is shown, based on ESP32, sending messages to Snap4City via LoraWAN. Other simpler versions can be located on fixed places, and may also exploit WiFi to send the measures obtained also from different, additional sensors. In the current example, they have been used to monitor a Museum in Antwerp and many other locations. A total of 22 devices have been installed. The measured values are sent to LoraWan operator The Things Network, which does not provide a system for data analysis, except for actually only offering a way to receive and manage devices.

Figure 6.13: A mobile PAX counter based on ESP32

Museum Case In the literature, there are several technologies for tracking movements into the museum and understanding the inside flows [215]. On the other hand, their costs are very high and for most applications it is not needed to have the internal traces. In the presented IoT Application and Museum case with free entrance, it has been of great interest the monitoring

the flows of entering and exiting of visitors, collecting also the history of such data and not only the real-time values with low cost solutions. In fact, the monitoring via PAX Counter of 10 doors would cost less than 3000 euro. To this end, data have to be saved permanently into time series data store, for example into MyKPI, MyPOI storage via MicroServices. The Snap4City approach allows the IoT Developers to create and retrieve data entities trends directly in Dashboard and for Data Analytics, as data Shadow. According to this solution, an IoT App has been created for PaxCounter to receive in Event Driven mode the new data coming from The Things Network (as IoT Gateway), according to the Publish Subscribe model of MQTT; then, such data are saved as MyKPI via MicroServices. Consequently, Dashboard Wizard automatically sees the data entity on the MyKPI and allows us to create automatically Widgets showing their trends, real-time values, etc. Actually, the Snap4City back end provides a suite of scheduled scripts specifically designed to populate automatically and programmatically the Dashboard Wizard with data coming from all the different data sources (Sensors, IoT App, External Services, Heatmaps, Personal Data, KPI, MyKPI, POI, MyPOI etc.). Figure 6.14 shows the Dashboard presenting the time trends of the entrances and exits of the museum and all its derived data. Such derived values of the initial real-time data can be computed in real-time, saved on personal storage and shown on dashboard via the IoT Application exploiting Snap4City MicroServices. At the same time, while sums are being calculated, the difference between people entering the museum and those leaving is also performed, in order to obtain the number of people inside the museum in another entity.

Through the Dashboard widget and IoT App, it is possible to send commands to the Pax Counter, via the IoT Gateway, to get the status of the device, as well as to change settings. A command that can be useful when having a Pax Counter installed in the museum can be changing the mode of counting people from cyclic to cumulative. In the cyclic mode, every 8 minutes the count is reset and the anonymous data of the devices counted by the Pax Counter are deleted: therefore, in the following 8 minutes, if a person stands near the Pax Counter, that person is counted again. In cumulative mode, the count is not reset but is increased with new people who arriving near the Pax Counter; therefore, duplicate counts of the same person are avoided until the device is in cumulative mode.

Mobile PaxCounter for monitoring events The mobile PAXCounter

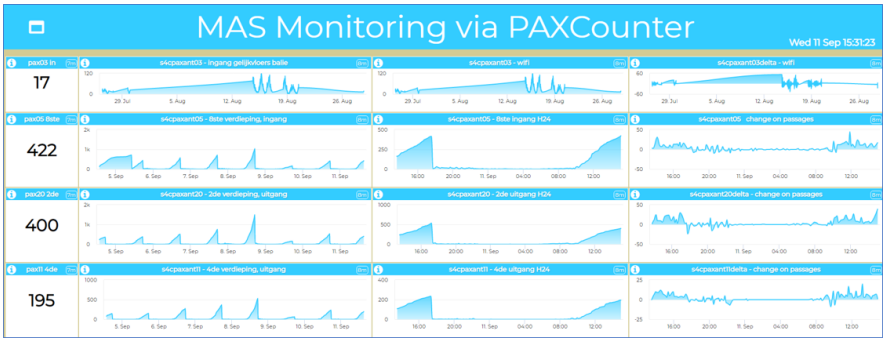


Figure 6.14: Monitoring MAS Museum people flows

presented in Figure 6.13 has been used for monitoring sport and entertainment events. Each new measured count is associated with GPS coordinates and sent to the platform, thus resulting in trajectories with associated values as depicted in Figure 6.15. This is done exploiting the Snap4City Tracker Widget.

In this case as already seen above, it could be very important to change modality, allowing to switch from Cyclic to Cumulative counting. To this end, in order to program the change of modality, an IoT Application with a dedicated Dashboard user interface has been realized, as reported in Figure 6.16.

In this case, the IoT Application is more complex since it requires to (i) collect data; (ii) program the IoT devices; (iii) interact with the user to collect the data for programming purposes; (iv) monitor the status of the IoT device to verify if the setting has been successfully applied; (v) report the status of the IoT device to the user. All these functionalities have been implemented in a single IoT App, as reported in Figure 6.17.

Thus, also a custom widget has been developed to allow the definition of both begin and end instants of the programmed period of cumulative mode. The activation of the programmed interval is performed by pressing the “Activate” button that saves the times within the IoT App. The IoT App performs change of modality at the due time, since the IoT Devices are not programmable. In addition, 3 Single Content widgets are instantiated to display the activated period of cumulative mode, the status of IoT App for setting data to the device, and the state of the IoT Device. In

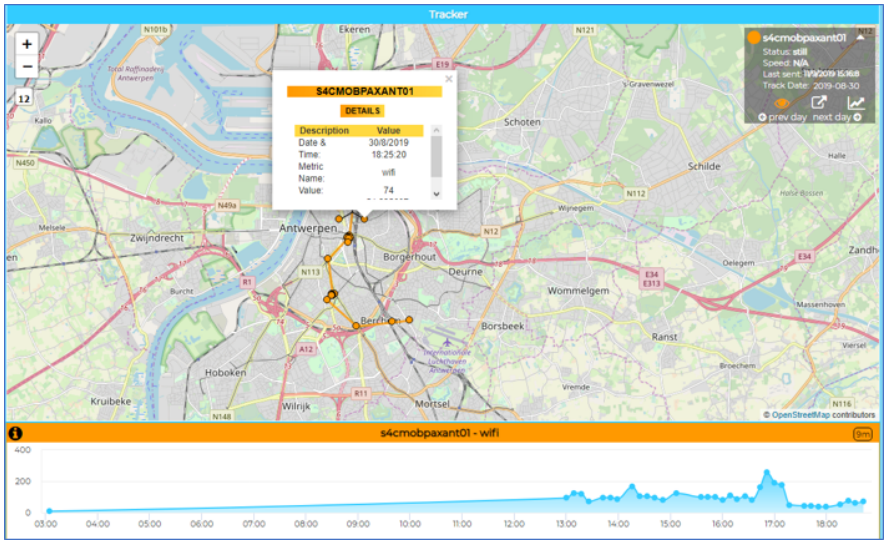


Figure 6.15: Tracking PAX Counters, measures over time and space

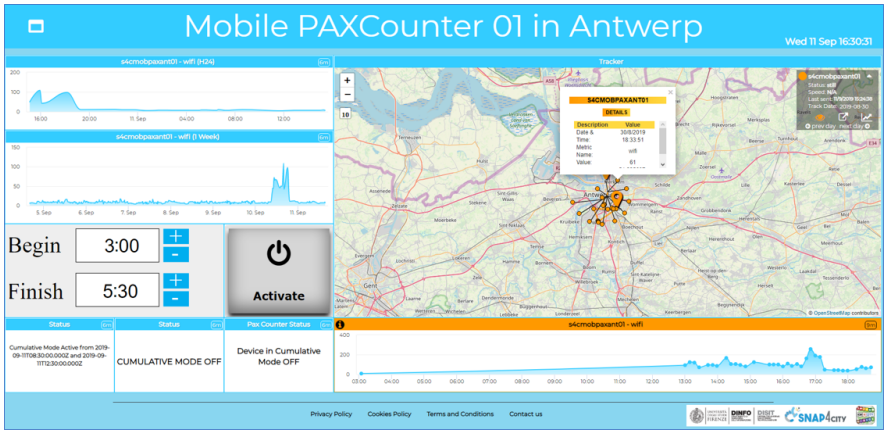


Figure 6.16: Management of the Mobile PAXCounter with the possibility of programming change of modality

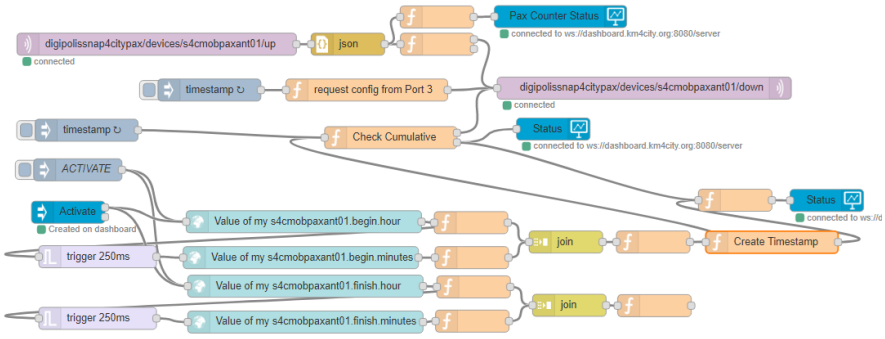


Figure 6.17: IoT App controlling mobile PAXCounter

Snap4City, Custom widgets can be developed by creating them in SVG and implementing a specific process to connect them to Snap4City MyKPI.

Once the custom widget has been created, the IoT App is created realizing the control logic covering all the functionalities listed above by exploiting the MicroServices for: dashboard widget, personal storage exploitation, IoT Gateway connection, etc..

From the point of view of the life cycle, the above presented IoT Applications can be saved into the Resources Manager, thus allowing other users to download and install in their own resource area, and use them with their own devices.

6.7 Development Life Cycle of Snap4City Applications

The design and development of IoT Cyber Physical System/Solutions, CPS, is much more complex than regular software developments approaches for SW life cycle. An aspect that they share is the adoption of strongly dynamic microcycles. In [171], a review of life cycles for IoT has been presented. The life cycle proposed is a simplification with respect to the actual needs, since it does not take into account: data analytics, personal/private data, complex dashboards. In most cases, when IoT development life cycles are discussed the focus goes on devices development, finalization, test, deploy, connection,

etc., while as in [90] the complexity of the life cycle is much wider.

The Life Cycle for the IoT Applications/Solutions in complex domains as Smart City involves a number of specialized tools such as those for Maps, Dashboards, etc., that need to be well placed in the life cycle. Therefore, in this Section, the development life cycle for Snap4City applications is presented, see Figure 6.18. In the represented life cycle only the software aspects are address.

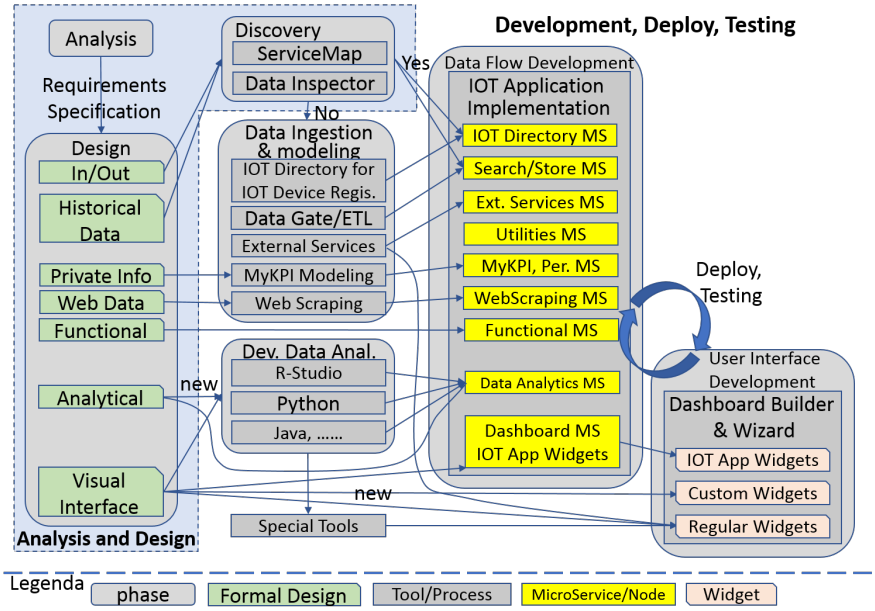


Figure 6.18: Snap4City Development Life Cycle for Cyber Physical Solutions, where we distinguish among phases, formal documents, tools and processes, MicroService/Nodes and Widgets of the Dashboards.

In the context of Snap4City IoT Smart City development, the life cycle should start with the Analysis and Design to produce requirements and specification of the application. This information is collected as details regarding:

- Data analysis have to be identified on the basis of the scenarios goal. And in particular: real time inputs and outputs that could be used;

Historical data that could be exploited for showing them or for Data Analytics; data to be scraped from Web pages according to the terms of use; private information of the users or entities involved according to GDPR guidelines. This process of data analysis also may imply the Discovery of Data on the available knowledge base via Service Map or Data Inspector. If the needed data are not available specific ingestion process will be developed.

- Functional transformations of data and commands, which can be needed on the basis of the data format accessible. This activity could also create business logic of the IoT Application taking into account historical data, inputs, MyKPI, scraped data and Data Analytics to produce outputs on other devices and/or dashboards.

- Analytical identified which are the data analytic to be developed for the application, if needed. For example, for a Smart Parking a free parking slot prediction will be needed and this is evident since the analysis. The Data Analytic will have to be developed from scratch or customized exploiting some ready to use MicroServices.

- Visual Interface. In most cases, the graphic interface is part of the problem. A good visual representation may do the difference from a tedious view a nice user centric view. This part includes the design of the Visual Interface in terms of Widget and Dashboard. The single elements may be presenting data coming from the data store/shadow, may be directly produced from/to the IoT Application data flow in real time, or may need to be developed as Custom Widgets.

Once the Analysis and Design has been completed, a global view of the Smart Application has been sketched but need to be implemented. Thus, the different aspects and phases may be addressed by different tools and for large factory also by different teams addressing:

- Data Ingestion and Modeling. In this case, different tools can be used:
  - IoT Directory for registering new IoT Devices, IoT Brokers, etc. In some cases, the definition of the IoT Device Model is needed to pass at the IoT Device Registration in Bulk.
  - DataGate/CKAN: for the ingestion of Open Data with geo information as POI.

  - ETL development, a process of ingestion of some file or data that has to be taken in PULL for example, from WS, FTP, HTTP, etc. They could be even developed into the IoT Application tool exploiting MicroServices.
  - External Services: the direct identification of external API that could be exploited into the IoT Application.
  - MyKPI, the modeling and registration of the KPI and data needed to have permanent and personal data into the IoT Application.

- Development of Data Analytic. This process is typically addressed by Data Analysts skilled on data science, accessing to historical and real time data and creating specific algorithms such as: prediction, anomaly detection, interpolation, clustering, etc., exploiting libraries and tools of operative research, statistic, machine learning; using languages and development environment provided by Snap4City for online development such as RStudio, Python, Java, etc. The Data Analytic may produce:

  - MicroServices that can be reused in a range of IoT Applications. They produce data on Storage to be visualized on Dashboards by regular widget such as Heatmaps.
  - Special Tools that may produce view as External Services on Widgets such as Origin Destination Matrices. Also these Special Tools may produce data into the Big Data Storage,

- Data Flow Development (that is the development of the IoT Applications. The data flow development is typically focused on producing outputs on storage, MyKPI and/or presenting resulting data on some Widget Dashboard. This IoT Application development exploits the Snap4City library of MicroServices/Nodes plus those that may be newly created for the specific solution. The development of some IoT Applications has been presented in the previous Section. The IoT Application implements the business logic of the solution and can be deployed (on cloud/container or on IoT Edge). The Deploy can be performed in a sandbox for testing. When the IoT Applications includes the usage of Dashboard Widgets they are created on Dashboard at the moment of the Deploy of the IoT App.

- User Interface Development (that consists in the development of the Dashboard(s)). The Snap4City Dashboards can be composed by several kind of Widgets mainly classified as:
  - IoT App Widget directly produced or that have a counterpart representation into the IoT Application (also called virtual sensors, virtual actuators). These widgets are used for managing event driven graphic elements as inputs and outputs of the IoT Application such as: buttons, messages, switch, gauge, data view, trend, keypad, semaphore status, etc.
  - Custom Widgets which can be produced by using SVG tools and may implement synoptic as specific interaction tools with animations, etc.
  - Regular Widgets taken from the large collection of data viewer of Snap4City Dashboard Builder.

The final result consists of a number of: data ingestion processes, data analytic processes and MicroServices, IoT Applications, and Dashboards connected each other. The above presented Life Cycle in Figure 6.18 distinguishes among phases, formal documents, tools and processes, MicroService/Nodes and Widgets of the Dashboards.

The process of the Development, deploy, testing, validation, acceptance testing, is typically iterative, to arrive at the final deploy in production with an acceptance testing. Then, after that the solution passes to maintenance, and has to be continuously monitored to detect eventual problems, that may spring from: missing data, change of data format, exceptions in the data analytic or in the data flow, etc. All these sources of fault should be managed with a correct design and implementation with error management and fault detection. The best solutions, try to produce a resulting services any way, for example in the lack of data for making predictions via machine learning provide the average values, etc.

6.8 Assessment and experimental results

In literature, several Smart City experimentation testbeds have been presented, expressing the common requirement of involving citizens, ICT operators and local administrations. This aspect implies the organization and

deployment of experimentation facilities such as crowdsensing campaigns, living labs and tools to monitor and enhance the citizens engagement [78]. In order to assess the effectiveness of Snap4City IoT Application Development Environment, a training course has been organized with Smart City operators, ICT city officials, City Council Operators (they have never used Node-RED before). During the course, we assigned a number of exercises (`https://www.snap4city.org/drupal/node/501`) and recorded all the activities. Before playing with exercises, a specific but short training has been performed on Node-RED programming and Snap4City MicroServices. Then, exercises have been presented, and a fixed time slot has been assigned to learners to produce the IoT App. In both cases, the IoT also included the development of a small Dashboard. At the end of the exercise, we closed the sections, asked the users to provide some answers through a written questionnaire, while the IoT App and dashboards have been saved on the platform closed for assessment. Thus, the assessment has taken into account both results, as described below:

- The answer in paper about the exercise, where users in the audience should mark the needed Node-RED smart city blocks required to solve the problem; the form included the list of NODES/MicroServices that can be used to compose the IoT App.
- The developed IoT Application into the platform, accessible for our review, with the possibility of testing to verify whether it was 100% complete or not.

Please note that, the assessment has considered the fact that results would not be unique, in the sense that each exercise could be solved in different manners.

The aim was to answer to the questions: "As to the training received from us, was it enough to allow users to develop an IoT App to solve the problem?", "how much faster is the solution with respect to other solutions?", "how has it been perceived?", "do they have already learnt the programming model?", etc. The audience included 24 skilled users, with less than 20% ICT developers. The following exercises have been proposed:

- Create an IoT App that reads a value from a Snap4City service (for example the parking lot seen in the previous demo, identified by the serviceUri:

`http://www.disit.org/km4city/resource/CarParkPieraccinMeyer`) and realize a flow which sends different messages (e.g.: "Almost Full Parking slots" or "Free Parking") on the dashboard, on the basis of a certain threshold.. Apart from these functionalities, the creation of a service which sends to you an email would be considered as a plus (time required: 15 minutes);

- Create an IoT Application / flow that: (i) reads a value from a list of Snap4city services (for example the car parks in the Florence City Area, as seen in previous demo), (ii) calculates the average of Free Parking Lots, (iii) sends the value on a dashboard with the four possible nodes seen in the demo (time required : 20 minutes).

The assessment has been carefully performed, since each exercise can be solved with multiple solutions. To assess the exercise on the paper questionnaire, we have only verified if test-takers have selected the right microservices. A score of 100% has been given if they selected all the right answers, according to the possible solutions, while a fraction of 100% has been assigned proportionally if some answers have been missed. In the assessment of the final IoT App we have verified if the App satisfies the requirements, and a vote has been given according to percentage of such satisfactory outcome. In some cases, we have positively considered a plus when the developer has added more features (for example the email sending) (see Table 6.1,).

Table 6.1: Results of the IoT App Development Assessment

	Exercise1	**Exercise2**
Number of user	24	24
Exercise Duration	15 min	20 min
Result on paper	**21**	**21**
Average score	71,42%	83,92%
Variance	302,8	470,9
Resulted IoT Apps	**20**	**16**
Average score	85,75%	83,42%
Variance	692,8	449,0
75% of requirements	80,00%	81,25%

As we can remark from the results summarized in Table 1, 21 users have performed Exercise 1 on paper. The average score on paper assessment has been 71,42% with variance of 302,8. This means that the mean value of scores assigned on the basis of the paper assessment has been 71,42% over 100%, which corresponds to the percentage of identified Node-RED blocks needed

to solve the problem. In addition, 20 of those 21 actually implemented an IoT App. Only one test-taker has presented the result on paper and he has not been interested in performing the exercise on real tool. Test-takers obtained an average score on real results in terms of requirements coverage equal to 85,75% and Variance of 692,8. In average, the developed applications covered 85,75% of the requirements. In addition, 80,00% of users realised the IoT App with at least 75% of requirements. That is, in synthesis, a good result confirming that the training has been enough to let them create the IoT App autonomously. The test-takers who solved the exercise in the IoT App editor have actually performed better in the VPL tool, rather than on paper. This also means that they could identify the blocks (the MicroServices), but it stressed also that the development environment has provided a concrete support in correctly identifying the nodes/MicroServices, by means of the over functionality that provide help description when the mouse passes on the icons.

The 24 participants have been also provided with additional questionnaires to assess their appreciation and effectiveness in a more general sense. In this case, we have used a Likert scale at 5 values:

- 91% have demonstrated to be satisfied of the training day, and among them 53,40% have been very satisfied
- 85% have ranked the IoT solution and Data Analysis very good and among them 42,17% have ranked excellent
- 100% have stated that solution would be useful for their work, 78,13% have declared that it would be very useful in their daily work
- 95,28% have found the creation of IoT App somewhat easy, and among them more than 56% easy and very easy
- 93% have described the functions for IoT Apps (MicroServices) creation as complete and satisfactory, and among them 31,25% have evaluated it as very complete
- 50% of users have stated that the IoT App development has been 5 times faster if compared to what they know as possible with other tools at the state of the art.

Therefore, answers to our initial questions have been obtained. A short training has been sufficient to learn how to use the Snap4City IoT App

development environment, and it is typically considered faster than any other tools (typically they have used ETL, programming languages, and rule bases solutions). The implementation of the applications we presented by using other solutions or even Node-RED standard version is somehow unfeasible due to the complexity some of our MicroServices hide, for example, the routing, the picker for the heatmap, the discovery, the dashboard, the IoT registration and discovery, etc.

6.9 Considerations

Smart Cities are approaching the IoT world. Most of the early smart city solutions of first generation have been based on ETL processes mainly supporting pull protocols and data silos. The last generation of smart cities are massively using the IoT solutions, and thus are moving towards the use of push protocols / event driven. Therefore, the concept of Event Driven IoT Applications is becoming more widespread and viable with respect to the ETL or to rule-based solutions. In this chapter, these aspects have been reviewed, highlighting the requirements for smart city IoT Applications, as well as the ones specific for the MicroServices implementation for IoT Applications in Smart City frameworks. In this chapter, we have also presented Snap4City library of MicroServices for Node-RED, that has enabled the creation of a wide range of innovative IoT Applications for Smart City involving Dashboards, IoT Devices, personal storage GDPR compliant, geographical routing and geo-utilities, data analytics, etc. The solution proposed also addressed the formalization of the development life cycle that we have adopted and proposed to develop Cyber Physical System in Snap4City. The proposed solution has been validated against a large number of IoT Applications for the cities of Firenze, Antwerp and Helsinki. In addition, three applications have been reported in the paper and described, putting in evidence the relationships among MicroServices, Dashboard and data. Besides, the assessment of the solution has been performed and reported. The assessment has shown that the suite of MicroServices significantly could make the development time shorter, and it has been strongly appreciated by expert users. The research has been developed in the context of Select4Cities PCP of the European Commission, as Snap4City platform.

Chapter 7

Conclusions

The research activity presented in this thesis has been carried out at the DISIT laboratory (Distributed Data Intelligence and Technologies) of the DINFO department (Department of Information Engineering) of the University of Florence.

The activities were performed in the context of two main projects:

- Sii-Mobility: a national Smart City project on mobility and transportation management (co-founded by MIUR). In this context, the research reported in this thesis has contributed on aspects of user engagement through mobile applications and on management of incentives and campaigns in order to stimulate sustainable behaviour of citizens;
- Snap4City: a Smart City IoT platform (co-founded by Select4City, a PCP project of the European Commission). In this context, the research reported in this thesis has contributed to the development of the Smart City IoT solution and to the study and development of algorithms and systems to implement a secure environment regarding the GDPR aspects.

The result presented in the this thesis can be summarized as:

- the original solution, realized for the Sii-Mobility project, in the context of models, algorithms and systems developed to implement: (a) a language for engagement's definition; (b) a platform for computation of stimuli and messages to deliver to users in a contextualized way; (c) a management system for strategies, incentives and rewarding schema

based on user behavior. These activities required aspects of analysis of user modelling, user classification, transformation of general conditions and rules into customized actions, management of consumption of incentives and presentation of information in terms of indicators on dashboard. The user engagement system and the user's transportation modality classification have been validated during a Pilot deployed in the period April-June 2018 in Firenze, Pisa and Prato areas. It involved around 650 registered users and the three major public transport companies (Ataf, BUS Italia and CTT Nord). The main goal of the Pilot was to promote the use of public transport to users that commute by car for repetitive trips. The system has been able to targeting the audience and dispatch a set of engagements to eventually follow up with a set of prizes for the most virtuous users.

- the original solutions for the implementation of a secure and GDPR compliant platform for IoT in the context of the Snap4City project. In particular, the research activities contributed to the development of the IoT solution with the coordination and the integration of the IoT Directory and IoT Broker modules with the Snap4City platform. It was also possible to define and implement the several security aspects for the overall IoT architecture regarding the GDPR. The IoT platform for Smart City, its management of privacy and security aspects respecting the GDPR and its suite of MicroServices have been validated during the challenge launched by Select4Cities H2020 of the European Commission. Select4Cities identified a large number of requirements for modern Smart Cities supporting IoT in hands of public administrations and Living Labs. Snap4City demonstrated to have satisfied all the requirements proposed. The stress security assessment has been performed in a piloting period with more than 1200 registered users, thousands of processes per day, users on mobile Apps and more than 1.8 million of complex data ingested per day.

Appendix A

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.[1]

International Journals

1. Badii Claudio, Bellini Pierfrancesco, Cenni Daniele, **Difino Angelo**, Nesi Paolo and Paolucci Michela. "Analysis and assessment of a knowledge based smart city architecture providing service APIs", *Future Generation Computer System*, vol. 75, pp. 14-29, 2017
[DOI:10.1016/j.future.2017.05.001]

2. Badii Claudio, Bellini Pierfrancesco, **Difino Angelo** and Nesi Paolo. "Sii-mobility: An IoT/IoE architecture to enhance smart city mobility and transportation services", *Sensors*, vol. 19, iss. 1, pp 1-15, 2019
[DOI:10.3390/s19010001]

3. Badii Claudio, Bellini Pierfrancesco, **Difino Angelo**, Nesi Paolo, Pantaleo Gianni and Paolucci Michela. "MicroServices Suite for Smart City Applications", *Sensors*, vol. 19, iss. 21, pp 4789, 2019
[DOI:10.3390/s19214798]

Submitted

1. Badii Claudio, **Difino Angelo**, Nesi Paolo, Paoli Irene and Paolucci Michela. "Automated Classification of Users' Transportation Modality in Real Con-

[1]The author's bibliometric indices are the following: H-index = 9, total number of citations = 560 (source: Google Scholar on Month 10, 2019).

ditions", *Mobile Networks and Applications*, 2018 (submitted after major revision)

2. Badii Claudio, Bellini Pierfrancesco, **Difino Angelo** and Nesi Paolo. "Smart City IoT Platform Respecting GDPR Privacy and Security Aspects", *IEEE Access*, 2019 (submitted after major revision)

International Conferences and Workshops

1. Badii Claudio, Bellini Pierfrancesco, Cenni Daniele, **Difino Angelo**, Nesi Paolo and Paolucci, Michela. "User Engagement Engine for Smart City Strategies", in *Proc. of 3rd IEEE International Conference on Smart Computing (SMARTCOM)*, Hong Kong (China), 2017 [DOI:10.1109/SMARTCOMP.2017.7947059]
2. Badii Claudio, Bellini Pierfrancesco, **Difino Angelo** and Nesi Paolo. "Privacy and security aspects on a smart city iot platform", in *Proc. of 19th IEEE International Conference on Scalable Computing and Communication (SCALCOM)*, Leicester (UK), 2019 [DOI 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00250]

National Conferences

1. Badii Claudio, Bellini Emanuele, Bellini Pierfrancesco, Cenni Daniele, **Difino, Angelo**, Nesi Paolo and Paolucci Michela. "Km4City: una soluzione aperta per erogare servizi Smart City", in *Proc. of Conferenza GARR*, Florence, Italy, 2016 [DOI:10.26314/GARR-Conf16-proceedings-12]

Appendix B

Acronyms

ACK Acknowledgement

ADK Application Development Kit

AHP Analytic Hierarchy Process

AI Artificial Intelligence

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

AWS Amazon Web Services

BDS BeiDou Navigation Satellite System

BLE Bluetooth Low Energy

BPMN Business Process Model and Notation

CC Cloud Computing

CDMA Code Division Multiple Access

CDN Content Delivery Network

CKAN Comprehensive Knowledge Archive Network

CNN Convolutional Neural Networks

COAP Constrained Application Protocol

CPU Central Processing Unit

CRM Customer Relationship Management

CSS Cascading Style Sheets

CSV Comma Separated Value

ETL Extract Transform and Load

GDF Geographic Data Files

GDPR European General Data Protection Regulation

GLONASS Global Navigation Satellite System

GPS Global Position System

GSM Global System for Mobile communication

DBMS Database Management System

DINFO Department of Information Engineering of University of Florence

DISCES Distributed Smart City EngineDISCES

DISIT Distributed Systems and Internet Tech lab & Distributed Data Intelligence Lab of UNIFI

EOSC European Open Science Cloud

EIP-SCC European Innovation Partnership on Smart Cities and Communities

ENOLL European Network of Living Lab association

FEEL Friendly Enough Expression Language

FTP File Transfer Protocol

GIS Geographic Information System

GUI Graphical User Interface

H2M Human to Machine

HDFS Hadoop Distributed File System

HG Human Geography

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IaaS Infastructure as a Service

ICT Information and Communication Technology

IP Internet Protocol

IFTTT IF This Then That

IOE Internet Of Everythings

IOT Internet Of Things

IT information technology

JDBC Java DataBase Connector

JS JavaScript

JSON JavaScript Object Notation

JWT JSON Web Tokens

KB Knowledge Base

Km4City Knowledge Model For City

KPI Key Performance Index

LBS Location Based Services

LDAP Lightweight Directory Access Protocol

LoRa Long Range

LPWAN Low-Power Wide Area Networks

M2M Machine 2 Machine

MDD Model Driven Development

MAC Media Access Control

MIUR Italian Ministry of University and Research

MQTT Message Queuing Telemetry Transport

MS MicroSoft

MVEL MVFLEX Expression Language

NoSQL No Structured Query Language

O3 Ozone

OAUTH Open Authentication

OWL Web Ontology Language

OS Operating System

PaaS Platform as a Service

PCP Pre Commercial Procurement

PM2.5 Particulate Matter 2.5

PM10 Particulate Matter 10

POI Point Of Interest

PPOI Personal Point Of Interest

QoS Quality of Service

RAM Random Access Memory

RDF Resource Description Framework

RDFS Resource Description Framework Schema

REST Representational State Transfer

RF Random Forest

SAS Shared Access Signature

SE Simulation Error

SHA Secure Hash Algorithm

SKOS Simple Knowledge Organisation System

SLA Service Level Agreement

SM Supervisor & Monitor

SME Small Medium Enterprise

SPARQL SPARQL Protocol and RDF Query Language

SQL Structured Query Language

SSAP Static Server Allocation Problem

SSID Service Set Identifier

SSo Single Sign on

SSL Secure Sockets Layer

SV Support Vector

SVG Scalable Vector Graphics

SVM Support Vector Regression

TCP Transmission Control Protocol

TLS Transport Layer Security

TS Time Series

TSC Transportation Services

UEEngine User Engagement Engine

UI User Interface

UML Unified Modelling Language

UNIFI University of Florence

URI Uniform Resource Identifier

URL Uniform Resource Locator

VM Virtual Machine on cloud

VPL Visual Programming Language

W3C World Wide Web Consortium

WS Web Service

XML Extensible Markup Language

XSLT eXtensible Stylesheet Language Transformations

Bibliography

[1] "Airvantage iot solution," https://www.sierrawireless.com/products-and-solutions/sims-connectivity-and-cloud-services/iot-cloud-platform/, accessed: 2019-10-21.

[2] "Android 9.0 compatibility definition,," https://source.android.com/compatibility/android-cdd/, accessed: 2019-09-25.

[3] "Ardublock, a graphical programming language for arduino," http://blog.ardublock.com/, accessed: 2019-09-26.

[4] "Aws, amazon iot," https://aws.amazon.com/iot/, accessed: 2019-10-21.

[5] "Bosch iot," https://www.bosch-iot-suite.com, accessed: 2019-10-21.

[6] "Carriots iot," https://www.carriots.com, accessed: 2019-10-21.

[7] "Cisco iot," https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html, accessed: 2019-10-21.

[8] "Citysdk service development kit," https://www.citysdk.eu/, accessed: 2019-09-24.

[9] "Ckan data management system," https://ckan.org/, accessed: 2019-09-24.

[10] "Common alerting protocol (cap)," https://www.fema.gov/common-alerting-protocol, accessed: 2019-09-26.

[11] "Drools business rules management system," https://www.drools.org/, accessed: 2019-09-24.

[12] "Eu general data protection regulation," https://eugdpr.org/, accessed: 2019-09-26.

[13] "Feel expression language," https://www.omg.org/spec/DMN/1.2/PDF, accessed: 2019-09-24.

[14] "Fi-ware," https://www.fiware.org/, accessed: 2019-10-21.

[15] "Gartner press releases," https://www.gartner.com/newsroom/id/3598917, accessed: 2019-09-25.

[16] "Gdpr compliance for apps," https://www.privacypolicies.com/blog/gdpr-compliance-apps/, accessed: 2019-10-21.

[17] "Google iot," https://cloud.google.com/solutions/iot/, accessed: 2019-10-21.

[18] "Home kit apple," https://www.apple.com/ios/home/, accessed: 2019-10-21.

[19] "Ibm bluemix," http://www.ibm.com/software/bluemix/welcome/solutions2.html, accessed: 2019-09-26.

[20] "Ifttt connect different apps and devices," https://ifttt.com/, accessed: 2019-09-26.

[21] "Iot eclipse," https://iot.eclipse.org/, accessed: 2019-10-21.

[22] "Iot ignite," https://devzone.iot-ignite.com/documents/, accessed: 2019-10-21.

[23] "Kaa," https://kaaproject.github.io, accessed: 2019-10-21.

[24] "Ms azure iot," https://azure.microsoft.com/en-us/free/iot/, accessed: 2019-10-21.

[25] "Mvel expression language," http://mvel.documentnode.com/, accessed: 2019-09-24.

[26] "Netlab, tools for tangible interaction," http://www.netlabtoolkit.org/flash/reference/iot-devices/, accessed: 2019-09-26.

[27] "Nodered, a visual tool for wiring the internet-of-things," https://nodered.org/, accessed: 2019-09-26.

[28] "Open source ecosystem for event-driven apps," https://www.flogo.io/, accessed: 2019-09-26.

[29] "Opendatasoft data sharing platform," https://www.opendatasoft.com/, accessed: 2019-09-24.

[30] "Pentaho data integration," https://help.pentaho.com/Documentation/7.1/0D0/Pentaho_Data_Integration, accessed: 2019-09-26.

[31] "Ptc axeda iot platform," http://www.ptc.com/axeda/product/iot-platform, accessed: 2019-09-26.

[32] "Siemens mindsphere," https://new.siemens.com/global/en/products/software/mindsphere.html, accessed: 2019-10-21.

[33] "Smartthing samsung," https://smartthings.developer.samsung.com/, accessed: 2019-10-21.

[34] "A system to process and distribute data," https://nifi.apache.org/, accessed: 2019-09-26.

[35] "Thingsboard," https://thingsboard.io, accessed: 2019-10-21.

[36] "Thingworkx," https://www.ptc.com/it/products/iiot/thingworx-platform, accessed: 2019-10-21.

[37] "Thingworx iot platform," http://www.thingworx.com/, accessed: 2019-09-26.

[38] "Transportapi platform for transport data and development," https://www.transportapi.com/, accessed: 2019-09-24.

[39] M. Agbali, C. Trillo, I. A. Ibrahim, Y. Arayici, and T. Fernando, "Are smart innovation ecosystems really seeking to meet citizens' needs? insights from the stakeholders' vision on smart city strategy implementation," *Smart Cities*, vol. 2, no. 2, pp. 307–327, 2019.

[40] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[41] A. Alcaide, E. Palomar, J. Montero-Castillo, and A. Ribagorda, "Anonymous authentication for privacy-preserving iot target-driven applications," *Computers & Security*, vol. 37, pp. 111–123, 2013.

[42] M. Ammar, G. Russello, and B. Crispo, "Internet of things: A survey on the security of iot frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[43] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous computing*, vol. 7, no. 5, pp. 275–286, 2003.

[44] H. I. Ashqar, M. H. Almannaa, M. Elhenawy, H. A. Rakha, and L. House, "Smartphone transportation mode recognition using a hierarchical machine learning classifier and pooled features from time and frequency domains," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 244–252, 2018.

[45] S. Attfield, G. Kazai, M. Lalmas, and B. Piwowarski, "Towards a science of user engagement (position paper)," in *WSDM workshop on user modelling for Web applications*, 2011, pp. 9–12.

[46] Ž. Bačić, T. Jogun, and I. Majić, "Integrated sensor systems for smart cities," *Tehnički vjesnik*, vol. 25, no. 1, pp. 277–284, 2018.

[47] A. Badii, D. Carboni, A. Pintus, A. Piras, A. Serra, M. Tiemann, and N. Viswanathan, "Cityscripts: Unifying web, iot and smart city services in a smart citizen workspace," *JoWUA*, vol. 4, no. 3, pp. 58–78, 2013.

[48] C. Badii, E. G. Belay, P. Bellini, M. Marazzini, M. Mesiti, P. Nesi, G. Pantaleo, M. Paolucci, S. Valtolina, M. Soderi *et al.*, "Snap4city: A scalable iot/ioe platform for developing smart city applications," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2018, pp. 2109–2116.

[49] C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi, and M. Paolucci, "Analysis and assessment of a knowledge based smart city architecture providing service apis," *Future Generation Computer Systems*, vol. 75, pp. 14–29, 2017.

[50] C. Badii, P. Bellini, D. Cenni, A. Difino, M. Paolucci, and P. Nesi, "User engagement engine for smart city strategies," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2017, pp. 1–7.

[51] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Sii-mobility: An iot/ioe architecture to enhance smart city mobility and transportation services," *Sensors*, vol. 19, no. 1, p. 1, 2019.

[52] C. Badii, P. Bellini, A. Difino, P. Nesi, G. Pantaleo, and M. Paolucci, "Microservices suite for smart city applications," *Sensors*, vol. 19, no. 21, p. 4798, 2019.

[53] C. Badii, P. Nesi, and I. Paoli, "Predicting available parking slots on critical and regular services by exploiting a range of open data," *IEEE Access*, vol. 6, pp. 44 059–44 071, 2018.

[54] A. B. Bakker and E. Demerouti, "Towards a model of work engagement," *Career development international*, vol. 13, no. 3, pp. 209–223, 2008.

[55] D. Balsamo, A. Elboreini, B. M. Al-Hashimi, and G. V. Merrett, "Exploring arm mbed support for transient computing in energy harvesting iot systems," in *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE, 2017, pp. 115–120.

[56] D. Banister, "The sustainable mobility paradigm," *Transport policy*, vol. 15, no. 2, pp. 73–80, 2008.

[57] P. Bellini, C. Badii, P. Nesi, and A. Difino, "Privacy and security aspects on a smart city iot platform," in *19th IEEE International Conference on Scalable Computing and Communication (SCALCOM)*, 2019, pp. 1371–1376.

[58] P. Bellini, S. Bilotta, P. Nesi, M. Paolucci, and M. Soderi, "Wip: Traffic flow reconstruction from scattered data," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2018, pp. 264–266.

[59] P. Bellini, F. Bugli, M. Paolucci, D. Cenni, P. Nesi, G. Pantaleo, and I. Zaza, "Data flow management and visual analytic for big data smart city/iot," in

19th IEEE International Conference on Scalable Computing and Communication (SCALCOM), 2019, pp. 1529–1536.

[60] P. Bellini, D. Cenni, and P. Nesi, "Optimization of information retrieval for cross media contents in a best practice network," *International Journal of Multimedia Information Retrieval*, vol. 3, no. 3, pp. 147–159, 2014.

[61] P. Bellini, M. Marazzini, N. Mitolo, M. Paolucci, D. Cenni, and P. Nesi, "Smart city control room dashboards exploiting big data infrastructure," in *DMSVIVA*, 2018, pp. 44–50.

[62] P. Bellini, P. Nesi, M. Paolucci, and I. Zaza, "Smart city architecture for data ingestion and analytics: Processes and solutions," in *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, 2018, pp. 137–144.

[63] F. Biljecki, H. Ledoux, and P. Van Oosterom, "Transportation mode-based segmentation and classification of movement trajectories," *International Journal of Geographical Information Science*, vol. 27, no. 2, pp. 385–407, 2013.

[64] R. Bonney, C. B. Cooper, J. Dickinson, S. Kelling, T. Phillips, K. V. Rosenberg, and J. Shirk, "Citizen science: a developing tool for expanding science knowledge and scientific literacy," *BioScience*, vol. 59, no. 11, pp. 977–984, 2009.

[65] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[66] T. Bovaird, "Beyond engagement and participation: User and community coproduction of public services," *Public administration review*, vol. 67, no. 5, pp. 846–860, 2007.

[67] L. Breiman *et al.*, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical science*, vol. 16, no. 3, pp. 199–231, 2001.

[68] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in europe," *Journal of urban technology*, vol. 18, no. 2, pp. 65–82, 2011.

[69] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola, "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[70] M. X. D. Carpini, F. L. Cook, and L. R. Jacobs, "Public deliberation, discursive participation, and citizen engagement: A review of the empirical literature," *Annu. Rev. Polit. Sci.*, vol. 7, pp. 315–344, 2004.

[71] M. Cavada, M. R. Tight, and C. Rogers, "A smart city case study of singapore is singapore truly smart?" in *Smart City Emergence*. Elsevier, 2019, pp. 295–314.

[72] D. Cenni, P. Nesi, G. Pantaleo, and I. Zaza, "Twitter vigilance: a multi-user platform for cross-domain twitter data analytics, nlp and sentiment analysis," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017, pp. 1–8.

[73] T. Chattopadhyay, A. Ghose, A. Mukherjee, S. Maiti, and A. Pal, "Automated workflow formation for iot analytics: A case study," in *International Internet of Things Summit*. Springer, 2015, pp. 36–43.

[74] I. Chatzigiannakis, A. Vitaletti, and A. Pyrgelis, "A privacy-preserving smart parking system using an iot elliptic curve based security platform," *Computer Communications*, vol. 89, pp. 165–177, 2016.

[75] R. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 482–495, 2014.

[76] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things journal*, vol. 1, no. 4, pp. 349–359, 2014.

[77] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, pp. 1–4, 2015.

[78] J. Choque, L. Diez, A. Medela, and L. Muñoz, "Experimentation management in the co-created smart-city: Incentivization and citizen engagement," *Sensors*, vol. 19, no. 2, p. 411, 2019.

[79] Y.-k. Chou, "Actionable gamification," *Beyond points, badges, and leaderboards*, 2015.

[80] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[81] B. Claerhout and G. DeMoor, "Privacy protection for clinical and genomic data: The use of privacy-enhancing techniques in medicine," *International Journal of Medical Informatics*, vol. 74, no. 2-4, pp. 257–265, 2005.

[82] A. Cocchia, "Smart and digital city: A systematic literature review," in *Smart city*. Springer, 2014, pp. 13–43.

[83] R. Contreras Masse, "Application of iot with haptics interface in the smart manufacturing industry," *Instituto de Ingeniería y Tecnología*, 2019.

[84] M. Csikszentmihalyi, *Flow: The psychology of happiness*. Random House, 2013.

[85] S. de Luca, "Public engagement in strategic transportation planning: An analytic hierarchy process based approach," *Transport Policy*, vol. 33, pp. 110–124, 2014.

[86] A. M. Del Esposte, F. Kon, F. M. Costa, and N. Lago, "Interscity: A scalable microservice-based open source platform for smart cities." in *SMARTGREENS*, 2017, pp. 35–46.

[87] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*. ACM, 2011, pp. 9–15.

[88] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Physical review letters*, vol. 77, no. 13, p. 2818, 1996.

[89] S. Dey, D. Jaiswal, H. S. Paul, and A. Mukherjee, "A semantic algorithm repository and workflow designer tool: signal processing use case," in *International Internet of Things Summit*. Springer, 2015, pp. 53–61.

[90] J. P. Dias and H. S. Ferreira, "State of the software development life-cycle for the internet-of-things," *arXiv preprint arXiv:1811.04159*, 2018.

[91] R. Díaz-Díaz, L. Muñoz, and D. Pérez-González, "Business model analysis of public services operating in the smart city ecosystem: The case of smartsantander," *Future Generation Computer Systems*, vol. 76, pp. 198–214, 2017.

[92] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, p. 116, 2019.

[93] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for iot," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*. ACM, 2017, pp. 173–178.

[94] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," in *Present and ulterior software engineering*. Springer, 2017, pp. 195–216.

[95] S. W. Draper, "Analysing fun as a candidate software requirement," *Personal Technologies*, vol. 3, no. 3, pp. 117–122, 1999.

[96] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *International conference on pervasive computing*. Springer, 2005, pp. 152–170.

[97] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, 2019.

[98] A. S. Elmaghraby and M. M. Losavio, "Cyber security challenges in smart cities: Safety, security and privacy," *Journal of advanced research*, vol. 5, no. 4, pp. 491–497, 2014.

[99] D. Evans and D. M. Eyers, "Efficient data tagging for managing privacy in the internet of things," in *2012 IEEE International Conference on Green Computing and Communications*. IEEE, 2012, pp. 244–248.

[100] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[101] M. Fazio, M. Paone, A. Puliafito, and M. Villari, "Heterogeneous sensors become homogeneous things in smart cities," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2012, pp. 775–780.

[102] J.-M. Fernandez, I. Vidal, and F. Valera, "Enabling the orchestration of iot slices through edge and cloud microservice platforms," *Sensors*, vol. 19, no. 13, p. 2980, 2019.

[103] J. Fiksel, "Sustainability and resilience: toward a systems approach," *Sustainability: Science, Practice and Policy*, vol. 2, no. 2, pp. 14–21, 2006.

[104] J. D. Finn, "Withdrawing from school," *Review of educational research*, vol. 59, no. 2, pp. 117–142, 1989.

[105] D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, and R. L. Mandryk, "Calibration games: making calibration tasks enjoyable by adding motivating game elements," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 403–412.

[106] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *icml*, vol. 96. Citeseer, 1996, pp. 148–156.

[107] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[108] R. Garris, R. Ahlers, and J. E. Driskell, "Games, motivation, and learning: A research and practice model," *Simulation & gaming*, vol. 33, no. 4, pp. 441–467, 2002.

[109] M. Gebresselassie and T. Sanchez, "Smart tools for socially sustainable transport: A review of mobility apps," *Urban Science*, vol. 2, no. 2, p. 45, 2018.

[110] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[111] Z. Ghahramani, "An introduction to hidden markov models and bayesian networks," in *Hidden Markov models: applications in computer vision*. World Scientific, 2001, pp. 9–41.

[112] F. Giannotti and D. Pedreschi, *Mobility, data mining and privacy: Geographic knowledge discovery*. Springer Science & Business Media, 2008.

[113] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.

[114] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[115] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, 2003, pp. 31–42.

[116] P. Gupta, S. Chauhan, and M. Jaiswal, "Classification of smart city research-a descriptive literature review and future research agenda," *Information Systems Frontiers*, pp. 1–25, 2019.

[117] J. Hamari, "Transforming homo economicus into homo ludens: A field experiment on gamification in a utilitarian peer-to-peer trading service," *Electronic commerce research and applications*, vol. 12, no. 4, pp. 236–245, 2013.

[118] J. Hamari, J. Koivisto, H. Sarsa *et al.*, "Does gamification work?-a literature review of empirical studies on gamification," in *HICSS*, vol. 14, no. 2014, 2014, pp. 3025–3034.

[119] J. Hamari and V. Lehdonvirta, "Game design as marketing: How game mechanics create demand for virtual goods," *International Journal of Business Science & Applied Management*, vol. 5, no. 1, pp. 14–29, 2010.

[120] T. Hammond, T. Hannay, B. Lund, and J. Scott, "Social bookmarking tools (i)," *D-lib Magazine*, vol. 11, no. 4, pp. 1082–9873, 2005.

[121] J. Haworth and T. Bantis, "Who you are is how you travel: A framework for transportation mode detection using individual and environmental characteristics," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 286–309, 2017.

[122] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM conference on embedded networked sensor systems*. ACM, 2013, p. 13.

[123] O. L. HOUSE, "Science and society. third report," https://publications.parliament.uk/pa/ld199900/ldselect/ldsctech/38/3802.htm, 2000, accessed: 2019-09-24.

[124] B. Hu, T. Patkos, A. Chibani, and Y. Amirat, "Rule-based context assessment in smart cities," in *International Conference on Web Reasoning and Rule Systems*. Springer, 2012, pp. 221–224.

[125] K. Huotari and J. Hamari, "Defining gamification: a service marketing perspective," in *Proceeding of the 16th international academic MindTrek conference*. ACM, 2012, pp. 17–22.

[126] M. Ignaccolo, G. Inturri, M. Garcia-Melon, N. Giuffrida, M. Le Pira, and V. Torrisi, "Combining analytic hierarchy process (ahp) with role-playing games for stakeholder engagement in complex transport decisions," *Transportation Research Procedia*, vol. 27, pp. 500–507, 2017.

[127] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[128] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet of Things journal*, vol. 1, no. 2, pp. 112–121, 2014.

[129] A. Kamilaris, F. Gao, F. X. Prenafeta-Boldú, and M. I. Ali, "Agri-iot: A semantic framework for internet of things-enabled smart farming applications," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 442–447.

[130] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *IEEE Internet of things Journal*, vol. 1, no. 3, pp. 265–275, 2014.

[131] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[132] R. Kitchin, "The real-time city? big data and smart urbanism," *GeoJournal*, vol. 79, no. 1, pp. 1–14, 2014.

[133] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, vol. 141, pp. 199–221, 2018.

[134] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city internet of things platform with microservice architecture," in *2015 3rd International Conference on Future Internet of Things and Cloud*. IEEE, 2015, pp. 25–30.

[135] J. H. Lee, M. G. Hancock, and M.-C. Hu, "Towards an effective framework for building smart cities: Lessons from seoul and san francisco," *Technological Forecasting and Social Change*, vol. 89, pp. 80–99, 2014.

[136] J. Lehmann, M. Lalmas, E. Yom-Tov, and G. Dupret, "Models of user engagement," in *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 2012, pp. 164–175.

[137] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[138] S. Li, L. Da Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1–9, 2018.

[139] X. Li, M. Ge, X. Dai, X. Ren, M. Fritsche, J. Wickert, and H. Schuh, "Accuracy and reliability of multi-gnss real-time precise positioning: Gps, glonass, beidou, and galileo," *Journal of Geodesy*, vol. 89, no. 6, pp. 607–635, 2015.

[140] C. Lim, K.-J. Kim, and P. P. Maglio, "Smart cities with big data: Reference models, challenges, and considerations," *Cities*, vol. 82, pp. 86–99, 2018.

[141] P. Lombardi, S. Giordano, H. Farouh, and W. Yousef, "Modelling the smart city performance," *Innovation: The European Journal of Social Science Research*, vol. 25, no. 2, pp. 137–149, 2012.

[142] I. Lorenzoni, S. Nicholson-Cole, and L. Whitmarsh, "Barriers perceived to engaging with climate change among the uk public and their policy implications," *Global environmental change*, vol. 17, no. 3-4, pp. 445–459, 2007.

[143] V. Manzoni, D. Maniloff, K. Kloeckl, and C. Ratti, "Transportation mode identification and real-time co2 emission estimation using smartphones," *SENSEable City Lab, Massachusetts Institute of Technology, nd*, 2010.

[144] I. G. Martí, L. E. Rodríguez, M. Benedito, S. Trilles, A. Beltrán, L. Díaz, and J. Huerta, "Mobile application for noise pollution monitoring through gamification techniques," in *International Conference on Entertainment Computing*. Springer, 2012, pp. 562–571.

[145] D. R. May, R. L. Gilson, and L. M. Harter, "The psychological conditions of meaningfulness, safety and availability and the engagement of the human spirit at work," *Journal of occupational and organizational psychology*, vol. 77, no. 1, pp. 11–37, 2004.

[146] C. M. Medaglia and A. Serbanati, "An overview of privacy and security issues in the internet of things," in *The internet of things*. Springer, 2010, pp. 389–395.

[147] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of internet-of-things platforms," *Computer Communications*, vol. 89, pp. 5–16, 2016.

[148] P. Misra and P. Enge, "Global positioning system: signals, measurements and performance second edition," *Global Positioning System: Signals, Measurements And Performance Second Editions,*, 2006.

[149] M. F. Muhammad, W. Anjum, and K. S. Mazhar, "A critical analysis on the security concerns of internet of things (iot)," *International Journal of Computer Applications (0975 8887)*, vol. 111, no. 7, 2015.

[150] C. I. Muntean, "Raising engagement in e-learning through gamification," in *Proc. 6th International Conference on Virtual Learning ICVL*, vol. 1, 2011, pp. 323–329.

[151] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," in *Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times*. ACM, 2011, pp. 282–291.

[152] P. Nesi, C. Badii, P. Bellini, D. Cenni, G. Martelli, and M. Paolucci, "Km4city smart city api: an integrated support for mobility services," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016, pp. 1–8.

[153] P. Nesi, P. Bellini, and D. Cenni, "Ap positioning for estimating people flow as origin destination matrix for smart cities," in *DMS*, 2016, pp. 202–209.

[154] P. Nesi, G. Pantaleo, and G. Sanesi, "A distributed framework for nlp-based keyword and keyphrase extraction from web pages and documents," in *DMS*, 2015, pp. 155–161.

[155] T. Neubauer and J. Heurix, "A methodology for the pseudonymization of medical data," *International journal of medical informatics*, vol. 80, no. 3, pp. 190–204, 2011.

[156] N. Novielli, "Hmm modeling of user engagement in advice-giving dialogues," *Journal on Multimodal User Interfaces*, vol. 3, no. 1-2, pp. 131–140, 2010.

[157] H. L. O'Brien and E. G. Toms, "What is user engagement? a conceptual framework for defining user engagement with technology," *Journal of the American society for Information Science and Technology*, vol. 59, no. 6, pp. 938–955, 2008.

[158] C. O'Faircheallaigh, "Public participation and environmental impact assessment: Purposes, implications, and lessons for public policy making," *Environmental impact assessment review*, vol. 30, no. 1, pp. 19–27, 2010.

[159] D. O'SULLIVAN, B. Smyth, and D. Wilson, "Understanding case based recommendation: A similarity knowledge perspective," *International Journal on Artificial Intelligence Tools*, vol. 14, no. 01n02, pp. 215–232, 2005.

[160] A. Pal, A. Mukherjee, and P. Balamuralidhar, "Model-driven development for internet of things: Towards easing the concerns of application developers," in *International Internet of Things Summit*. Springer, 2014, pp. 339–346.

[161] E. Patti and A. Acquaviva, "Iot platform for smart cities: Requirements and implementation case studies," in *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE, 2016, pp. 1–6.

[162] G. Pau, T. Campisi, A. Canale, A. Severino, M. Collotta, and G. Tesoriere, "Smart pedestrian crossing management at traffic light junctions through a fuzzy-based approach," *Future Internet*, vol. 10, no. 2, p. 15, 2018.

[163] J. Pavlus, "The game of life," *Scientific American*, vol. 303, no. 6, pp. 43–44, 2010.

[164] T. Pereira, L. Barreto, and A. Amaral, "Network and information security challenges within industry 4.0 paradigm," *Procedia Manufacturing*, vol. 13, pp. 1253–1260, 2017.

[165] P. Persson and O. Angelsmark, "Calvin–merging cloud and iot," *Procedia Computer Science*, vol. 52, pp. 210–217, 2015.

[166] E. C. Polley and M. J. Van Der Laan, "Super learner in prediction," 2010.

[167] J. Poncela, P. Vlacheas, R. Giaffreda, S. De, M. Vecchio, S. Nechifor, R. Barco, M. C. Aguayo-Torres, V. Stavroulaki, K. Moessner *et al.*, "Smart cities via data aggregation," *Wireless personal communications*, vol. 76, no. 2, pp. 149–168, 2014.

[168] A. C. Prelipcean, G. Gidófalvi, and Y. Susilo, "Transportation mode detection–an in-depth review of applicability and reliability," *Transport reviews*, vol. 37, no. 4, pp. 442–464, 2017.

[169] A. C. Prelipcean, G. Gidófalvi, and Y. O. Susilo, "Mobility collector," *Journal of Location Based Services*, vol. 8, no. 4, pp. 229–255, 2014.

[170] D. Preuveneers, W. Joosen, and E. Ilie-Zudor, "Data protection compliance regulations and implications for smart factories of the future," in *2016 12th International Conference on Intelligent Environments (IE)*. IEEE, 2016, pp. 40–47.

[171] L. F. Rahman, T. Ozcelebi, and J. Lukkien, "Understanding iot systems: a life cycle approach," *Procedia computer science*, vol. 130, pp. 1057–1062, 2018.

[172] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, "Urban planning and building smart cities based on the internet of things using big data analytics," *Computer Networks*, vol. 101, pp. 63–80, 2016.

[173] P. P. Ray, "A survey on visual programming languages in internet of things," *Scientific Programming*, vol. 2017, 2017.

[174] P. Ray Partha, "A survey on internet of things architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018.

[175] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.

[176] C. C. Reyes-Aldasoro, M. K. Griffiths, D. Savas, and G. M. Tozer, "Caiman: an online algorithm repository for cancer image analysis," *Computer methods and programs in biomedicine*, vol. 103, no. 2, pp. 97–103, 2011.

[177] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, and A. Bouabdallah, "A systemic approach for iot security," in *2013 IEEE international conference on distributed computing in sensor systems*. IEEE, 2013, pp. 351–355.

[178] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the internet of things," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 147–159, 2011.

[179] A. Roukounaki, J. Soldatos, R. Petrolo, V. Loscri, N. Mitton, and M. Serrano, "Visual development environment for semantically interoperable smart cities applications," in *International Internet of Things Summit*. Springer, 2015, pp. 439–449.

[180] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *American psychologist*, vol. 55, no. 1, p. 68, 2000.

[181] T. L. Saaty, "What is the analytic hierarchy process?" in *Mathematical models for decision support*. Springer, 1988, pp. 109–121.

[182] Y. B. Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Trust management system design for the internet of things: A context-aware and multi-service approach," *Computers & Security*, vol. 39, pp. 351–365, 2013.

[183] K. Salen, K. S. Tekinbaş, and E. Zimmerman, *Rules of play: Game design fundamentals*. MIT press, 2004.

[184] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," technical report, SRI International, Tech. Rep., 1998.

[185] E. F. Z. Santana, A. P. Chaves, M. A. Gerosa, F. Kon, and D. S. Milojicic, "Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 78, 2018.

[186] A. Sauerländer-Biebl, E. Brockfeld, D. Suske, and E. Melde, "Evaluation of a transport mode detection using fuzzy rules," *Transportation research procedia*, vol. 25, pp. 591–602, 2017.

[187] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, "Smart cities and the future internet: Towards cooperation frameworks for open innovation," in *The future internet assembly*. Springer, 2011, pp. 431–446.

[188] W. B. Schaufeli, "The measurement of work engagement," in *Research Methods in Occupational Health Psychology*. Routledge, 2012, pp. 162–178.

[189] P. Sethi and S. R. Sarangi, "Internet of things: architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.

[190] R. C. Shah, C.-y. Wan, H. Lu, and L. Nachman, "Classifying the mode of transportation on mobile phones using gis information," in *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing*. ACM, 2014, pp. 225–229.

[191] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer networks*, vol. 76, pp. 146–164, 2015.

[192] S. K. Sowe, T. Kimata, M. Dong, and K. Zettsu, "Managing heterogeneous sensor data on a big data platform: Iot services for data-intensive science," in *2014 IEEE 38th International Computer Software and Applications Conference Workshops*. IEEE, 2014, pp. 295–300.

[193] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and gis information," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2011, pp. 54–63.

[194] P. Stopher, C. FitzGerald, and J. Zhang, "Search for a global positioning system device to measure person travel," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 3, pp. 350–369, 2008.

[195] K. Su, J. Li, and H. Fu, "Smart city and the applications," in *2011 international conference on electronics, communications and control (ICECC)*. IEEE, 2011, pp. 1028–1031.

[196] E. Swyngedouw, "Governance innovation and the citizen: The janus face of governance-beyond-the-state," *Urban studies*, vol. 42, no. 11, pp. 1991–2006, 2005.

[197] O. Tene, "Privacy law's midlife crisis: A critical assessment of the second wave of global privacy laws," *Ohio St. LJ*, vol. 74, p. 1217, 2013.

[198] S. Valtolina, F. Hachem, B. R. Barricelli, E. G. Belay, S. Bonfitto, and M. Mesiti, "Facilitating the development of iot applications in smart city platforms," in *International Symposium on End User Development*. Springer, 2019, pp. 83–99.

[199] M. J. Van Der Laan and S. Dudoit, "Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples," 2003.

[200] M. J. van der Laan, S. Dudoit, and A. W. van der Vaart, "The cross-validated adaptive epsilon-net estimator," *Statistics & Decisions*, vol. 24, no. 3, pp. 373–395, 2006.

[201] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Statistical applications in genetics and molecular biology*, vol. 6, no. 1, 2007.

[202] J. Van Doorn, K. N. Lemon, V. Mittal, S. Nass, D. Pick, P. Pirner, and P. C. Verhoef, "Customer engagement behavior: Theoretical foundations and research directions," *Journal of service research*, vol. 13, no. 3, pp. 253–266, 2010.

[203] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2724–2737, 2013.

[204] P. C. Verhoef, W. J. Reinartz, and M. Krafft, "Customer engagement as a new perspective in customer management," *Journal of service research*, vol. 13, no. 3, pp. 247–252, 2010.

[205] G. Viale Pereira, M. A. Cunha, T. J. Lampoltshammer, P. Parycek, and M. G. Testa, "Increasing collaboration and participation in smart city governance: a cross-case analysis of smart city initiatives," *Information Technology for Development*, vol. 23, no. 3, pp. 526–553, 2017.

[206] S. D. Vivek, S. E. Beatty, and R. M. Morgan, "Customer engagement: Exploring customer relationships beyond purchase," *Journal of marketing theory and practice*, vol. 20, no. 2, pp. 122–146, 2012.

[207] J. Voas, R. Kuhn, C. Kolias, A. Stavrou, and G. Kambourakis, "Cybertrust in the iot age," *Computer*, vol. 51, no. 7, pp. 12–15, 2018.

[208] S. Wang, C. Chen, and J. Ma, "Accelerometer based transportation mode recognition on mobile phones," in *2010 Asia-Pacific Conference on Wearable Computing Systems*. IEEE, 2010, pp. 44–46.

[209] N. Whitton, "Game engagement theory and adult learning," *Simulation & Gaming*, vol. 42, no. 5, pp. 596–609, 2011.

[210] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2014, pp. 417–423.

[211] G. Yanyun, Z. Fang, C. Shaomeng, and L. Haiyong, "A convolutional neural networks based transportation mode identification algorithm," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2017, pp. 1–7.

[212] K. Yasumoto, H. Yamaguchi, and H. Shigeno, "Survey of real-time processing technologies of iot data streams," *Journal of Information Processing*, vol. 24, no. 2, pp. 195–202, 2016.

[213] M.-C. Yu, T. Yu, S.-C. Wang, C.-J. Lin, and E. Y. Chang, "Big data small footprint: the design of a low-power classifier for detecting transportation modes," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1429–1440, 2014.

[214] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[215] A. O. Zezzatti, R. Contreras-Massé, and J. Mejía, "Innovative data visualization of collisions in a human stampede occurred in a religious event using multiagent systems," in *2019 23rd International Conference in Information Visualization–Part II*. IEEE, 2019, pp. 62–67.

[216] K. Zhao and L. Ge, "A survey on the internet of things security," in *2013 Ninth international conference on computational intelligence and security*. IEEE, 2013, pp. 663–667.

[217] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma, "Recommending friends and locations based on individual location history," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 5, 2011.

[218] G. Zichermann and C. Cunningham, *Gamification by design: Implementing game mechanics in web and mobile apps*. " O'Reilly Media, Inc.", 2011.

[219] O. Zimmermann, "Microservices tenets," *Computer Science-Research and Development*, vol. 32, no. 3-4, pp. 301–310, 2017.