Design and Develop of a Smart City Digital Twin with 3D Representation and User Interface for What-If Analysis

Design and Develop of a Smart City Digital Twin with 3D Representation and User Interface for What-If Analysis

Lorenzo Adreani, Pierfrancesco Bellini, Marco Fanfani, Paolo Nesi, Gianni Pantaleo

DISIT Lab, https://www.disit.org, https://www.sanp4city.org,
University of Florence, Florence, Italy

<name>.<surname>@unifi.it

Abstract. Digital Twins of Smart Cities are fundamental tools for decision makers since they can provide interactive 3D visualizations of the city enriched with real-time information and connected to actual complete digital model of the entities with all their heterogeneous data/info. Such a technology can be exploited to observe the status of the city, and to perform analysis and simulations, and thus to develop strategies. Indeed, such solutions must satisfy a series of requirements spanning from the 3D construction to the interactive functionality of user interface for the decision makers. In this paper, a Smart City Digital Twin model and tools are presented, which satisfy a wide range of requirements. The principles at the basis of the design and development are reported and discussed. The solution has been developed on top of Snap4City platform and validated on Florence City case (Italy), in CN Mobility of Ministry. Finally, a comparison among several different Smart City Digital Twin solutions is offered.

Keywords: Digital Twin, Smart City, 3D City Model.

1 Introduction

Nowadays Smart City Digital Twins, SCDTs, are becoming ever more relevant in academic, government, and industrial fields since they can offer a virtual context that replicates a real city with high fidelity, typically exploiting 3D models of the buildings enriched with structural and contextual information coming from IoT (Internet of Things) sensors, heatmaps, analytic services, building information modelling, etc. [1]. Such solutions can provide a fundamental tool for city decision makers and stakeholders which can observe in real-time the status of the city, and perform analysis and simulations in different application domains like urban planning, mobility and transport, energy, disaster analysis and prevention, air pollution monitoring, city planning, etc.

On the one hand, in the past years several works addressed the problem of 3D city modelling in order to create realistic virtual visualizations of the city structure. Clearly, since such 3D models must cover city-wide areas, both the production of the 3D models, as well as their handling and processing pose a challenging task still to be solved [2]. With the purpose of defining an adequate format for the data to be displayed on a web interface, CityGML [3] proposed a series of requirements according to different

Level of Details (LoD) covered by the model. Five LoD have been defined: LoD0 includes 2D maps and 3D terrain only, LoD1 introduce simple box-like 3D building models that are enhanced with 3D roof structures in LoD2. LoD3 introduces realistic textures, and finally LoD4 describes building interiors. In CityGML3.0, Building Information Modelling (BIM) are included using the Industrial Foundation Class (IFC) format [4].

On the other hand, a SCDT must include information about Point of Interest (POI) locations, IoT Sensor positions and readings, heatmaps to show for example the dispersion of pollutants, paths (e.g., cycling paths) and areas (e.g., city districts). To enhance the 3D model with this kind of knowledge, the SCDT should be embedded into an adequate platform capable to model any data kind, and capable to ingest, manipulate, and index static and real-time data that can then be retrieved using specific API.

In this paper, firstly, a series of requirements for SCDT are reported and discussed in Section 2 expanding those presented in [5] considering the required data to be included into a SCDT, and taking into account the users' interaction functionalities that the SCDT interface should provide, and some additional operative requirements that should be satisfied in order to enhance the functionalities offered by a SCDT. Then, we present our SCDT solution describing, in Section 3, the development phases; in Section 4, how the 3D building models were obtained, and, finally in Section 5, how the web interface was developed to integrate all the data and distribute proposed SCDT on a web browser. In Section 6, the case of study for Florence (Italy) is presented, showing the possibility of performing What-If analysis via the 3D visual interface. Section 7 provides a comparison with other SCDT and 3D city representations on the basis of the identified requirements. Finally, in Section 8, conclusions are drawn and future work highlighted.

2 Requirement analysis

In this section, the requirements that a SCDT should satisfy are reported and discussed. Clearly, the quality of the 3D representations of buildings is a key aspect of a SCDT. For example, in [3] different levels of detail (LoD) were proposed. However, a SCDT must comply with additional requirements that take into account the 3D representation and the integration of additional massive data to provide a complete tool at support of decision makers. In [5], a list of requirements was proposed to specify which elements should be visualized on a 3D representation for SCDT, and to point out the interactivity aspects to be addressed to guarantee a suitable user experience.

In this paper, we strongly revised and further expanded the requirement list considering aspects related to the data management, advanced interactivity, costs and licensing aspects. Hereafter, a revised and augmented list of requirements is reported. Firstly, requirements on data are presented, then the requirements to offer interactive controls to the user are reported. Finally, the most relevant operative requirements are included.

Requirements on data (RD) to be included in a SCDT are:

RD1. Buildings of the city. Each single building should be represented with details in terms of shape (facades, roof, towers, cupolas, etc.), and **patterns on facades**

and roofs. Multiple representations at different LoD could be included. For example, (i) LoD1 structure obtained by extruding the building plans up to the heights of the eaves, or (ii) higher LoD structure represented as 3D meshes, and (iii) BIM should be included.

RD2. Ground information as road shapes and names, names of squares and localities, etc., exploiting the so called Orthomaps, with eventual real aerial view patterns, and the actual graph road information for picking and connecting elements. Orthomaps are typically provided in terms of multi resolution tiled images from GIS systems using WMS protocol, while the road graph can be coarsely recovered from Open Street Map or from institutional cadastre.

RD3. Heatmaps are typically superimposed (with variable transparency) on the ground level without overlapping the buildings. For example, to represent temperature, traffic flow, pollutant distribution, people flow, noise, humidity, etc. In this case, they are typically provided in term of multi resolution geolocated tiled images, provided by GIS using WMS protocol. In some cases, a time sequence of heatmap can be used to show the evolution of the distribution over time. This aspect adds complexity to the model, because the heatmap can be shaped, calibrated with colormaps, and compounded by different elementary blocks of any shape, or points.

RD4. Paths and areas shapes can be super-imposed over the ground and heatmaps levels without overlapping the buildings. Such data can be used to describe perimeters of gardens, cycling paths, trajectories, border of gov areas, elements of origin destination matrices, etc. This information is quite specific and must be produced on the basis of information recovered from some Open Data. Once recovered it can be distributed by using GIS in WFS/WMS protocols.

RD5. PINs marking the position of services, IoT Devices, Point of Interest (POI), Key Performance Indicator (KPI), etc., and providing clickable information according to some data model which may provide access to Time Series, shapes, etc. This information is quite specific and can be produced on the basis of the information recovered from Private and/or Open Data.

RD6. Terrain information and elevation: the elevation of each single building has to be taken into account and thus the skyline of the city may include surrounding mountains, in city hills, etc. This means that the buildings and Orthomaps should be placed according to the terrain elevation, the so-called Digital Terrain Model, DTM. The DTM can be distributed via WMS from GIS.

RD7. Additional 3D entities for completing the realism of the scenario, such as: (i) trees, benches, fountains, semaphores, digital signages, luminaries, and any other city furniture, etc.; and (ii) water bodies should be included into the digital twin to better represent rivers, lakes, fountains, etc.

In addition, the framework used to present the SCDT must met some requirements of interactivity with the model and the 3D elements (RI):

RI1. Map controls to change the point of view by zooming, rotating, tilting, and panning the scene. Changing the light source position, simulating different times of day/night, should be possible. This should lead to produce shades projected by buildings on ground and other buildings, and a different illumination from direct to

indirect exposition to daylight, eventual reflections, and transparencies. Picking on map to recover position and eventual information associated with the city structure and road graph.

RI2. Dynamic sky to show different sky conditions according to the time of the day, weather, and/or weather forecast.

RI3. PIN data access to show data associated with IoT Devices, POI, KPI, etc., including real time and historical data, time series, or detailed information, and corresponding drill down facilities.

RI4. Building picking/manipulation: to provide the possibility to select single building to: (i) show detailed information associated with the building, or (ii) move into a BIM view of the building, with the possibility of navigating into the building structure, and again to access the internal data associate to PINs into the building, or (iii) to change the building 3D model with a new one to have an evolution over time.

RI5. Twin management as independent element management loading to have the possibility to hide, show, replace specific element as entities modelled as Digital Twins, for example to disable the building view to see only the city PINs, or to load different heatmaps or paths.

RI6. Business logic call-back to provide the possibility of selecting an element (3D, PIN, ground, heatmap) to provoke a call back into a business logic tool for provoking events and actions in the systems, at which the developers may associate intelligence activities, analytics, other views, etc.

RI7. Underground and elements inspection to provide the possibility of selecting and inspecting specific areas and see detailed 3D elements placed underground or inside the buildings, such as water pipes, metro lines, etc. And interacting with complex elements for example, traffic flow, cycling paths, scenarios, city decors, etc.

RI8. Virtual 3D structures as dynamic PIN changing colour and/or size/shape according to some data value, OD flows with jumps/arcs, etc. Dynamic pins as SVG shape and colour by changing with some real time value. Dynamic pins as 3D solids changing colour or size according to some real time value.

Finally, there are some operative requirements (RO) that must be met to guarantee an accessible, integrated, and affordable SCDT solution:

RO1. Data analytic processes must be available to let the user develops and/or execute specific data analytics.

RO2. Smart Data Model compatibility to guarantee interoperable and replicable Smart Cities, interoperability at level of data formats, federation at level of protocols and APIs.

RO3. Logics for data ingestion and transformation required to ingest data from IoT sensors, and other sources and transform them into different data models and formats.

RO4. Dynamic data management to have new PINs or elements to be automatically reported in the SCDT as soon as they are included in the platform, event driven rendering of data.

RO5. Integration with workflow management systems for ticket management. For example, when an event of a streetlamp fault is detected, in the SCDT the faulty streetlamp is highlighted together with its connection to the electric infrastructure in order to ease the maintenance work.

RO6. Web player: (i) the SCDT must be accessible thought a web browser without additional plugins, and (ii) the player must be released with open or free license.

RO7. No reloading: changes in the SCDT must be rendered without the needs of a full reload of the map.

RO8. Automatic 3D building construction: (i) 3D buildings must be created automatically, to be able to scale and replicate the SCDT framework; and (ii) the used software must be released with open or free license.

In the following sections, our solution is presented and described wrt the defined requirements. Then, in Section 7 a comparison among a selection of the most diffuse SCDT solutions is reported highlighting for each solution the satisfied requirements.

3 SCDT Development phases

In order to develop a SCDT solution capable to fulfil the above requirements, we identified the following main phases:

A. **Data acquisition**: city graph, IoT sensor/actuators, POIs, orthomaps, paths, digital surface model (DSM) and terrain elevation (DTM), images, etc.

B. **Production**: Heatmaps computation, traffic flow reconstruction, OD productions, 3D building construction, etc.

C. **Integration and distribution**: acquired and produced data are integrated into a global digital twin model and rendered as 3D multi-data map and distributed as an interactive web interface.

Phases A, and B are those that mainly respond to reach the data requirements (RD1 – RD7), while phase C must be designed in order to satisfy the requirements on the interactivity (RI1 -RI8). Instead, the operative requirements are mainly addressed by the IoT platform on top of which the SCDT is built on. In our case, the Snap4City platform [6, 7] – an open-source platform developed at DISIT Lab, University of Florence (https://www.snap4city.org/) – was used. The platform includes Data Analytics processes to perform analyses and simulations (RO1), is compatible with a large number of protocols and format and with FIWARE Smart Data Model (RO2), integrates IoT Apps, based on Node-RED, for data ingestion and transformations (RO3), and, thanks to the semantically indexing of data in an RDF Knowledge Base, offers dedicated APIs to query the stored data and to model the road graph and related elements, heatmaps, OD matrices, traffic, scenarios, etc. (RO4).

The developed 3D multi-data map is an open-source web interface (RO6), created as a dashboard in the Snap4City platform. It can allow the visualization of an interactive 3D reconstruction of the city, with the possibility of showing and inspecting different

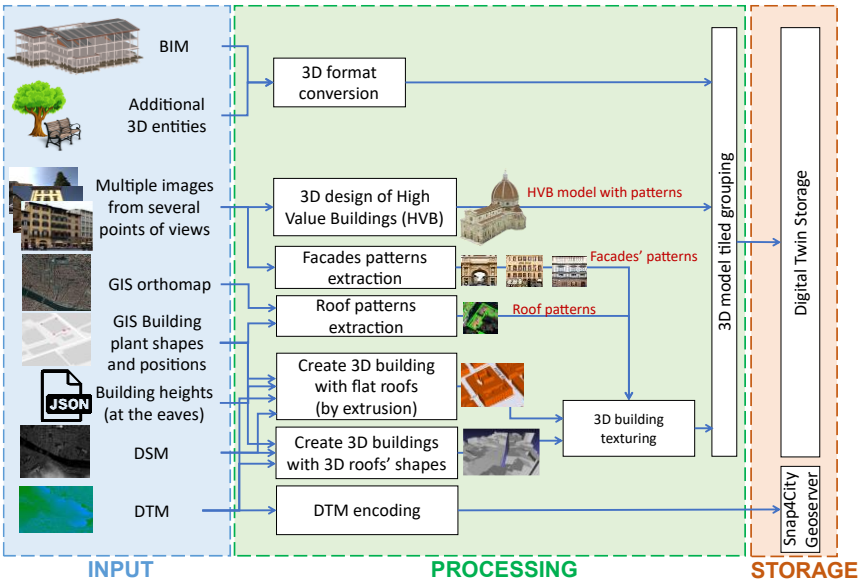


Fig. 1. Proposed production process to create the 3D structures of our Smart City Digital Twin.

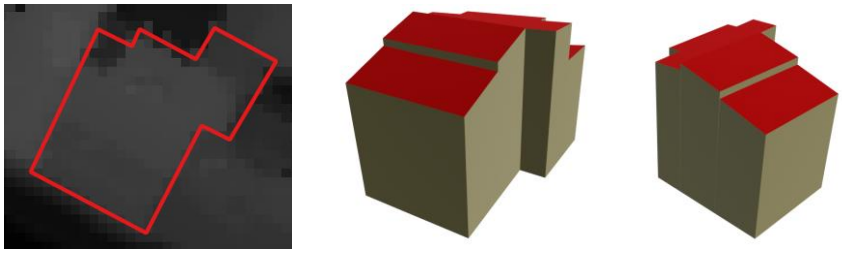
kinds of entities and related data, such as IoT devices, POI, heatmaps, geometries related to bus routes, cycling paths, traffic flows, etc. In this way, the Snap4City platform allows to exploit a complete open-source framework that can collect, process, and manage all the data needed to obtain a high-fidelity SCDT.

Indeed, phase A is realized exploiting the Snap4City capabilities to ingest, manage, retrieve heterogeneous data: ground information (RD2) and paths and geometry (RD4) from OSM and Open Data, IoT sensors data, POI and KPI (RD5) to be shown as PINs. In phase B, heatmaps (RD3) are produced from the acquired data using some Data Analytics [8, 9], and the 3D buildings (RD1) and the terrain elevation (RD6) are created following the procedure described in Section 4. Note that, the algorithm to create the 3D building structure is completely automatic and released as open source[1], satisfying RO8. Additional 3D entities (RD7) can be obtained from free archives of general 3D models and included into the map. Finally, in phase C, all the data are integrated into an interactive web interface as shown in Section 5.

4 3D Data Production

A block diagram depicting the 3D production process is reported in Figure 1. As can be see, inputs, sub-processing blocks and the final storage in the Snap4City platform are highlighted. Regarding the inputs, the production process requires street-level and

[1] Available at https://github.com/disit/3d-building-modelling



(a) (b) (c)

Fig. 2. Example of DSM modelling to obtain building models with 3D rooftops. In (a) the original DSM with superimposed the building plant shape. In (b) and (c) the obtained 3D model from two different views.

aerial (i.e., orthomaps) RGB photos to obtain roof and façade textures and to model possible High Value Buildings (HVB). Additionally, building plant shapes from OSM are used to geographically localize the buildings to be produced. Finally, building height information (in the format of GEOJson files) and DSM data are required to properly model the 3D structures, while the DTM is used to compute a terrain level and to put the building at the right elevation position. BIM and additional 3D entities are considered, possibly requiring some format conversion. Hereafter, more details on the sub-processing blocks are reported.

To obtain high quality models for the HVBs manual 3D design or automatic computer vision techniques, such as Structure from Motion, can be employed. The obtained models are then put in the right scale, position, and elevation.

Regarding the roof and façade patterns, they are respectively extracted from orthomaps and street level images. In order to obtain an accurate orthomap segmentation to extract the roof texture, a deep net was used [10] to find the similarity transformation required to locally warp the orthomaps and make them accurately fit the building plant shapes. Diversely, façade's patterns are extracted by identifying the building façade into the acquired images and then rectifying them using planar homographies.

3D structure of ordinary buildings can be obtained with two different approaches. Flat-roof buildings are obtained by extrusion from the building plant shapes using the building height. Such height attribute can be obtained from manual measurements of the eave heights, or by evaluating the average height of the DSM samples included into the building plant shape. Differently, 3D-roof buildings are obtained by analysing the DSM and fitting on its samples planar primitives to describe the different roof slopes. Such a process includes spatial clustering using region growing [11] and HDBSCAN [12], multiple line model regression [13] and finally robust plane fitting. In Figure 2, an example of input DSM and output model is provided. Both flat-roof and 3D-roof building models can be put at the right terrain elevation exploiting the information encoded in the DTM, and roof and façade texture are then applied to the 3D models using the Python Blender API.

To guarantee a fast model loading through by the browser interactive web interface we exploited a tiled approach (see Section 5). The complete city map was divided in

non-overlapping tiles and for each building we computed the tile it belongs considering the building plant centroid to uniquely assign a building to a tile. All the building 3D textured models falling into the same tile, considering HVB when available, were then collected into a single folder and saved into the Snap4City 3D Storage, following a hierarchical Z/X/Y folder organization used by most of GIS applications like OSM, QGIS, etc. Z is the zoom factor (fixed at 18 for our tiles) that describe the tile dimension, while X and Y are the tile coordinates. Note that, even if the models are grouped in tiles, each single building is represented as a separate entity in order to enable the picking functionality (RI4). Additional 3D entities (RD7) such as tree, streetlamp or other minor urban structures can be obtained from free 3D repositories and then placed into the map exploiting positioning information obtained from Open Data.

4.1 High resolution DTM encoding

To exploit the DTM as a terrain level in our interactive user interface (see Section 5), the DTM, expressed in float values, was converted to RGB format and deployed in the Snap4City Geoserver, a WebServer that mainly uses the WMS protocol over HTTP to serve through REST API calls tiled images of specific areas of wide and huge GIS maps. In order to accomplish the DTM conversion, we use the following mapping function:

$$\begin{cases} R = \left\lfloor \dfrac{100000 + 10v}{256^2} \right\rfloor \\ G = \left\lfloor \dfrac{100000 + 10v}{256} \right\rfloor - 256R \\ B = \lfloor 100000 + 10v \rfloor - 256^2 R - 256G \end{cases}$$

where v ∈ R is the DTM raw value. In this way we can obtain an RGB image able to encode elevation differences up to 0.1 m. Then, the encoded DTM is loaded into the Snap4City GeoServer to be retrieved in real-time.

5 Interactive web interface

The general architecture developed for distributing the SCDT as a 3D multi-data map dashboard in the Snap4City platform is able to distribute and reassemble all the data required by the requirements RD1-RD7: different version of 3D models of buildings (LoD1, LoD3, BIM) and additional entities, heatmaps (for traffic flow, pollutant dispersion, etc.), PINs (IOT, POI, etc.), 3D terrain from DTM, sky pattern, ground information, paths and areas. The architecture implements a client-side business logic that exploits a series of REST API calls to load the data independently on user demand (as requested by RI5). For example, the 3D tiled representations are retrieved via HTTPS protocol, as well as data for POI, IoT devices, paths, etc. obtained with specific geographic queries on the SuperService Map of Snap4City [14]. Differently, heatmaps (static or animated) and the encoded DTM are retrieved via WMS protocol over HTTPS

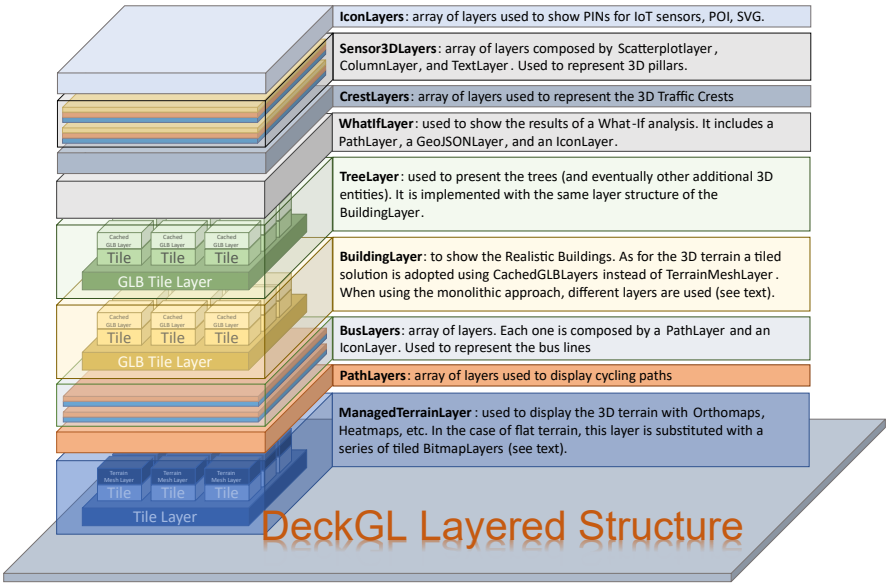


Fig. 3. Hierarchical layers structure of the interactive web interface.Mapping of the Digital Twin model on layer structure on rendering on the client.

by querying the Snap4City GeoServer, limited to the portion of the map visualized by the user. The hierarchical layered structure depicted in Figure 3 and described in this section is used to model and represent the different kinds of information provided.

The rendered solution has been implemented via layered WebGL and APIs, in order to access to the GPU thanks to the passthrough available in Web Browsers without the needs of plugin, to satisfy RO6, exploiting the open-source library Deck.gl. Deck.gl offers some default layers that we used to display 3D elements. And, in order to handle specific needs, specific layers were modified and/or completely implemented from scratch. All layers are loaded at runtime on user demand. Thanks to the multi-layer structure of Deck.gl, layers were implemented individually with their own safe context, to avoid reciprocal interferences. Every layer has its own scope, managing its own data type.

First, the Deck.gl application has been realized by using a custom implementation and management of the ViewState object, in which the geographical information for the map (such as latitude, longitude, zoom, etc.), are defined. We implemented a custom rendering system in order to add features like SkyBox – to be able to include a sky representation into the 3D map – that needs direct access to the WebGL context. The solution is able to handle two different kinds of terrains, in order to respond at the requirement RD6. The first one uses a flat model, while the second one instead exploits a three-dimensional terrain with accurate elevation (referring to them as flat or elevated). Note that, each of them uses different layers and structures, as reported in Figure

3. The terrain called ManagedTerrainLayer is displayed in tiles using the default TileLayer, with every tile loading different resources from the WMS servers using business logic call-back (RI6). The most important data source is the encoded DTM of the tile, that is used to generate the mesh of the terrain using the Martini tessellation algorithm. Note that, it possible to mix multiple DTM files, for example with different resolutions: in such a case one of the DTM has higher priority over the others. The terrain texture is instead created by merging multiple images: the base image is the Orthomap of the terrain, over which different heatmaps can be shown on user demand. This data integration forms a layer called TerrainMeshLayer. The texture merging process is carried out directly inside the GPU in order to have maximum performance. In order to handle the opacity level selected, we used the following equations to merge different texture inside the fragment shader:

$$\begin{cases} mix_\alpha = 1 - (1 - \alpha_2) * (1 - \alpha_1) \\ mix_R = \left(\frac{R_2 * \alpha_2}{mix_\alpha}\right) + \left(\frac{R_1 * \alpha_1 * (1 - \alpha_2)}{mix_\alpha}\right) \\ mix_G = \left(\frac{G_2 * \alpha_2}{mix_\alpha}\right) + \left(\frac{G_1 * \alpha_1 * (1 - \alpha_2)}{mix_\alpha}\right) \\ mix_B = \left(\frac{B_2 * \alpha_2}{mix_\alpha}\right) + \left(\frac{B_1 * \alpha_1 * (1 - \alpha_2)}{mix_\alpha}\right) \end{cases}$$

where α_i is the alpha channel of the background image ($i = 1$), and the additive image ($i = 2$) to be merged, while R_i, C_i, and B_i are the RGB channels. When the merge has to be performed using three or more images, the process is performed progressively for each pair, cumulating the next on the first couple merged.

For the flat terrain, the background Orthomaps are implemented through tiles, i.e., the TileLayer. The mesh is a flat square, and, in this case, a single bitmap is used to display an image as a texture exploiting the BitmapLayer. This method has been used to represent heatmaps, which are essentials to provide a fast access/representation to large amounts of data. In both cases, data are automatically retrieved from GeoServer. Heatmaps can be static or animated: static heatmaps are provided as single PNG images, while animated ones are provided in GIF format in multiple images and rendered sequentially with a custom delay.

For the implementation of data coming from different sources like IoT devices, trajectories, cycling paths, etc., various layers with a specific JSON mapping have been implemented. To display paths and geometries, different layers depending on the type of geometry to be displayed have been used - e.g., PathLayer for the cycling path or the BusLayer to show bus routes.

Two 3D representations of buildings are used: Extruded (i.e., LoD1 model) and Realistic (including textured LoD3 models, and HVB). The Realistic representation can be retrieved as a monolithic file or divided in different non overlapping tiles. Extruded buildings are provided as GeoJSON and loaded in GeoJSONLayer. Differently, Realistic buildings can be loaded as both SceneGraph and 3D tiles. In the case of Realistic building in glTF/GLB format the SceneGraphLayer is used. This type of integration works well to achieve impressive visualization without impacting too much on the

application performances. In the case of monolithic representation, the model is loaded and shown as it is regardless of the ViewState. Differently, in the tiled case, the CachedGLBTileLayer is used, and models are loaded taking into account the viewing position and angle. Models are loaded from the nearest to the farthest w.r.t the point of view. Note that it is possible to define a limit to the amount of tile that to be displayed in order to avoid GPU overloading. This dynamic loading of the building ensures that only the amount of resource needed to display the current scene are used, since the buildings outside the ViewState cannot be processed. This is particularly useful when dealing with Smart City composed by a huge number of buildings. Trees and additional 3D entities are shown in separate layers with a structure similar to the one used to represent the 3D buildings.

IoT devices and POIs are displayed as pickable PINs on the map. If the terrain is elevated PINs are raised according to the elevation of the terrain. When a user selects one of them, a popup is shown with the relative information (static attributes as well as real-time data, if available), satisfying the requirement RI3. Whenever the sensor provides real-time data, they can be displayed on dedicated widgets, such as time trends. A relevant feature refers to the visualization of dynamic pins (RI8). Dynamic pins allow to graphically represent sensor markers changing shape and/or colour depending on the value of the metric to which they are associated with. In this way, dynamic pins enable a fast and immersive data-driven and event-driven visualization of data coming from physical or virtual sensors. Multiple views for different types of sensors can be exploited, ranging from dynamic SVG to 3D column representations of real-time data values. When a SVG is selected for PIN visualization, the dynamic PIN is created dynamically in the backend; then it is retrieved and displayed by using the IconLayer. In the case of a 3D column representing a device attribute value, a composite layer, called Sensor3DLayer, is generated to recreate a thermometer effect: it can display the value of any sensors with a 3D cylinder shape whose height is proportional to the considered metric while the actual value is reported in textual form on top of the cylinder. The colour of the column typically represents the category of the sensors, and all these elements can be customized by the user.

The light effects have been modelled with two types of lights: an ambient light to affect all the scene, and a directional light to model the position of the sun. In order to calculate the position of the sun, the formula given by the Astronomy Answer articles about “the position of the sun” [15] was used. This process created the lights and shadows of the scene, and it is useful to simulate when a particular area is well illuminated or not.

It should be noted that, most of the features working on the 2D representation of data needed to be revised or completely modified to pass at the 3D representation environment. For instance, in 2D traffic flow representations are typically represented by coloured lines in the map that are marginally visible into the 3D representation. Therefore, in order to provide a 3D visualization of traffic flows density a new layer called CrestLayer has been developed. In this layer, traffic density is displayed as raised crests (with amplitude proportional to the computed traffic density values) following the terrain elevation. The crests are coloured using a standard colour map that can be defined by the user. Due to the nature of the problem, traffic flow polylines can be fragmented.

Therefore, a data pre-process is performed to have a smoother representation. Every crest segment is created from three points: a middle point, in which the value is the traffic density of the specific road segment, and two extrema ones, where densities are obtained considering the average of all density values of roads connected to it.

A key functionality for urban planning and management that can be offered by a SCDT is the possibility to observe the effects produced by a change in the contextual environment modelled in terms of RDF coding of the road graph. For example, to observe how vehicle routing can change due to a scenario in which an area is blocked to traffic. Such a functionality is called a What-If analysis [18]. To implement various scenarios, the platform needs to offer the user the capability of drawing different shapes. In such a way a street or multiple areas can be blocked to traffic in the road network for hours and days: which is the definition of a so called What-If scenario. Then, the results of the What-If analysis can be shown to the user together with the defined shapes using the new WhatIfLayer. For example, after having selected a city area to be traffic free, the What-If analytic provides the SCDT with novel routing possibilities that do not enter into the zones defined in the scenarios (which may be constrained according to a set of different descriptors). This can be applied to different kinds of analysis to understand the impact of these scenarios to traffic flow, possible routing approaches, pollutant diffusions, people flows, etc.

6 Case of study: Florence city, Italy

To validate the proposed SCDT solution, we selected as case of study the city of Florence in Italy. Our SCDT encompass the full Florence municipality (as shown in Figure 4) covering an approximate extension of 151 km2 (wider than the area of the Florence municipality of 102 km2). Our SCDT of Florence, freely accessible through a web interface[2] and presented in Figure 5, includes LoD1 and LoD3 3D building models, and terrain elevation.

The DSM and DTM data used in this work to model respectively the buildings and the terrain were kindly provided by the "Sistema Informativo Territoriale ed Ambientale" of Tuscany Region. They were obtained from a LiDAR survey and are composed by several tiles covering the city of Florence, with a resolution of 1 square meter. 3D building models were enhanced with roof textures obtained from orthomaps of the city of Florence. The RGB photos are tiles with a resolution of 8200x6200 pixels, with partial overlap and rough geo-localization in the EPSG 3003 (Monte Mario / Italy zone 1) coordinate system. The SCDT includes PIN indicating position of POI and IoT Sensors. Thanks to the semantic indexing of data offered by the Snap4City Knowledge Base, different PINs can be represented with specific icons according to their semantic category [16, 17]. Information associated with a specific sensor or city element can be accessed by simply clicking on the device PIN: a popup is shown to the user presenting static attributes and, when available, real-time and historical data. Moreover, heatmaps

[2] https://www.snap4city.org/dashboardSmartCity/view/Gea-Night.php?iddasboard=MzQ5OA==

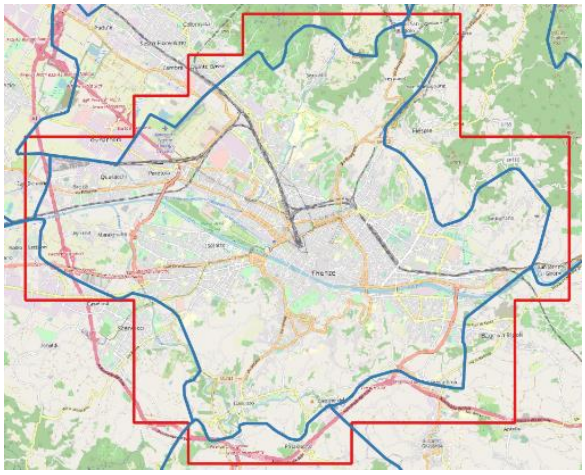


Fig. 4. Extension of the modelled Florence area. In blue the administrative border of Florence municipality; in red the area covered by our Digital Twin. This map is shown in the EPGS:3003 coordinate system.

can be loaded at user demand to show for example real-time dispersion of pollutant, and traffic reconstruction representations are shown as animated 3D crest.

In order to show the capability of our SCDT system to let the user carry out simulation and analysis on routing, controls for What-If analysis [18] were implemented into the web interface. By clicking on the map, the user can select specific points or areas to simulate a traffic restriction. Then the routing algorithm produces trajectories between any start and end position considering the defined restriction. An example of What-If on routing is presented in Figure 6: as can be seen an area was selected to ban traffic from the enclosed streets and the updated routing is shown to the user.

7 SCDT solution comparison

According to the requirements identified in Section 2, a comparison among different SCDT solutions has been carried out. To produce the comparison, we manually inspected the available information of a set of other solutions: except for [3] who has associated papers describing it, the other solutions have been studied mainly by analysing their 3D web interfaces, and, if not available (as for example [23]) by exploiting all the available information such as web pages and videos. In this comparison, we included at first CityGML [3] in order to define a baseline for SCDT solutions. Then we considered the following cases: the SCDT of the city of Helsinki, that includes LoD3 city model was implemented and made publicly available [19]. However, such a system does not provide integration with IoT data or other kinds of city related information. Another similar solution was proposed by the city of Rotterdam [20], exploiting LoD2 building models, without integrating neither any decoration elements nor elevation of terrain. A 3D model for the city of Berlin was presented in [21], providing pickable

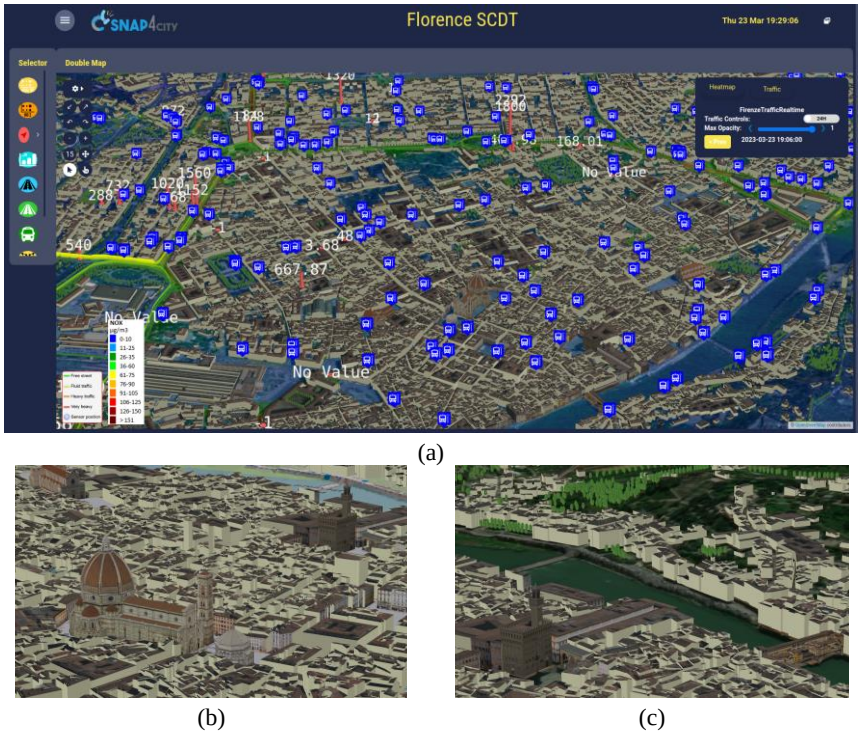
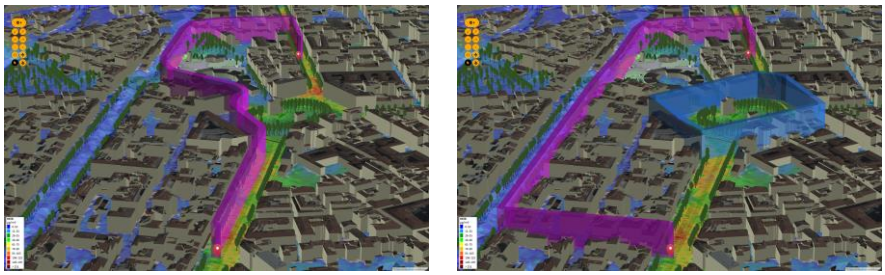


Fig. 5. Snap4City dashboard showing the Smart City Digital Twin of Florence. In (a) the dashboard is presented with LoD3 and HVB models shown together with PINs, 3D Crests for traffic, heatmaps, 3D Cylinders, trees. In (b) a close-up view of the Florence city center. In (c) another close-up showing entities correclty elevated according to the 3D terrain (textured with satellite orthomaps). To try our SCDT of Florence the reader is invited to visit the following link *https://www.snap4city.org/dashboardSmartCity/view/Gea-*

LoD2 models of buildings, supporting WMS and terrain layers. The city of Stockholm [22] implemented many aspects of the Digital Twin concept, such as POI, LoD3 buildings, either with 3D tiles and modelled ones, and others 3D entities. However, the solution lacks in the implementation of WMS heatmaps. More recently, a SCDT of Wellington was proposed [23]. Powered by the Unreal Engine, such a solution can offer LoD3 building models, paths, sensor data, terrain elevation, and additional 3D entities. However, the usage of the Unreal Engine can limit the accessibility of such a solution. Finally, in the scope of the DUET project [24], Digital Twins of Flanders, Athens, and Pilsen were produced in a project not actually accessible. Such solutions implement building models with different LoD, terrain elevation, heatmaps, and additional 3D elements. However, this solution seems to be a work in progress, with several test cases with a public web interface very limited. In Table 1, the full comparison is reported w.r.t. the previously defined requirements. As can be seen, most of the solutions are able to satisfy the data requirements (RD1 – RD7). Differently, the interactivity requirements seem to be more difficult to be respected, in particular those that are related to



(a) (b)

Fig. 6. Example of What-If analysis on vehicle routing. In (a) the intial routing, shown as a purple elevated line. In (b) a What-If scenario is enabled: a blue polygon highlight the blocked area and the updated routing is shown to the user.

the dynamic change of some models (RI4.ii, RI4.iii) and for the underground inspection (RI7). For the interactive requirements, the most advanced solution appear to be [23]: in our opinion, this is due to the fact that is the only solution powered by Unreal Engine. Indeed, such a 3D engine can offer additional functionality respect to simpler 3D engine/web player, and it requires higher computational resources, at least from server-side, since the 3D web interface (that should be based on Pixel Streaming) requires the 3D rendering be carried out on the server while the client receive only a video stream of the 3D scene. Finally, the operative requirements (RO1 – RO8) seem to be the hardest ones to be meet by all the compared approaches except for our solution since our SCDT is embedded into the IoT platform Snap4City. The platform offers business logic, data ingestion and manipulation capabilities, data analytics, etc., enhancing the geospatial data rendered in the 3D map with a plethora of additional information and services, realizing a complete Smart City Digital Twin.

8 Conclusions

In this paper, the processes used to develop a Smart City Digital Twin were described. Firstly, a series of requirements were presented and discussed. Then the development phases, guided by the previously defined requirements, were presented. Data and processes used to build the 3D structure to be included into the Digital Twin model infrastructure were described, as well as the layered structured used on client-side rendering on interactive web interface. In order to include additional information - e.g., IoT sensor data, POI locations, heatmaps, paths, etc. - our solution has been implemented into the Snap4City platform, to exploit specific API calls to retrieve the data and show them on the map on specific layer loaded on user demand. The implementation of the Digital Twin of the Florence city – publicly available at *https://www.snap4city.org/dashboardSmartCity/view/Gea-Night.php?iddasboard=MzQ5OA==* – was discussed as a case of study, showing in particular some of the most complex functionalities offered such as the possibility to perform What-If analysis on demand. Finally, a comparison

Table 1. Comparison of SCDT platforms: (*) defines only the building model, (**) functionality implemented in CESIUM but without any model placed underground, (x) use CESIUM, it could be possible, (C) based on CESIUM.

	CityGML [3]	Helsinki [19]	Rotterdam [20]	Berlin [21]	Stockholm [22]	Wellington [23]	DUET [24]	Snap4City (our)
RD1.i	Yes	No	No	No	No	No	Yes	Yes
RD1.ii	Yes (LoD3)	Yes	Yes (LoD2)	Yes (LoD2)	Yes (LoD3)	Yes (LoD3)	Yes (LoD2/LoD3)	Yes
RD1.iii	No	No	No	No	No	Proba-bly	No	Yes
RD2	No	Yes	Yes (C)	Yes (C)	Yes	Yes	Yes	Yes
RD3	No	No	No	Yes	No	Proba-bly	Yes	Yes
RD4	No	Yes (C)	Yes (C)	No (x)	Yes	Yes	No	Yes
RD5	No	No	No	No	Yes	Yes	No	Yes
RD6	Yes	Yes	No	No	Yes	Yes	Yes	Yes
RD7.i	Yes	Yes	No	No	Yes	Yes	Yes	Yes
RD7.ii	Yes	No	No	No	No	Yes	No	No
RI1	No (*)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RI2	No (*)	No	No	No	No	Yes	No	Yes
RI3	No (*)	No	Yes	No	Yes	Yes	No	Yes
RI4.i	Not clear (maybe)	Yes (s)	Yes	Yes	No	Proba-bly	No	Yes
RI4.ii	No	No	No	No	No	Proba-bly	No	No
RI4.iii	No	No	No	No	No	Yes	No	No
RI5	No	No	No	Yes	No	Yes	No	Yes
RI6	No (*)	No	No	No	Yes	No	Yes	Yes
RI7	No (*)	Yes (**)	Yes (**)	No (x)	No	No	No	No
RI8	No	No	No	No	Yes	Yes	No	Yes
RO1	No	No	No	No	No	No	No	Yes
RO2	No	No	No	No	No	No	No	Yes
RO3	No	No	No	No	No	No	No	Yes
RO4	No	No	No (not speci-fied)	No	No (not speci-fied)	No (not speci-fied)	No	Yes
RO5	No	No	No	No	No	No	No	No
RO6.i	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RO6.ii	n/a	Non-free	Free	Free	Non-free	Non-free	Non-free	Free
RO7	No	Yes	Possi-ble (x)	Possi-ble (x)	Possible	Yes	Possible	Yes
RO8.i	No	Yes	Yes	No	Yes	Yes	Not clear	Yes
RO8.ii	n/a	Non free	Non free	n/a	Non free	Non free	Not clear	Yes

with other state of the art Digital Twin solutions was carried out, showing that our ap-

proach can offer a more complete solution, considering in particular the interactive and the operative requirements as reported in Table 1.

In future works, the digital twin and its 3D representation will be further enriched with additional kind of models, and novel functionalities will be introduced in our interactive web inter-face. Moreover, the Snap4City knowledge base (Km4City) will be further expanded considering all the entities included into the Digital Twin to be able to perform semantic, relational, temporal and geographical queries to handle and retrieve all the data to be presented in our Smart City Digital Twin.

Acknowledgement

The authors would like to thank the MIUR, the University of Florence and the companies involved for co-founding the national Center on Sustainable Mobility, MOST. A thanks to the many developers on snap4city platforms. Snap4City (https://www.snap4city.org) is open technologies of DISIT Lab.

References

1. G. Mylonas, A. Kalogers, G. Kkalogeras, C. Anagnostopoulos, C. Alexakos, L. Muñoz, “Digital Twins From Smart Manufacturing to Smart Cities: A Survey”, in IEEE Access, vol. 9, pp. 143222-143249, 2021, doi: 10.1109/ACCESS.2021.3120843.
2. E. Shahat, C. T. Hyun and C. Yeom, “City Digital Twin Potentials: A Review and Research Agenda” MDPI, pp. 3, 2021.
3. G. Gröger and L. Plümer, “CityGML Interoperable semantic 3D city models,” ISPRS Journal of Photogrammetry and Remote Sensing, pp. 16-21, 2012.
4. D. Jovanovic, S. Milovanov, I. Ruskovski, M. Govedarica, D. Sladic , A. Radulovic, and V. Pajic, “Building Virtual 3D City Model for Smart Cities Applications: A Case Study on Campus Area of the University of Novi Sad,” ISPRS International Journal of Geo-Information, pp. 16-21, 2020.
5. Adreani, L., Bellini, P., Colombo, C., Fanfani, M., Nesi, P., Pantaleo, G., Pisanu, R. “Digital Twin Framework for Smart City Solutions”. In proceedings of the 28th International DMS Conference on Visualization and Visual Languages (DMSVIVA 2022), 2022.
6. Q. Han, P. Nesi, G. Pantaleo, I. Paoli, "Smart City Dashboards: Design, Development and Evaluation", Proc. of the IEEE ICHMS 2020, International Conference on Human Machine Systems, September 2020. http://ichms.dimes.unical.it/
7. C. Garau, P. Nesi, I. Paoli, M. Paolucci, P. Zamperlin, A Big Data Platform for Smart and Sustainable Cities: Environmental Monitoring case studies in Europe. Proc. of International Conference on Computational Science and its Applications, ICCSA2020. Cagliari, Italy, 1-4 July 2020. http://www.iccsa.org/ https://link.springer.com/chapter/10.1007%2F978-3-030-58820-5_30
8. Bilotta, Stefano, and Paolo Nesi. "Traffic flow reconstruction by solving indeterminacy on traffic distribution at junctions." Future Generation Computer Systems 114 (2021): 649-660.
9. S. Bilotta, E. Collini, P. Nesi and G. Pantaleo, "Short-Term Prediction of City Traffic Flow via Convolutional Deep Learning," in IEEE Access, vol. 10, pp. 113086-113099, 2022, doi: 10.1109/ACCESS.2022.3217240.

10. N. Girard, G. Charpiat and Y. Tarabalka, «Aligning and Updating Cadaster Maps with Aerial Images by Multi-task, Multi-resolution Deep Learning,» in ACCV, 2018.
11. Pal, Nikhil R; Pal, Sankar K (1993). "A review on image segmentation techniques". Pattern Recognition. 26 (9): 1277–1278. doi:10.1016/0031-3203(93)90135-J
12. Campello, R.J.G.B., Moulavi, D., Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science(), vol 7819. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_14
13. Toldo, R., Fusiello, A. (2008). Robust Multiple Structures Estimation with J-Linkage. In: Forsyth, D., Torr, P., Zisserman, A. (eds) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5302. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88682-2_41
14. Badii, C., Bellini, P., Cenni, D., Difino, A., Nesi, P., & Paolucci, M. (2017). Analysis and assessment of a knowledge based smart city architecture providing service APIs. Future Generation Computer Systems, 75, 14-29.
15. https://www.aa.quae.nl/en/reken/zonpositie.html
16. Nesi, Paolo, et al. "An integrated smart city platform." Semantic Keyword-based Search on Structured Data Sources. Springer, Cham, 2017.
17. P. Bellini, F. Bugli, P. Nesi, G. Pantaleo, M. Paolucci, I. Zaza, "Data Flow Management and Visual Analytic for Big Data Smart City/IOT", 19th IEEE Int. Conf. on Scalable Computing and Communication, IEEE SCALCOM 2019, Leicester, UK https://www.slideshare.net/paolonesi/data-flow-management-and-visual-analytic-for-big-data-smart-cityiot
18. P. Bellini, S. Bilotta, L. A. Ipsaro Palesi, P. Nesi, G. Pantaleo, "Vehicular Traffic Flow Reconstruction Analysis to Mitigate Scenarios with Large City Changes", IEEE Access, 2022, ISSN: 2169-3536. https://ieeexplore.ieee.org/document/9984661
19. Helsinki 3D city model. Available online: https://kartta.hel.fi/3d/#/
20. Rotterdam 3D. Available online: https://www.3drotterdam.nl
21. Berlin 3D, 3dcitydb. Available online: https://www.3dcitydb.org/3dcitydb-webmap/1.7/3dwebclient/index.html?title=Berlin_Demo&batchSize=1&latitude=52.517479728958044&longitude=13.411141287558161&height=534.3099172951087&heading=345.2992773976952&pitch=-44.26228062802528&roll=359.933888621294&layer_0=url%3Dhttps%253A%252F%252Fwww.3dcitydb.org%252F3dcitydb%252Ffileadmin%252Fmydata%252FBerlin_Demo%252FBerlin_Buildings_rgbTexture_ScaleFactor_0.3%252FBerlin_Buildings_rgbTexture_collada_MasterJSON.json%26name%3DBrlin_Buildings_rgbTexture%26active%3Dtrue%26spreadsheetUrl%3Dhttps%253A%252F%252Fwww.google.com%252Ffusiontables%252FDataSource%253Fdocid%253D19cuclDgIHMqrRQyBwLEztMLeGzP83IBWfEtKQA3B%2526pli%253D1%2523rows%253Aid%253D1%26cityobjectsJsonUrl%3D%26minLodPixels%3D100%26maxLodPixels%3D1.7976931348623157e%252B308%26maxSizeOfCachedTiles%3D200%26maxCountOfVisibleTiles%3D200
22. Stockholm Opencities Planner. Available online: https://eu.opencitiesplanner.bentley.com/stockholm/stockholmvaxer
23. Wellington DT: https://buildmedia.com/work/wellington-digital-twin
24. DUET Project: https://www.digitalurbantwins.com/