SoftwareX 27 (2024) 101805

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Microservices' libraries enabling server-side business logic visual programming for digital twins

Marco Fanfani, Luciano Alessandro Ipsaro Palesi, Paolo Nesi *

DISIT Lab, Department of Information Engineering, University of Florence, Via S. Marta 3 50139, Florence, Italy

ARTICLE INFO

Keywords:
Microservices
Digital twin
Server-side business logic
Node-RED, Business Intelligence

ABSTRACT

Digital twins are evermore adopted for planning activities in smart city and industrial contexts, thus requiring platforms able to handle their complexity, continuously adapting and improving data flow business logic behind to the user needs. To respond to such needs microservice architecture paradigm and business logic scripting solutions can be exploited. They should include facilities for data ingestion, transformation, visualization & event driven user interaction, formal definition of functional aspects, exploitation and management of data analytics and simulation, interoperability with external services of any kind, etc. To provide an easy and quick development tools, a large number of microservice has been formalized in a suite of new Nodes for the Node-RED framework and distributed in terms of Libraries via JS Foundation. The proposed suites of nodes (e.g., Snap4City libraries on Node-RED) are widely adopted by academic and industrial groups and fully integrated into Snap4City, an open-source platform for digital twins realization which can be used on cloud and on premise.

Metadata

Nr	Code metadata description	*Please fill in this column*
C1	Current code version	*v0.9.50 (node-red-contrib-snap4city-user)* *v0.5.17 (node-red-contrib-snap4city-developer)* *v0.0.13 (node-red-contrib-snap4city-d3-dashboard-widgets)* *v0.0.4 (node-red-contrib-snap4city-tunnel)* *v0.0.10 (node-red-contrib-snap4city-milestone)*
C2	Permanent link to code/repository used for this code version	https://flows.nodered.org/search?term=snap4city https://github.com/disit/node-red-contrib-snap4city-user https://github.com/disit/node-red-contrib-snap4city-developer https://github.com/disit/node-red-contrib-snap4city-d3-dashboard-widgets https://github.com/disit/node-red-contrib-snap4city-tunnel https://github.com/disit/node-red-contrib-snap4city-milestone
C3	Permanent link to reproducible capsule	https://github.com/disit/node-red-cauldron https://www.snap4city.org
C4	Legal code license	GNU AFFERO GENERAL PUBLIC LICENSE, Version 3.
C5	Code versioning system used	*Git*
C6	Software code languages, tools and services used	*JavaScript, HTML, CSS, Shell, Roff*
C7	Compilation requirements, operating environments and dependencies	*Node-RED, Node.js*
C8	If available, link to developer documentation/manual	https://www.snap4city.org/download/video/Snap4Tech-Development-Life-Cycle.pdf (Section IV.B.3) https://www.snap4city.org/drupal/node/593 https://www.snap4city.org/drupal/node/790
C9	Support email for questions	paolo.nesi@unifi.it

* Corresponding author.
E-mail address: paolo.nesi@unifi.it (P. Nesi).

https://doi.org/10.1016/j.softx.2024.101805
Received 24 April 2024; Received in revised form 27 May 2024; Accepted 17 June 2024
Available online 4 July 2024

1. Motivation and significance

Academic and industrial communities are posing a growing attention toward digital twins [1,2], new technologies modelling faithful replicas of real assets as single entities [3], industrial plants [4], and whole urban environments [5]. Thanks to the possibility of monitoring the status of entities via sensors, and computing predictions/simulations, digital twins are nowadays fundamental tools for smart systems addressing current and future challenges. Digital twins must integrate data from the field and keep their representations aligned with the real physical entities. Due to the multiplicity and diversity of data, flexible data ingestions are needed [6]. Moreover, digital twin status and evolution have to be computed via scalable analytic tools, and interactive dashboards are required to allows user to make changes on the digital twin and show the results [7]. Due to the complexity of the integration modular architectures based on microservices are necessary [8,9]. However, except for some basic building blocks readily available – such as storage systems, dataflow managers, or brokers – business logic approaches for data ingestion, analytics, and graphical interfaces usually require complex adaptations to be exploited on different domains. Therefore, there is the need of novel, open-source, easy-to-use solutions for developing complex scripting logics to exploit microservices for implementing specific functionalities, possibly requiring low coding skills to enable non-expert developers to compound basic functionalities for building applications, complex scenarios, business intelligence tools, and decision support systems. Indeed, the advanced level of scripting logic among microservices realizes scalable platforms that can quickly evolve and adapt to novel requirements, enabling the development of smart adaptive applications for digital twins.

Solutions for scripting logic among microservices include the Amazon Web Services (AWS) IoT ecosystem [10], and Azure IoT by Microsoft [11] where logic can be implemented in Java, Python, C#, NET, Node.js, or Python to acquire devices' data and display them on proprietary dashboards. In [12] an open-source framework for developing digital twin platform is presented. The usage of classic programming languages as Java, Python, etc., offers interfaces, function, and classes to define IoT entities to be mirrored in the digital twin. However, a remarkable competence in programming is required to extend and adapt these solutions to specific needs. Visual programming is an effective instrument for developing scripting logics, e.g., Internet of Things Applications (IoTApps) [13], and solutions emerged over the years [14–16]. Functional requirements of these solutions are mainly covered by the provided microservices which can be added to perform: data ingestion, transformation, storage load/retrieve, interaction, simulation, data analytic, machine learning, interaction with user, etc. However, several non-functional requirements have to be satisfied to enable the creation of IoTApps: performance, community support, extendibility, level of expressiveness, wide portability, open licensing. For example, Atmosphere IoT [17], Mendix [18], and VISUINO [19] are proprietary solutions with limited functionalities. Open solutions like S4A [20], miniBloq [21], Eclipse Kura [22], Wyliodrin [23] are mostly focused on embedded devices and are not adequate for deployment on-cloud. More flexible is Node-RED [24,25], an open-source framework from the JS Foundation based on Node.JS, since it is open-source, easy to use, extendable, and not only focused on embedded devices. It provides a web interface, i.e., the Flow Editor, for creating, editing, and executing event-driven data flows with visual programming by connecting nodes. A wide range of predefined nodes/functionalities are provided through the Node Library [26], and additional nodes can be created. Node-RED integrates with services via API using brokers, IoT devices, databases, and a large range of microservices/tools. Data flows in Node-RED can be triggered by external events (push messages), such as MQTT messages, HTTP listeners, etc., and access to services in pull using Rest Calls, Web Services, FTPs, etc. Quite similar to Node-RED is FLOGO [27], which provides limited functionalities for managing flows. Even if, Node-RED is a valid framework for IoTApp development; the available nodes (the basic and those provided by the community) are not sufficient for creating business logics for extensive digital twin platforms. Microservices/nodes can be easily added. Moreover, such services must be compliant with data privacy regulations, e.g., GDPR [28], and if they are not integrated with each other in a suite adopting the same authentication and authorization (A&A) mechanisms, as single sign on (SSO), are not professionally usable since the developers are constrained to continuously insert credentials for each microservice, while the duty of a framework should be to facilitate the integration. Other missing aspects of Node-RED that have been identified in the literature are a clear and punctual debugging system [29], and the semantic enrichment of data flows and data [30,31].

In this paper, to sustain the increasing interest of academical and industrial sectors toward the development of digital twins, we present the Digital Twin business logic development environment of Snap4City which is open source, and it is based on an extension of Node-RED plus Snap4City MicroService Libraries for Node-RED created to introduce novel and essential functionalities required for the development of digital twin solutions/applications. The Snap4City microservices are accessible thanks to more than 190 Node-RED nodes. The programming in terms of microservices can be regarded as a Server-Side Business Logic (SSBL), i.e., an evolution of IoTApps [32,33,34]. Snap4City provides five libraries of nodes to add basic and application levels functionalities for multiple domains: smart city, energy, mobility, industry, and smart applications in general. In simple manner, we state that SSBL, in short Proc.Logic, is defined as:

$$Proc.Logic = Node - RED + Snap4City\ MicroServices\ Libraries\ of\ nodes.$$

Additionally, a solution to add advanced runtime debugging facilities to Node-RED nodes has been proposed by extending the Cauldron approach (Capacitating Agile Users with Live Debugging Resources On Node-RED) [29] into Node-RED instances on Snap4City.

To give the reader the coverage overview of the proposed Proc.Logic, in Fig. 1, a simplified representation of the Snap4City MicroService Digital Twin platform [35] is shown, not including authentication and authorization aspects and corresponding machine-to-machine secure communications [36]. Green blocks indicate MicroServices. Interactions with the real world are performed via sensors, actuators, edge computing – following the Internet of Things (IoT)/Web of Things (WoT) paradigms – with some gateways (GW) and External Services, e.g., Intelligent Transport Systems (ITS). Push/pull protocols can be used, and processing logic can be installed into Node-RED on edge devices. Data may be directly ingested/produced through some Broker, and/or their services offered as MicroServices. Data/messages may flow into multiple storages (relational and NoSQL databases, RDF stores, documental storages as OpenSearch, etc.) directly passing from brokers to Data Ingestion HUB through some dataflow manager (e.g., Apache NiFi), or via their offered MicroServices. Data Analytic solutions and Dashboards can access to the storage as well as interact among them to provide event-driven functionalities based on MicroServices, and even directly acting on the actuators of the physical environment, via end-to-end secure connections. When needed, data on Brokers can be directly visualized/received on/from dashboards in real-time, skipping the storages to offer quick renderings. Actions performed on the visual representation of the Digital Twin can be sent back as feedback to brokers and then to actuators, closing the loop from digital to real entities. Snap4City was used to implement digital twin smart solutions in industrial [37] and urban [38,39] contexts, covering essential aspect of the digital twin concept – like interoperability, continuous updates, interactive 2D and 3D visualizations, control of actuators in the real world – by exploiting the microservices based on Node-RED. Note that, the proposed libraries were designed to be primarily integrated into the Snap4City platform. However, with some work on the source code, endpoints called by the nodes can be changed and adapted to different platforms.

The paper is organized as follows: in Section 2 the novel suite of

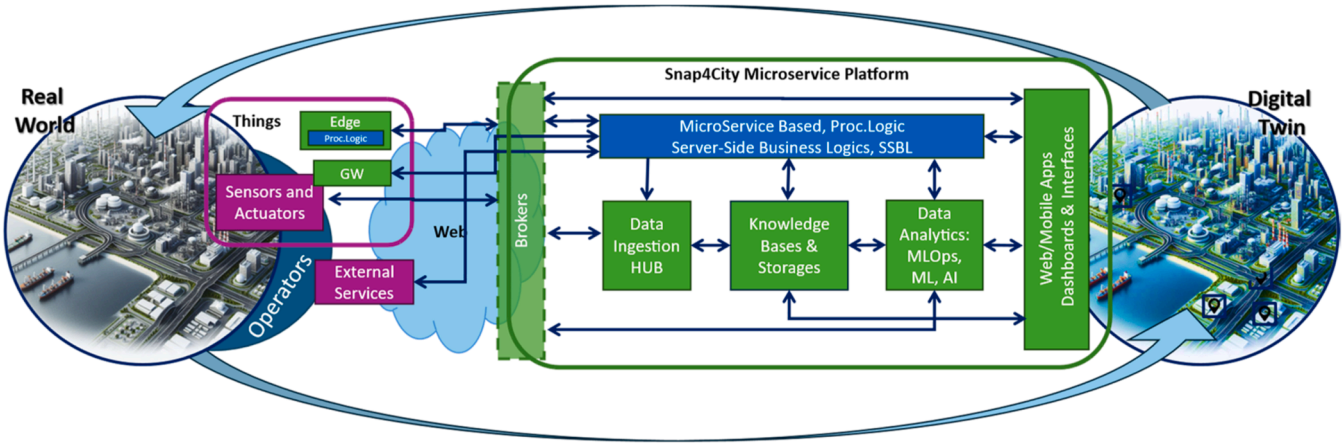

Fig. 1. Schematic representation of the Snap4City Microservice Digital Twin platform. All green modules are exploited via their exposed microservices, and the blue one allows to visual programming processing logic for scripting business logic and thus for implementing SSBL processes exploiting those microservices, defining event driven data flows, connecting digital twins with real world, and mediation with process logic SSBL.

Node-RED MicroServices is presented. Section 3 provides examples on the usage of the introduced nodes, while in Section 4 impact of presented libraries of MicroServices is discussed. Conclusions are drawn in Section 5.

2. Software description

In this section, the framework of Snap4City MicroServices of Nodes (in short Snap4City Nodes or Nodes) developed to introduce a large set of functionalities according to the Node-RED approach is presented. Note that, since the novel nodes have been developed strictly following the Node-RED approach, any developer with minimal experience in using Node-RED can easily use the novel nodes without any additional training. In Fig. 2, a graphic representation of main categories of Nodes for developing Proc.Logic data flows is shown. Grey areas represent standard Node-RED functionalities/nodes and the exploitation of third parties microservices from the open community. Green areas indicate introduced functionalities, namely: Entity Management, Visualization and Interaction, External Services Integration, Analytic Services, Platform Management. For each of them, sub-categories are discussed in Section 2.2. The introduced Nodes include A&A to access functionalities according to specific user's profile. The machine-to-machine authentication is automatically carried out exploiting a SSO schema using OpenID Connect (OAuth and access token) standard, also compatible with SAML and others. The A&A can refer to one or more Snap4City installations on public or private cloud and on premise. When Nodes are used in other contexts, authentication parameters can be specified in dedicated Node configuration panels.

The Snap4City Nodes are open-source and are accessible from the Node-RED Library as five distinct libraries, which can be installed/updated separately [40]. Nodes' codes can be obtained from NPM and from GitHub [41]. The two main libraries are: *node-red-contrib-snap4city-user* including Nodes requiring low-level programming capabilities (parameters are specified within the node's panel and most of the Nodes produce single variable outputs instead of complex JSONs); *node-red-contrib-snap4city-developer* providing advanced functionalities for expert users accepting complex JSONs to create strongly dynamic and flexible Proc.Logics. The other three libraries include: *node-red-contrib-snap4city-d3-dashboard-widgets* to introduce dashboard widgets based on D3.js [42]; *node-red-contrib-snap4city-milestone* to interoperate with Video Management Systems (VMS) of Milestone [43]; *node-red-contrib-snap4city-tunnel* to offer tunneling for remote

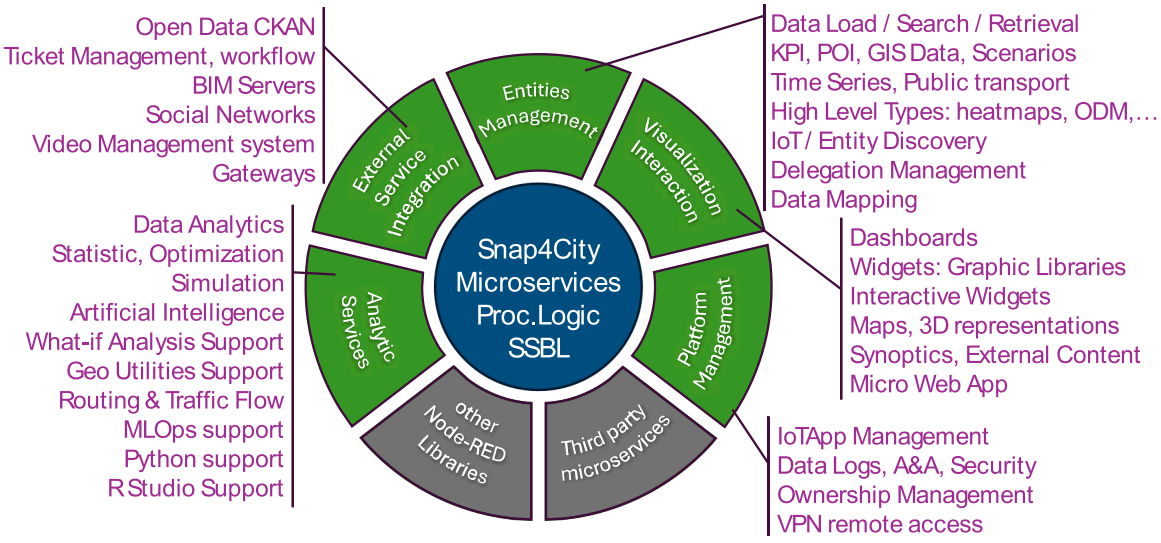

Fig. 2. Categories of microservices used to define Proc.Logics in Snap4City digital twin platform. Green blocks indicate areas of microservices functionalities added in Snap4City libraries. Grey blocks represent standard and additional functionalities from community which can be exploited by third parties. For each Green block a set of sub-categories of accessible MicroServices are listed.

management of edge including Proc.Logic. All the libraries can be installed in any Node-RED tool, both on cloud and on edge, on any operating system.

At system level, an extended debugging capability has been developed for the Node-RED editor to monitor passages of JSON messages and setting breakpoints to halt flow execution. These features have been obtained by extending [29] and are automatically deployed for Node-RED containers on Snap4City. In Fig. 3, a Proc.Logic on the Node-RED Flow Editor with the extended debugging activated is presented. The debugging offers the possibility to monitor messages entering any kind of node (not only those of Snap4City), set breakpoints to halt the flow and analyze it, proceed step-by-step, and release or delete the message queue. It is possible to observe the trend of numerical values of the last 10 messages and additional information can be visualized by expanding the debug panel of the node to allow the user to inspect or delete the history of messages received and sent, inject personalized messages, and set breakpoints. The user can specify attributes to be tracked using standard JavaScript syntax. Such extension offers an augmented development environment where programmers can obtain better understanding of the flows and the exchanged messages and be able to create more complex Proc.Logics that could be difficult to develop using the standard debug panel offered by the Node-RED editor.

Additionally, a Snap4City version of Node-RED engine and tool have been developed for Android as an accessible APK [44]. It includes Snap4City main libraries to communicate and exploit MicroServices.

2.1. Software architecture

In Fig. 4, a conceptual representation of each Snap4City nodes with respect to Node-RED, A&A, and cloud microservices is reported. Each Node is executed by Node.JS engine and supported by the above-described central debug directly enforced on the latter. In addition, Node-RED provides a webpage for visual programming, debugging and configuration. The Node input is a JSON message (msg), and the output messages of a Node can be provided via one or more channels/pins/outs. Input/output messages are exchanged as events among Nodes into a data flow according to connections defined in the visual editor. Any input message has priority over manual setup/configurations. A specific Node implementation may call API in different protocols (HTTPs, FTPs, etc.). This is the mechanism by which the Snap4City MicroServices, as well as External Services, are provided as Nodes. The A&A Node is responsible for keeping active/refreshed the A&A to handle requests toward Snap4City MicroServices, passing a valid token to the other nodes. To this end, specific A&A support is provided in front of all services and may be controlled via API management system. Among them there are APIs for Data Management, Data Analytic exploitation (via Plumber for RStudio and via Flask for Python), Platform Management, Dashboards and Synoptics setting and control, Web Socket (WS) for bidirectional message passing, Brokers and Gateways, and VPN Edge access for remote management.

2.2. Software functionalities

In Section 2.1 we stressed the scientific contribution as non-functional requirements: integrated and uniform A&A, debug and interoperability from cloud and edge. In this section, a description of the Snap4City Nodes/MicroServices is provided according to green blocks of Fig. 2.

Entity Management Nodes include functionalities to handle the data entities modeled in the digital twin platform and allows Data Load/Search/Retrieval on each of them with suitable Nodes (exploiting as MicroServices Snap4City API, knowledge base in RDF stores, OpenSearch, etc.). The data entities can be KPI (Key Performance Indicators), POI (Point of Interest), GIS (Geographic Information System) data, as maps via WMS/WFS protocols, Scenarios (describing areas with road graph and entities), Time Series, Public transport data (traffic, parking, people flow, etc.), High Level Types, HLT (heatmaps, origin destination matrices, trajectories, etc.), IoT Device and smart data models (e.g.,

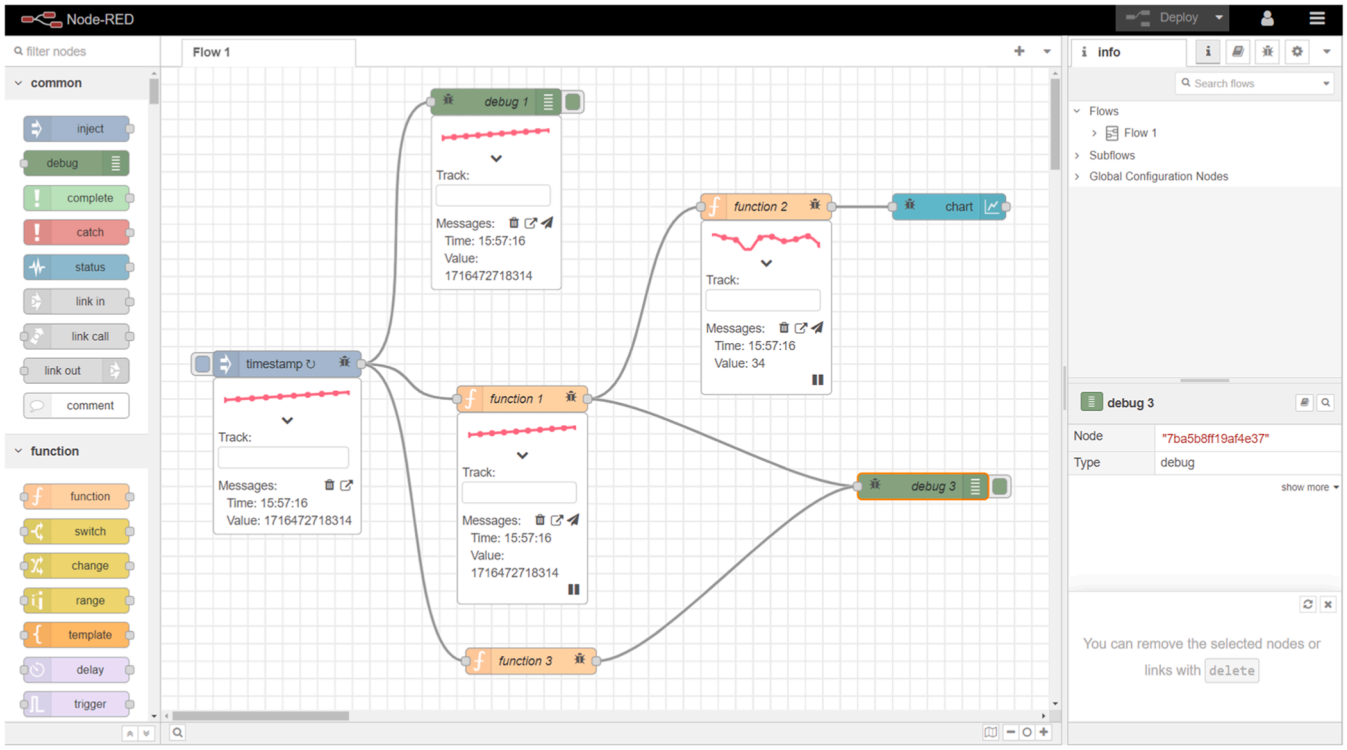

Fig. 3. Example of Proc.Logic flow in the Node-RED editor with activated the extended debugging capabilities. For each node, a panel reports values and trends of data specified by the developer. Breakpoints can be activated to halt the flow, proceed step-by-step, resume, or abort the process. In background execution after closing the editor the break points are automatically quitted.

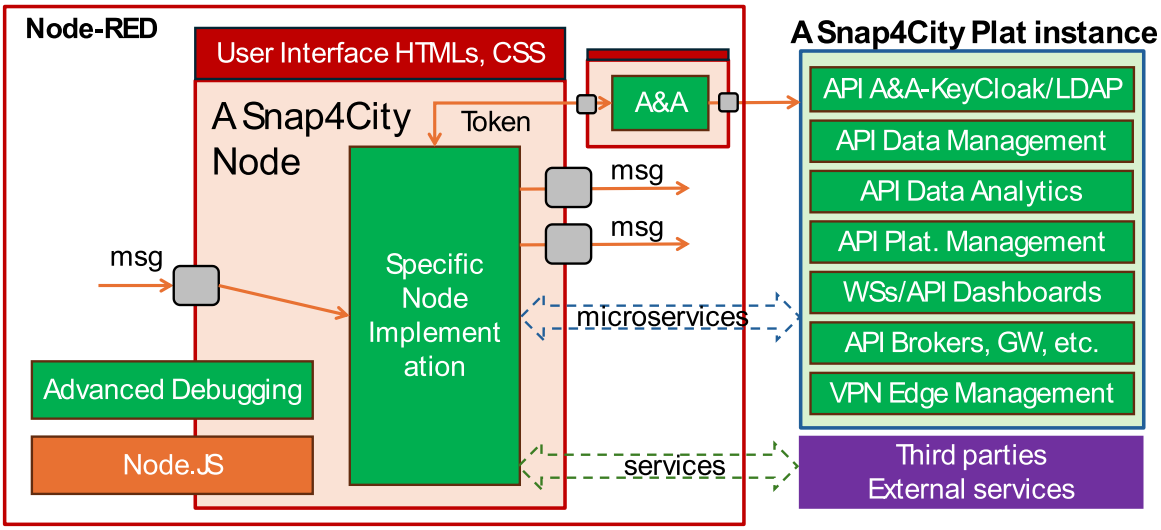

Fig. 4. Architecture of a Snap4City Node with respect to Node-RED and Snap4City platform.

FIWARE NGSI [45]) managed via IoT Directory and brokers, 3D models. The Data Entities can be mapped each other, and the Authorization module controls the access for their manipulation. The microservices allow to discover and retrieve information exploiting the API and performing spatial, temporal and relational semantic queries on knowledge base on the basis of the Km4City Ontology [46].

Visualization & Interaction Nodes are fundamental to visualize real-time and historical data and results of any entity kind specified above, data coming from historical data storage and event driven [47]. For vent driven, specific widgets and synoptics can be placed in dashboards [48]. Widget and communication with them are made available as Nodes to render and get events from the user interactions based on secure WS to set an open communication channel between the users and the Proc.Logic, thus producing secure real-time event-driven data flows, as end-to-end.

External Service Integration Nodes are used to easily integrate in the digital twin platform third party services. For example, to manage maintenance tickets and events with OpenMAINT [48]; to manage events from TV cameras and send events to the VMS of Milestone [43]; to collect or publish data from/on social networks (Facebook, Telegram, etc.); to send/receive SMSs; to manage Open-Data via CKAN API; to interoperate with GIS, BIM (Building Information Modeling), ITS, etc. There are no particular limits to the integration capabilities of services.

Analytic Services Nodes are used to call/develop: (i) ready to use analytic services such as routing, geo-utilities (GPS distance, GPS inclusion verification, geo reversing, etc.), traffic flow reconstruction, prediction models, traffic simulation, computing KPI, pollutant simulation/prediction, etc.; (ii) custom data analytic processes can be coded in Python or R Studio by developers on the platforms, and are automatically transformed in microservices on containers by Snap4City platform and engine. They can implement statistics, optimizations, simulations, predictive models, clustering, classification, What-if analysis, etc. [49]. The data analytics processes can exploit parallel architectures with dedicated hardware such a NVIDIA boards as well as any CPU/GPU solution via a distributed MLOps support.

Platform Management Nodes provide access to services Proc.Logic Management (such as restart, upgrade, etc., of processes), Ownership Management (change ownership) of processes or entities, Data Logs, general A&A, and VPN for remote access to Proc.Logic in execution on edge Node-RED installations.

In Table 1, a comparison between the proposed Snap4City Proc.Logic and the state-of-the-art solutions presented in Section 1 is shown. As can be seen, AWS and Azure cover most of the functionalities, however they are closed-source solutions and thus with difficult extendibility, and they do not provide visual programming, synoptics and have limited event driven support on dashboards. Open-source solutions show limits in particular for visualization, event driven management on dashboards, integration of external services and analytics, and platform management. Moreover, it is worth noticing that almost none of these solutions support a comprehensive system for A&A with single sign on, and GDPR support: this is a fundamental aspect of digital twin platforms to comply with data protection and privacy regulations and having to perform independent authentications and authorizations for each entity, analytic or service is tedious and time consuming. Differently the proposed solution offers a wide range of functionalities, exploit visual programming, is open-source, is compliant with data and privacy regulations using a single sign-on schema, and offer debugging capabilities.

The actual change of paradigm from traditional IoT App to Digital Twin process logic resides in the integrated paradigm of a range of microservices covering the whole area described in Fig. 2, plus the non-functional aspects of the development environment such as the possibility of maintaining aligned the physical and digital twin, and also the possibility of simulating changes on the digital side without affecting the physical, for example during the optimization and what if analysis. Thus, multiple scenarios can be explored at the same time, while the twin is maintained aligned. Moreover, digital twins, having to continuously update to reflect the real-world entities, requires more frequent maintenance activities. Snap4City Proc.Logic as an evolution of the IoTApps enforcing a large set of microservices/node and some changes on development tools: debug, MLOps, data analytics, and a more powerful interfaces, both in 2D and 3D. The flows in the Node-RED can be downloaded, saved and shared. On Snap4City is also possible to save them on a resource manager, and a daily incremental back up is performed. Moreover, each Node-RED can be set to save versioned flow on Github, under the responsibility of the single user.

3. Illustrative examples

Fig. 5 shows an example of how a Proc.Logic implements the back-office logic for managing critical events, connecting video cameras and recording events, adding events from webpages, observing the position of events, and managing the evolution of the events. In this case, the Milestone VMS (Video Management System) microservice is exploited to enable monitoring and management of data from surveillance cameras within Snap4City. VMSs are widely used to control specific areas: obtained data are directly sent to the digital twin to provide operators direct access to video feeds, and to perform simulation and computing HLT data, e.g., heatmaps representing the number of

Table 1
Comparison among state-of-the-art Proc.Logic/IoTApp development frameworks. Symbols: *Y*=yes, *N*=no, *L*=limited, *E*=via third party solution, Y/*E*= via basic plus third party.

Solution	Open Source	Extensibility	Visual Programming	On Cloud	On Edge	Community Support	Entity Management	Visualization & Interaction	External Service Integration	Analytic Services	Platform Management	Single A&A, GDPR	Debugging	Event Driven Dashboard	Event Driven Synoptics	Semantic Modeling
AWS IoT [10]	N	N	N	Y	Y	L	Y	Y	Y	Y	Y	Y	Y	Y	N	N
Azure IoT [11]	N	N	N	Y	Y	L	Y	Y	Y	Y	Y	Y	Y	Y	Y/E	Y
WLDT [12]	Y	Y	N	Y	Y	L	Y	N	L	N	N	N	N	N	N	N
Atmosphere IoT [17]	N	N	Y	Y	Y	N	Y	Y	L	Y	Y	Y	N	Y	N	N
Mendix [18]	N	Y	Y	Y	Y	L	N	L	Y/E	Y	N	N	Y	Y	N	N
VISUINO [19]	N	N	Y	N	Y	L	N	N	N	N	N	N	N	N	N	N
S4A [20]	Y	Y	Y	N	Y	L	L	N	N	N	N	N	N	N	N	N
miniBloq [21]	Y	Y	Y	N	Y	L	N	N	N	N	N	N	N	N	N	N
Eclipse Kura [22]	Y	Y	Y	L	Y	L	L	N	L	N	N	N	N	N	N	N
Wyliodrin [23]	Y	Y	Y	N	Y	Y	L	L	L	N	N	N	N	L	L	N
Traditional Node-RED [24]	Y/E	Y	Y	Y	Y	Y/E	N	L	E	E	N	N	N	L	N	N
FLOGO [27]	Y	Y	Y	Y	Y	L	L	N	L	L	N	N	N	N	N	N
Steinmetz et al., 2022 [30]	Y	Y	Y	Y	Y	Y	Y	L	L	N	N	N	N	L	L	Y
Thuluva et al., 2020 [31]	Y	Y	Y	Y	Y	Y	Y	L	L	N	N	N	N	L	L	Y
Snap4City Proc.Logic	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

presences or level of safety and security, etc. In the Proc.Logic of Fig. 5a, the yellow nodes are calling microservices of VMS. The Proc.Logic is responsible for collecting events and save them in the Snap4City storage via a broker (bright-green nodes) and to VMS [50]. In addition, images captured by the TV cameras are grabbed and saved for showing them to the operator, recording the critical case. For each fired event, a heatmap is generated and saved via Python analytics called by node *HeatmapUpdate*. Blue nodes are responsible for sending data to dashboards presenting data tables and widgets to provide interactive user interfaces to monitor and manage data and events collected by the integrated surveillance system (see Fig. 5b and 5c). The application's routing module (activated by the *selector-to-map* node on the left side of the dashboard) computes the optimal route from the current location of the rescue team to the event/target location and display it on map taking into account the traffic conditions to reach the target location.

Routing can also be requested by clicking on a point on the map to select the starting point, with the destination set to the selected event or the last event. A section of the workflow is dedicated to managing other functionalities of the dashboard. These include tasks, such as retrieving saved images from cameras and displaying them. The management also include the evolution of event state and the analysis of contextual information in the area. Without the adoption of the Snap4City microservices discussed in this paper, the implementation of a similar solution would have been extremely complex. This would require the use of numerous nodes from external services not well integrated and interoperable on security aspects, significantly increasing the complexity and cost of the system.

The reader is invited to visit www.snap4city.org to experiment with the Snap4City platform and the Proc.Logic development framework that implements all the microservices presented in the previous sections.

4. Impact

The development of Proc.Logic is a fundamental tool to handle the complexity of digital twin platforms. The proposed model providing a large set of Nodes/MicroServices can be exploited on the front-end to manage widgets showing data and results [51], and on the server-side to build complex event-driven business logic, to execute analytic processes [52], to ingest, transform, retrieve/load data [53], and to activate process executions on user request. Differently from state-of-the-art solutions, our Proc.Logic model is based on a visual programming paradigm usable by both non-expert and skilled programmers. The additional functionalities introduced with the proposed Nodes/MicroServices help to quickly code and deploy new Proc.Logic facilitating the continuous evolution of the digital twin. A list of scientific publications exploiting solutions developed with the proposed tools is available in [54].

A usability test was carried out including twenty-four expert and non-expert users (less than 20 % of them was a professional developer) [55,32]. After a brief introduction about Node-RED and Snap4City Nodes, users were asked to complete two Proc.Logic development exercises, including data retrieval, processing, and result visualization on a dashboard. Note that such task included the exploitation of the proposed novel nodes for building dashboards, retrieve data from devices modeled into the platform, call some analytic service for statistical analysis. Produced data flows were judged according to the fulfilment of given requirements: a 100 % score was assigned when fully compliant; inferior scores, proportional with satisfied requirements, were given to incomplete exercises. 87.5 % of users completed both exercises on paper, with an average score of 71.42 % and 83.92 % for the first and second test respectively. 83.33 % and 66.66 % of users also completed the implementation of the Proc.Logic for the first and second exercises respectively. In this case, a score of 85.75 % was obtained in average for the first test, and 83.42 % for the second. When users were asked how

Fig. 5. Example of an Proc.Logic for smart city domain. In (a) the process implements the business logic of the solution for managing critical events. In (b) and (c) the related dashboards.

simple the flow development was, 95.28 % of users found the development somewhat easy, and more than 56 % of them found the activity easy or very easy. Moreover, 50 % of users stated that the process for creating Proc.Logic was five times faster compared to other state-of-the-art tools they knew.

The above described Snap4City Libraries of Nodes for Node-RED have been largely adopted. This solution is integrated into the Snap4City platform as main tool for Proc.Logic development. The platform is in use in smart cities and industries mostly in Italy and Europe (see [56] for a full list), including the ISPRA JRC of the European Commission. The largest installation is multi-tenant with 19 organizations, used by more than 8000 users, and counting more than 700 Proc. Logics in operation. Snap4City has been used in several multidisciplinary applications and scenarios (e.g., [57,58]) in which Proc.Logic development has been carried out also by non-expert programmers.

Snap4City Libraries of Nodes for Node-RED have been downloaded more than 40,000 times[1] from September 2022, with the most downloaded packages being the user (27,543 downloads) and developer (10,401 downloads) libraries. The proposed tools are also used by Snap4 [59] and DAI [60], two start-ups that develops data ingestion, monitoring, and analytic solutions for several industrial realities and public administrations.

5. Conclusions

Digital twins are becoming fundamental tools for smart cities and industries, exploiting the IoT/WoT paradigm. Proc.Logic/IoTApp, implemented exploiting MicroServices, are required to script server-side business logic using functionalities for data ingestion and transformation, data analytics, interoperability, and interactive visualization. To obtain digital twin representations providing business intelligence facilities the logic counterpart must be easily developed and updated to guarantee a continuous integration of new scenarios, data, and analytics. Therefore, we developed and proposed since a number of years a suite of Nodes extending the Node-RED framework to quickly create Proc.Logic/IoTApp by visual programming. Novel functions/nodes for entity management, integration of external services, data analytic management and exploitation, visualization and user interaction, and platform management have been illustrated. Moreover, we introduced

[1] Statistics obtained from http://npm-stats.org/

extended debugging functionalities to help the development of data flows. The proposed libraries of Nodes have been largely used in several research and industrial applications/projects and well accepted from expert and non-expert users, confirming their usefulness in smart platform development.

Since digital twin technologies are evolving in multiple contexts, in order to provide solutions for easy development of Proc.Logics, additional nodes specialized to carry out particular functions will be required in the future. This is the typical activity behind this kind of complex platform which are in continuous development continuous integration model. The main observed directions of evolution are on developing: high level nodes to reduce the complexity of visual programming, simple nodes for specific protocols and services, microservices on data analytics and on visual programming of client-side business logic on dashboards to simply and making faster the development of new smart applications, and business solutions.

CRediT authorship contribution statement

Marco Fanfani: Writing – original draft, Software, Project administration, Formal analysis. **Luciano Alessandro Ipsaro Palesi:** Writing – original draft, Validation, Software, Investigation. **Paolo Nesi:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The author declare to do not have any conflict.

Data availability

No data was used for the research described in the article.

Acknowledgements

This research was financed by the European Union-—NextGenerationEU (National Sustainable Mobility Center CN00000023, Italian Ministry of University and Research Decree n. 1033— 17/06/2022, Spoke 9). A special thanks to the many users and developers working on the Snap4City platforms. Snap4City (https://www.snap4city.org), and Km4City are open technologies of DISIT Lab (https://www.disit.org).

References

[1] Xu H, Wu J, Pan Q, Guan X, Guizani M. A survey on digital twin for industrial internet of things: applications, technologies and tools. IEEE Commun Surv Tutor 2023;25(4):2569–98. https://doi.org/10.1109/COMST.2023.3297395. Fourthquarter.
[2] Lei B, Janssen P, Stoter J, Biljecki F. Challenges of urban digital twins: a systematic review and a Delphi expert survey. Autom Constr 2023;147:104716.
[3] Bondarenko Oleksiy, Fukuda Tetsugo. Development of a diesel engine's digital twin for predicting propulsion system dynamics. Energy 2020;196:117126.
[4] Xu Bin, Wang June, Wang Xinping, Liang Zhihong, Cui Liming, Liu Xiao, Ku Anthony Y. A case study of digital-twin-modelling analysis on power-plant-performance optimizations. Clean Energy September 2019;3(3):227–34. https://doi.org/10.1093/ce/zkz025.
[5] Adreani Lorenzo, Bellini Pierfrancesco, Colombo Carlo, Fanfani Marco, Nesi Paolo, Pantaleo Gianni, Pisanu Riccardo. Digital twin framework for smart city solutions. In: Proceedings of the DMSVIVA; 2022.
[6] Lei B, Janssen P, Stoter J, Biljecki F. Challenges of urban digital twins: a systematic review and a Delphi expert survey. Autom Constr 2023;147:104716. https://doi.org/10.1016/j.autcon.2022.104716.
[7] Barricelli BR, Casiraghi E, Fogli D. A survey on digital twin: definitions, characteristics, applications, and design implications. IEEE Access 2019;7:167653–71. https://doi.org/10.1109/ACCESS.2019.2953499.
[8] Ciavotta Michele, et al. A microservice-based middleware for the digital factory. Procedia Manuf 2017;11:931–8.
[9] Grübel J, et al. Outlining the Open Digital Twin Platform. In: 2023 IEEe smart world congress (SWC); 2023. p. 1–3. https://doi.org/10.1109/SWC57546.2023.10448743.
[10] Amazon.com, Inc. Amazon web services. 2024 [Computer software] URL: https://aws.amazon.com/(Accessed on April 05, 2024).
[11] Microsoft Corporation. Microsoft Azure IoT. 2024 [Computer software] URL: https://azure.microsoft.com/en-us/solutions/iot (Accessed on April 05, 2024).
[12] Picone Marco, Mamei Marco, Zambonelli Franco. WLDT: a general purpose library to build IoT digital twins. SoftwareX 2021;13:100661.
[13] Ali Zainab H, Ali Hesham A, Badawy Mahmoud M. Internet of Things (IoT): definitions, challenges and recent research directions. Int J Comput Appl 2015;128(1):37–47.
[14] Ray Partha Pratim. A survey on visual programming languages in internet of things. Sci Program 2017;2017:1231430. https://doi.org/10.1155/2017/1231430. Article ID6 pages.
[15] Ihirwe Felicien, Ruscio Davide Di, Mazzini Silvia, Pierini Pierluigi, Pierantonio Alfonso. Low-code engineering for internet of things: a state of research. In: Proceedings of the 23rd ACM/IEEE International conference on model driven engineering languages and systems: companion proceedings (MODELS '20). New York, NY, USA: Association for computing machinery; 2020. p. 1–8. https://doi.org/10.1145/3417990.3420208. Article 74.
[16] Silva M, Dias JP, Restivo A, Ferreira HS. A review on visual programming for distributed computation in IoT. In: Paszynski M, Kranzlmüller D, Krzhizhanovskaya VV, Dongarra JJ, Sloot PM, editors. Computational science – iccs 2021. Cham: Springer; 2021. https://doi.org/10.1007/978-3-030-77970-2_34. ICCS 2021. Lecture Notes in Computer Science( ), vol 12745.
[17] Atmosphere IoT Corp. Atmosphere IoT platform. 2024. (Version 1.5.5) [Computer software] URL: https://atmosphereiot.com/(Accessed on April 05, 2024).
[18] Mendix Technology BV. Mendix. 2024 [Computer software] URL: https://www.mendix.com/building-iot-applications/(Accessed on April 05, 2024).
[19] Visuino.com. VISUINO. 2024 [Computer software] URL: https://www.visuino.com/(Accessed on April 05, 2024).
[20] M. Conde, V. Casado, J. Güell, J. Garcìa, B. Romagosa, J. Delgado. Scratch for Android. 2024. (Version 1.6) [Computer software] URL: https://s4a.cat/(Accessed on April 05, 2024).
[21] J. Pizarro, R. Cossovich, A. Kharsansky, F. Lanza, D. Vilaseca, A. Lawrance. miniBloq. 2024. (Version 0.83) [Computer software] URL: https://blog.minibloq.org/(Accessed on April 05, 2024).
[22] Eclipse Foundation AISBL. Kura. 2024 (Version 5.4.0) [Computer software] URL: https://eclipse.dev/kura/(Accessed on April 05, 2024).
[23] Wyliodrin.com. Wyliodrin STUDIO. 2024 (Version 2.3.2) [Computer software] URL: https://wyliodrin.studio/(Accessed on April 05, 2024).
[24] OpenJS Foundation. Node-RED. 2024. (Version 3.1.8) [Computer software] URL: https://nodered.org/(Accessed on April 05, 2024).
[25] OpenJS Foundation. Node-RED tutorial. 2024. [Computer software] URL: https://nodered.org/docs/tutorials/(Accessed on May 21, 2024).
[26] OpenJS Foundation. Node-RED Library. 2024 URL: https://flows.nodered.org/ (Accessed on April 05, 2024).
[27] Cloud Software Group, Inc. Project FLOGO. 2024.
[28] European Parliament and Council of the European Union. "European general data protection regulation (GDPR)." 2016. URL: https://gdpr.eu/(Accessed on May 21, 2024).
[29] Diogo Torres. Node-RED cauldron tool. 2024. (Version 0.1-alpha) [Computer software] URL: https://github.com/SIGNEXT/node-red-cauldron (Accessed on April 05, 2024).
[30] Steinmetz C, Schroeder GN, Sulak A, Tuna K, Binotto A, Rettberg A, Pereira CE. A methodology for creating semantic digital twin models supported by knowledge graphs. In: 2022 IEEE 27th International conference on emerging technologies and factory automation (ETFA). IEEE; 2022. p. 1–7.
[31] Thuluva AS, Anicic D, Rudolph S, Adikari M. Semantic Node-RED for rapid development of interoperable industrial IoT applications. Semant Web 2020;11(6):949–75.
[32] Badii C, Bellini P, Difino A, Nesi P, Pantaleo G, Paolucci M. Microservices suite for smart city applications. Sensors 2019;19(21):4798.
[33] Udoh IS, Kotonya G. Developing IoT applications: challenges and frameworks. IET Cyber-Physic Syst 2018;3(2):65–72.
[34] Asghari P, Rahmani AM, Javadi HHS. Internet of Things applications: a systematic review. Comput Netw 2019;148:241–61.
[35] Snap4City of DISIT Lab. 2024. [Computer software] URL: https://www.snap4city.org of https://www.disit.org (Accessed on April 05, 2024).
[36] C. Badii, P. Bellini, A. Difino, P. Nesi, "Smart City IoT platform respecting GDPR privacy and security aspects", IEEE Access, 2020. 10.1109/ACCESS.2020.2968741.
[37] Bellini Pierfrancesco, et al. High level control of chemical plant by industry 4.0 solutions. J Ind Inf Integr 2022;26:100276.
[38] Adreani L, Bellini P, Colombo C, Fanfani M, Nesi P, Pantaleo G, Pisanu R. Implementing integrated digital twin modelling and representation into the Snap4City platform for smart city solutions. Multimed Tools Appl. 2023. p. 1–26.
[39] DISIT Lab. Snap4City Smart city digital twin of florence. 2024 [Computer software] URL: https://digitaltwin.snap4city.org/(Accessed on April 05, 2024).
[40] DISIT Lab. MicroServices/Nodes. [Computer software] URL: https://flows.nodered.org/search?term=snap4city&type=node&type=flow&type=collection (Accessed on April 05, 2024).
[41] DISIT Lab. DISIT Lab GitHub page. 2024. URL: https://github.com/disit (Accessed on April 05, 2024).
[42] M. Bostock and Observable Inc. D3.js library. 2024. [Computer software] URL: https://d3js.org/(Accessed on April 05, 2024).

[43] VMS Milestone Systems A/S. 2024. [Web page] URL: https://www.milestonesys.com/(Accessed on April 05, 2024).
[44] DISIT Lab. Snap4All, a mobile app for Android including Node-RED and Snap4City LIBRARIES, 2023. [Computer software] URL: https://www.snap4city.org/824 (Accessed on April 05, 2024).
[45] J.M. Cantera Fonseca, F.G. Márquez, T. Jacobs. FIWARE NGSIv2 (Next generation service interface, version 2) specification. 2024. (Version 2.0) [API specification] URL: https://fiware.github.io/specifications/ngsiv2/stable/(Accessed on April 05, 2024).
[46] Bellini P, Benigni M, Billero R, Nesi P, Rauch N. Km4City ontology building vs data harvesting and cleaning for smart-city services. Internat J Visual Lang Computi 2014. https://doi.org/10.1016/j.jvlc.2014.10.023. Elsevier.
[47] Bellini P, Fanfani M, Nesi P, Pantaleo G. Snap4City dashboard manager: a tool for creating and distributing complex and interactive dashboards with no or low coding. SoftwareX 2024;26:101729. https://doi.org/10.1016/j.softx.2024.101729. ISSN 2352-7110.
[48] Tecnoteca Srl. openMAINT. 2024 (Version 2.3) [Computer software] URL: https://www.openmaint.org/en/home (Accessed on April 05, 2024).
[49] Adreani L, Bellini P, Fanfani M, Nesi P, Pantaleo G. Design and develop of a smart city digital twin with 3d representation and user interface for what-if analysis. In: Proceedings of the international conference on computational science and its applications, Athens, Greece, 3–6 July 2023. Cham, Switzerland: Springer Nature; 2023. p. 531–48.
[50] Milestone Systems A/S. XProtect platform. 2024 [Computer software] URL: https://www.milestonesys.com/products/software/xprotect/(Accessed on April 05, 2024).
[51] Collini E, Palesi LAI, Nesi P, Pantaleo G, Zhao W. Flexible thermal camera solution for Smart city people detection and counting. Multimed Tools Appl 2024;83(7):20457–85.
[52] Bellini P, Cenni D, Palesi LAI, Nesi P, Pantaleo G. A deep learning approach for short term prediction of industrial plant working status. In: 2021 IEEE seventh international conference on big data computing service and applications (BigDataService). IEEE; 2021. p. 9–16.
[53] Bellini P, Palesi LAI, Giovannoni A, Nesi P. Managing complexity of data models and performance in broker-based internet/web of things architectures. Internet Things 2023:100834.
[54] DISIT Lab. Scientific publication list from DISIT Lab. 2024 [Web page] URL: https://www.snap4city.org/426 (Accessed on April 05, 2024).
[55] P. Nesi. Snap4City training course. 2024. [Web page] URL: https://www.snap4city.org/577 (Accessed on April 05, 2024).
[56] P. Nesi. List of Snap4City registered instance installations. 2024. [Web page] URL: https://www.snap4city.org/661 (Accessed on April 05, 2024).
[57] Alberti F, et al. Mobile mapping to support an integrated transport-territory modelling approach. The international archives of the photogrammetry. Remote Sens Spatial Informat Sci 2023;48:1–7.
[58] P. Nesi. List of Snap4City scenarios in which has been adopted. 2024. [Web page] URL: https://www.snap4city.org/4 (Accessed on April 05, 2024).
[59] Snap4 s.r.l. 2024. [Web page] URL: https://www.snap4.eu/(Accessed on April 05, 2024).
[60] DAI – data analytics insights S.r.l. 2024. [Web page] URL: https://www.d-ai.eu/(Accessed on April 18, 2024).