A Scalable and Modular Open-Source Framework for Urban Digital Twins

Jakob Tanz[1]
[1]Geodetic Institute and Chair of Computing in Civil Engineering and GIS, RWTH Aachen University
E-mail(s): jakob.tanz@gia.rwth-aachen.de

Abstract: Digital twins are becoming increasingly important in urban planning, design, and simulation, resulting in a growing number of heterogeneous and often proprietary implementations for data processing, sensor integration, and visualization. This fragmentation poses significant challenges for interoperability, scalability, and long-term maintainability. To address these issues, this paper proposes a modular urban digital twin infrastructure that emphasizes the use of open-source technologies and open standards. The proposed architecture provides a unified framework for integrating geospatial, building and sensor-based urban data into a coherent digital representation of the city. Particular focus is placed on automated data processing pipelines, such as downloading CityGML building datasets and converting them into efficient “3D Tiles” tile sets for web-based visualization. Furthermore, the paper presents approaches for storing and managing urban representations, sensor streams, and related metadata. At the core of the infrastructure is a CesiumJS-based urban digital twin platform that enables interactive visualization and exploration of urban environments. The presented approach aims to reduce technological fragmentation while supporting extensibility, transparency, and interoperability in future smart city applications.

Keywords: Urban Digital Twin, Smart City, CesiumJS, Data Pipelines, 3D Tiles



Published in the conference proceedings of the 37. Forum Bauinformatik 2026, Rostock, Germany, DOI: https://doi.org/10.18453/rosdok_id00005855
© 2026 The copyright for this article lies with the authors. Use is permitted under a Creative Commons license Attribution 4.0 International.

1 Introduction

With growing interest in modernizing urban planning and design, urban digital twins have attracted increasing attention [1], [2]. Today, numerous frameworks and technologies exist, both proprietary and open-source, to support the development of urban digital twins [3]. However, these tools, technologies, and frameworks can also present challenges during adoption due to their heterogeneous, tightly integrated, and proprietary nature [3]. These challenges include integrating all-in-one urban digital twin solutions into existing digital infrastructures, replacing or extending individual components without restructuring the entire system, and ensuring long-term interoperability across evolving urban data standards [4]. Furthermore, proprietary platforms introduce vendor lock-in, which constrains both the portability of urban datasets and introduces additional deployment costs [3].
At the same time, municipalities are increasingly publishing high-quality urban datasets under open

licenses, creating new opportunities for cost-effective and standards-compliant digital twin development. Among these are Level of Detail 2 (LoD2) building models distributed in the CityGML format, which encode not only three-dimensional building geometry but also semantic information in a machine-readable representation [5]. Leveraging this open-data approach effectively requires autonomous data ingestion, conversion, storage, and processing pipelines.

Beyond static building geometry, a comprehensive urban digital twin should also integrate dynamic data streams from sensors embedded within the urban environment [4]. The Open Geospatial Consortium (OGC) SensorThings Application Programming Interface (API) has emerged as an open standard for exposing Internet of Things (IoT) observations and sensor metadata in an interoperable manner [6]. While separating sensor data from the 3D environment and presenting it within a dashboard can be useful, the value of intelligently integrating and visualizing sensor information directly within the 3D scene should not be underestimated [1], [7].

This paper outlines an approach that combines open-source components to achieve a highly modular urban digital twin implementation. Section 3 provides an overview of the selected components, the data-processing pipeline used to integrate semi-static 3D building models and live sensor data, and the resulting Next.js-based visualization.

2 State of the Art

Existing urban digital twin solutions range from commercial platforms such as ArcGIS or OpenCities Planner to open-source initiatives such as Snap4City and Masterportal-based applications [3], [8]. These platforms typically integrate geospatial data, 3D city models, and live sensor data into a unified environment [2], [4]. To support interoperability between different systems, several open standards like CityGML or the SensorThings API have emerged [5], [6]. For web-based visualization of large-scale 3D environments, the 3D Tiles specification has become widely adopted [9]. Despite this rate of adoption, support for the current `3DTiles1.1` standard is limited across different geo-visualization technologies, with CesiumJS providing one of the few supporting open-source viewers [10]. Among open-source urban digital twin solutions, Snap4City offers a comprehensive framework for urban data integration, analytics, and visualization [11]. In contrast, Masterportal primarily focuses on web-based visualization and integration of geospatial services [12]. While these platforms provide valuable functionality, they either rely on tightly integrated software ecosystems or emphasize visualization over data ownership and extensibility.

3 System Architecture

Five main components are used to deploy the urban digital twin. The system consists of a data storage layer for 3D city models, a standard-compliant sensor data API, an orchestration framework for automated data-processing workflows, a visualization dashboard for sensor analytics, and a web-based frontend for interactive exploration of the digital twin.

3DCityDB is used to provide persistent data storage and management for all internal data used by the digital twin [13]. This includes the 3D building models, semantic building metadata, sensor data, and information about current and historical data-processing pipelines managed by the orchestration

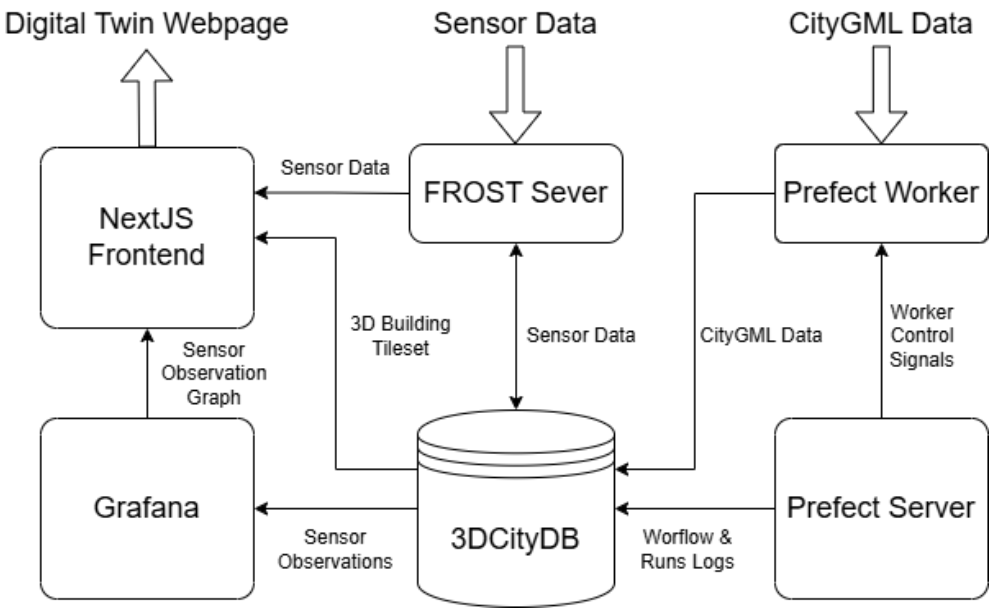


Figure 1: System architecture of the developed urban digital twin, showing the interaction between the Next.js frontend, FROST Server, 3DCityDB, Grafana, and Prefect-based data-processing workflows.

framework. This component was chosen since it is built upon the widely used and mature database management system PostgreSQL/PostGIS and provides easy-to-use tools for importing and exporting CityGML data [13].

Sensor observations are managed through FROST Server, an open-source implementation of the OGC SensorThings API standard [6]. By accepting and exposing sensor measurements and metadata through a standardized interface, the platform enables interoperable access to IoT devices and observation sources. This improves compatibility with existing tools and simplifies the integration of additional sensor systems in the future. FROST Server connects to the previously mentioned 3DCityDB database to store and retrieve sensor metadata and observations.

Data processing is performed through deployed Extract, Transform, Load (ETL) pipelines using Prefect [14]. It orchestrates Python scripts as recurring workflow processes across one or more worker nodes. Section 3.1 discusses two developed pipelines:

1. A pipeline that extracts 3D building models for a given bounding box from the North Rhine-Westphalia (NRW) OGC “3DG” features API in CityGML format.

2. A pipeline that converts the imported 3D building data from CityGML to 3D Tiles, as this is the native format for rendering 3D objects in CesiumJS, the library used by the frontend to display interactive 3D globes and maps within the web application [15].

The frontend of the urban digital twin is implemented as a Next.js web application using Resium, a React wrapper for CesiumJS. It visualizes all building models and sensors stored in the database. Grafana, a web application for creating data-visualization dashboards, is used to display live and historical observation values for each sensor [16]. These visualizations are embedded into the urban digital twin using inline frames.

All used components are open-source and communicate through well-defined interfaces and open standards. This allows individual system components to be replaced or extended without requiring significant modifications to the overall architecture. Figure 1 shows the overall component architecture and how these components interact.

3.1 Data Processing

To integrate CityGML building models into the frontend, these models must first be retrieved and loaded into the database. As previously outlined, data-processing pipelines are implemented as Prefect workflows.

The workflow begins by requesting building data from the NRW 3D geodata service, which exposes building models through an OGC API - Features interface. The area of interest can either be defined by a bounding box or by specifying a city name, and data is retrieved in multiple pages to support datasets of any size. Since the NRW dataset provides building heights relative to a geoid-based height reference while CesiumJS operates using ellipsoidal heights based on the World Geodetic System 1984 (WGS84) reference ellipsoid, every coordinate in the retrieved CityGML files is converted from `EPSG:4326+5773` to `EPSG:4979` using the Python library pyproj before import, to avoid incorrectly positioned geometries.

The transformed CityGML files are imported into the 3DCityDB instance using the CityDB-Tool, deployed automatically as a Docker container by the workflow. The tool appends new objects while overwriting those with identical identifiers, allowing the database to be expanded incrementally without duplicating existing data, as the NRW CityGML API supports queries based on modification dates.

A second Prefect workflow generates 3D Tiles directly from the database using the open-source tool pg2b3dm, which reads the geometric and semantic building information from the 3DCityDB tables and converts it into an optimized tileset without requiring intermediate conversion formats, optionally including additional semantic attributes [17]. Implicit tiling is used to improve scalability for larger urban areas: this 3D Tiles extension generates tiles using quadtrees and octrees instead of explicitly listing all tiles in the `tileset.json` file, allowing faster tile access without parsing explicit tile metadata [18]. After generation, the workflow validates the resulting `tileset.json` file and all referenced content files, and stores the tileset in the frontend's data directory for direct loading by the CesiumJS viewer.

When deployed, the first workflow can be executed on a schedule, while the second workflow is triggered through Prefect's event system whenever new data has been imported into the database. This ensures that both the database and the generated 3D city model remain synchronized and up to date without requiring manual intervention.

3.2 Visualization Component

To visualize the 3D city, a React viewer component is developed. The viewer encapsulates the CesiumJS viewer, a scene component, and the User Interface (UI), which provides controls for opening a settings panel and switching between windowed and full-screen views. This segmentation enables a clear separation of concerns when developing the urban digital twin frontend. The viewer instantiates all other components with their respective default settings, defines the overall layout of the viewer and its UI, and handles communication between child components.



Figure 2: The developed viewer showing the generated 3D building tileset and one active sensor pin.

The scene component contains all layers used by the urban digital twin. In this contribution, two developed layers are discussed in greater detail:

1. The 3D building layer displays the previously generated 3D Tiles tileset containing LoD2 building models. It also applies dynamic styling to individual buildings, enabling them to be highlighted based on their registered usage or specific building parts using the 3D Tiles Styling language.
2. The sensor layer displays all registered sensors as pins.

While the first layer is comparatively simple, as most of the required processing is performed during tileset generation, the second layer requires additional logic and UI components. It automatically retrieves all sensor metadata from the SensorThings API and places a pin at each location associated with at least one datastream, grouping datastreams within 5 meters of each other to avoid overlap. Each pin's icon indicates the sensor type, matched against a static configuration file, while its color indicates whether the sensor is currently transmitting live data; the layer periodically refreshes this process without requiring a page reload.
Clicking on a pin opens a modal, or a carousel of modals for grouped datastreams, built using open-source shadcn/ui components that adapt responsively to different devices. These embed the corresponding observation graph generated by Grafana as an iframe. Figure 2 shows the viewer component with the building and sensor layers.

4 Evaluation

The architecture is evaluated against four criteria, chosen to reflect recurring limitations of existing open-source urban digital twin platforms discussed in Section 2: *modularity* (whether individual components can be replaced or reused independently through well-defined interfaces), *support for open standards* (whether the platform relies on established geospatial and sensor data standards

rather than proprietary formats), *interoperability* (whether the platform's components can be integrated into other software ecosystems), and *data ownership* (whether the operator retains control over stored data and processing, rather than depending on external services). Each criterion is discussed below in comparison to related platforms.

Modularity. A key strength of the proposed architecture is its modular design. All major components communicate through well-defined interfaces and can therefore be replaced independently. For example, while 3DCityDB is used in the presented implementation, any PostgreSQL/PostGIS-based data source containing equivalent geometric and semantic information could be used by either adding the 3DCityDB schema to it or migrating the data to a new 3DCityDB instance. Likewise, the visualization frontend is implemented as a reusable Next.js component and can therefore be integrated into other Next.js applications with minimal effort. Since Resium is built upon React, adapting the viewer to other React-based applications would also require only minor modifications. This level of flexibility is often not available in more tightly integrated digital twin platforms, since they are designed as standalone products [3]. Because of the deployed Prefect instance and developed data pipelines, integrating 3D data into the digital twin requires only a single interaction via the Prefect web UI.

Data ownership and control. Compared to solutions such as Masterportal, the proposed architecture additionally provides full control over the underlying database and processing pipelines. While Masterportal offers powerful visualization capabilities for geospatial data, it focuses primarily on data presentation and integration of external services [12]. The presented architecture extends this approach by maintaining ownership of all imported city and sensor data. This enables advanced use cases such as custom analytics, simulation workflows, machine learning applications, and AI-assisted processing directly on the stored datasets.

Interoperability and scope. The architecture also shares similarities with platforms such as Snap4City, which combine urban data integration, IoT platforms, and visualization capabilities [11]. However, Snap4City provides a comprehensive all-in-one framework consisting of numerous tightly integrated services. In contrast, the architecture presented in this paper deliberately favors a lightweight and component-oriented approach, at the cost of the breadth of built-in functionality that a platform such as Snap4City offers out of the box.

Support for open standards. The presented architecture relies exclusively on established open standards for representing and exchanging urban data: 3D building models are stored and exchanged as CityGML [5], sensor observations and metadata are exposed through the OGC SensorThings API via FROST Server [6], and web-based 3D visualization is based on the OGC 3D Tiles specification [18]. No proprietary data or interface formats are introduced at any stage of the pipeline. Masterportal similarly builds on OGC service standards such as WMS and WFS for map integration [12], but does not natively expose a standardized sensor API or a persistent, standards-based storage layer of its own. Snap4City supports a broad range of IoT and geospatial protocols but combines them within a proprietary platform-specific data model [11], which limits the extent to which stored data remains standards-compliant outside the platform itself.

This comparison is qualitative rather than benchmark-based: no quantitative performance metrics or user studies were conducted as part of this work. The assessment instead reflects an architectural analysis of how the presented system and the compared platforms are designed and documented.

5 Conclusion

Existing open-source urban digital twin platforms tend to favor either tight, feature-rich integration, as in Snap4City, or flexible but presentation-focused tooling, as in Masterportal, and neither is designed around giving the operator full ownership of the underlying data and processing pipeline. We address this gap by combining 3DCityDB, a Prefect-based data pipeline, and a reusable Resium/Next.js visualization component into a lightweight, component-oriented architecture that keeps all imported city and sensor data under the operator's control while remaining modular and standards-compliant. The developed prototype demonstrates that a functional urban digital twin can be realized entirely using open-source and standards-compliant technologies while maintaining a high degree of modularity, interoperability, and data ownership. The architecture successfully integrates semi-static 3D city models and dynamic sensor observations within a web environment, while remaining flexible enough to support future extensions. The architecture and software components presented in this work are currently being further developed. Future work includes extending the digital environment with additional urban features such as 3D foliage models and integrating simulation capabilities, for example pedestrian movement simulations informed by sensor observations. Furthermore, the public release of the implementation as open-source software is planned once the platform has reached a sufficient level of maturity and documentation.

Acknowledgements

This work was funded by the Bundesministerium für Verkehr (BMV) under the Bundesamt für Logistik und Mobilität within the "Modellierung urbaner Szenarien" (MODUS) project - VBFU2501A.

References

[1] B. Ketzler, V. Naserentin, F. Latino, C. Zangelidis, L. Thuvander, and A. Logg, "Digital twins for cities: A state of the art review", *Built Environment*, vol. 46, pp. 547–573, Dec. 2020. DOI: 10.2148/benv.46.4.547.

[2] A. Huzzat, A. Anpalagan, A. S. Khwaja, I. Woungang, A. A. Alnoman, and A. S. Pillai, "A comprehensive review of digital twin technologies in smart cities", *Digital Engineering*, vol. 4, p. 100 040, Mar. 1, 2025, ISSN: 2950-550X. DOI: 10.1016/j.dte.2025.100040.

[3] J. B. Hoff, "An exploratory comparison analysis of smart city digital twin development protocols: Open-source versus proprietary development pipelines", Mar. 3, 2025.

[4] V. V. Lehtola, M. Koeva, S. O. Elberink, *et al.*, "Digital twin of a city: Review of technology serving city needs", *International Journal of Applied Earth Observation and Geoinformation*, vol. 114, p. 102 915, Nov. 1, 2022, ISSN: 1569-8432. DOI: 10.1016/j.jag.2022.102915.

[5] G. Gröger, T. H. Kolbe, C. Nagel, and K.-H. Häfele, "OGC city geography markup language (CityGML) encoding standard", 2012.

[6] S. Liang, T. Khalafbeigi, H. v. D. Schaaf, *et al.*, *OGC SensorThings API part 1 : Sensing version 1.1*, Aug. 2021.

[7] T. Deng, K. Zhang, and Z.-J. Shen, “A systematic review of a digital twin city: A new pattern of urban governance toward smart cities”, *Journal of Management Science and Engineering*, vol. 6, no. 2, pp. 125–134, Jun. 1, 2021, ISSN: 2096-2320. DOI: 10.1016/j.jmse.2021.03.003.

[8] R. Rajora, A. Rajora, B. Sharma, P. Aggarwal, and S. Thapliyal, “Unveiling the synergy: IoT and digital twins in transforming industries”, in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, Dec. 2023, pp. 1–5. DOI: 10.1109/SMARTGENCON60755.2023.10442175.

[9] L. Adreani, P. Bellini, C. Colombo, *et al.*, *Digital Twin Framework for Smart City Solutions*. Jun. 29, 2022, 1 p. DOI: 10.18293/DMSVIVA2022-012.

[10] Z. Usta, A. T. Akın, K. A. Ohori, and J. Stoter, “Visualization of urban digital twins on the web with attribute-driven adaptive tiling”, *Environmental Modelling & Software*, vol. 197, p. 106 863, Feb. 1, 2026, ISSN: 1364-8152. DOI: 10.1016/j.envsoft.2026.106863.

[11] C. Badii, E. Belay, P. Bellini, *et al.*, “Snap4city: A scalable IOT/IOE platform for developing smart city applications”, in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, pp. 2109–2116. DOI: 10.1109/SmartWorld.2018.00353.

[12] M. Knezevic, A. Donaubauer, and T. H. Kolbe, “SDDI - minimal ecosystem for the establishment of urban digital twins”, *DGPF-Jahrestagung 2024 - Stadt, Land, Fluss - Daten vernetzen*, in collab. with Technische Informationsbibliothek (TIB), pp. 151–168, 2024. DOI: 10.24407/KXP:1885659474.

[13] Z. Yao, C. Nagel, F. Kunde, *et al.*, “3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML”, *Open Geospatial Data, Software and Standards*, vol. 3, no. 1, p. 5, May 14, 2018, ISSN: 2363-7501. DOI: 10.1186/s40965-018-0046-7.

[14] Prefect Technologies, Inc., *Prefect: A modern data orchestration framework*, 2026. [Online]. Available: https://www.prefect.io.

[15] Z. Xu, L. Zhang, H. Li, Y.-H. Lin, and S. Yin, “Combining IFC and 3d tiles to create 3d visualization for building information modeling”, *Automation in Construction*, vol. 109, p. 102 995, Jan. 1, 2020, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2019.102995.

[16] M. Chakraborty and A. P. Kundan, “Grafana”, in *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*, M. Chakraborty and A. P. Kundan, editors, Berkeley, CA: Apress, 2021, pp. 187–240, ISBN: 978-1-4842-6888-9. DOI: 10.1007/978-1-4842-6888-9_6.

[17] Geodan, *Pg2b3dm: Tool for creating 3D Tiles from PostGIS geometries*, https://github.com/Geodan/pg2b3dm, 2026. (visited on 06/11/2026).

[18] P. Cozzi and S. Lilley. “3D Tiles Specification”. (Jan. 12, 2023), [Online]. Available: https://docs.ogc.org/cs/22-025r4/22-025r4.html.