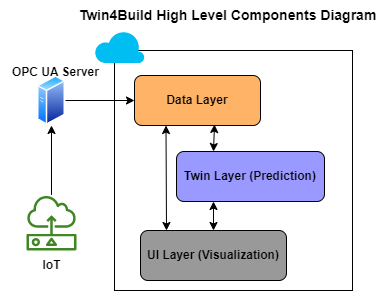
**Architecture of the Digital Twin System**

we've constructed a system that acts as the digital counterpart to a physical entity, enabling us to simulate, analyse and visualize the system. The architecture of this system is multifaceted and sophisticated, incorporating various components and layers that work in concert to achieve its intended functionality. Below is a detailed explanation of the system's architecture depicted in the diagram, broken down for a clear and comprehensive understanding.



High level diagram explained:

* IoT (Internet of Things): This symbolizes the network of physical devices, such as sensors and actuators, embedded with software and connectivity which enables them to connect and exchange data with the OPC UA server and, in turn, with the digital twin system. These IoT devices could be monitoring various aspects of a building, like temperature, energy usage, structural integrity, etc.
* OPC UA Server: OPC UA (Open Platform Communications Unified Architecture) is a machine-to-machine communication protocol for industrial automation. An OPC UA server would collect data from physical devices (IoT devices) and send it to the digital twin system. It may also receive commands from the digital twin system to control the physical devices.
* Data Layer: This component refers to the storage, management, and processing of data collected from the IoT devices. It's where data is analysed and transformed into information that the digital twin can use to do simulation, analysis and visualization.
* Twin Layer (Prediction): This layer of the digital twin uses the information from the data layer to create predictive models. These models help to forecast future states of the building based on current and historical data. It can predict things like equipment failure, energy usage trends, or the need for maintenance.
* UI Layer (Visualization): This is the user interface layer where the data and predictions from the twin layer are presented in a visual format that can be easily understood and interacted with by humans. It would include dashboards, alerts and other visualization tools that help users make decisions about the management of the building.

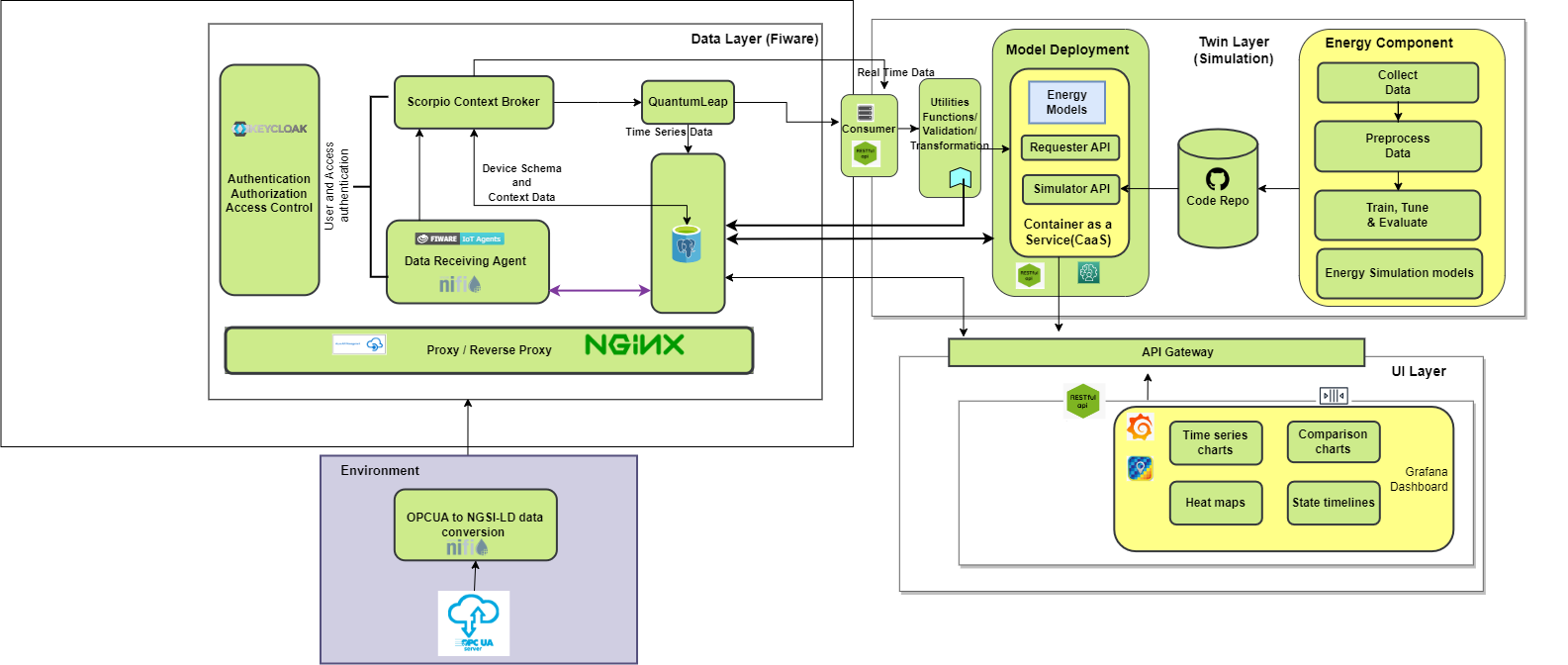


Figure 2: Detailed Archecture Diagram

Data Layer: This layer is a multiheaded data ingestion and data transformation layer it contains several components:

* Keyclock Authentication: It verifies the identities of users and confirms their permissions, acting as a gatekeeper that upholds security through robust authentication, authorization, and access control mechanisms.
* Scorpio Context Broker: Think of this as the system's town crier or data hub. It receives, manages, and disseminates information about the state and activities of devices, ensuring that the digital and physical realms are aligned.
* QuantumLeap: Acting much like a historian, this component meticulously records the timeline of data, storing time series information so that the system can understand and analyse patterns over time.
* PostgreSQL Database: PostgreSQL is used to store, manage, and query high volumes of data collected from various sensors and devices in real-time. Its powerful data processing capabilities allow for complex analyses and queries that are essential for Twin4build.
* Data Receiving Agent (NIFI): This agent can be likened to a postman, collecting data sent from various sources and ensuring it reaches the right destinations within the system.
* NGINX: Serving as a traffic conductor, NGINX directs data requests and responses to and from the right places, functioning as both a proxy and a reverse proxy.

Twin ML Layer: The Twin ML Layer depicted in this diagram is central to the functionality of the Digital Twin, focusing on machine learning (ML) aspects. In this layer, energy models are developed and deployed using a Container as a Service (CaaS) architecture, which allows for scalability and ease of model management.

The core components of this layer include:

* ML pipelines: This involves the processes of data collection, pre-processing, training, tuning, and evaluating machine learning models. These steps are crucial for building accurate and reliable models that can simulate various scenarios and predict outcomes based on the data ingested from the database.
* Code Repository (GitHub): Version control is essential for managing the development of machine learning models, especially in collaborative environments. Using GitHub, developers can track changes, manage updates, and maintain the integrity of the codebase for the energy models. It also allows for versioning of models and rollback in case of issues.
* Deployment as a Service: Once the models are trained and version-controlled, they are deployed as a service. This means they are packaged into containers, which are then managed by a CaaS platform. This setup enables easy deployment, scaling, and management of the machine learning models, facilitating their seamless integration with the rest of the Digital Twin architecture.

This Twin ML Layer serves as a bridge between the raw data coming from the physical systems and the actionable insights provided to the end-users, allowing for advanced simulation and predictive capabilities within the Digital Twin framework.

UI Layer: Finally, we have the UI Layer. The UI Layer in the provided diagram is the visual front-end that users interact with to understand and manage the data processed by the Digital Twin. Within this layer, Grafana stands out as a key component:

* Grafana Dashboard: Grafana is employed to create and display interactive and real-time visualization dashboards. These dashboards provide a powerful way to monitor and analyse time-series data, which is essential for tracking the performance and predicting future states of the system modelled by the Digital Twin.
* Connection to PostgreSQL: The Grafana dashboard is connected to a PostgreSQL database, which acts as the data source. PostgreSQL's robustness and ability to handle complex queries efficiently make it an excellent backend for Grafana, ensuring that the dashboard can pull in large volumes of data and render visualizations quickly and accurately.
* Types of Visualizations: The UI Layer includes various types of visualizations such as time series charts, comparison charts, heat maps, and state timelines. These visualizations help to distil complex data into easily comprehensible formats, allowing users to track changes over time, compare different metrics, and visualize data distribution or the progression of the system's state.

Overall, the UI Layer with Grafana provides users with a powerful toolset for real-time data visualization and analysis, contributing to better decision-making and management of the system represented by the Digital Twin.