

Higher Diploma in Computer Science  
Final Project

# SiteVisor

Digital Twin Application for Buildings Monitoring  
and Asset Management

**Grzegorz Piotrowski**  
**Student number: 20099926**

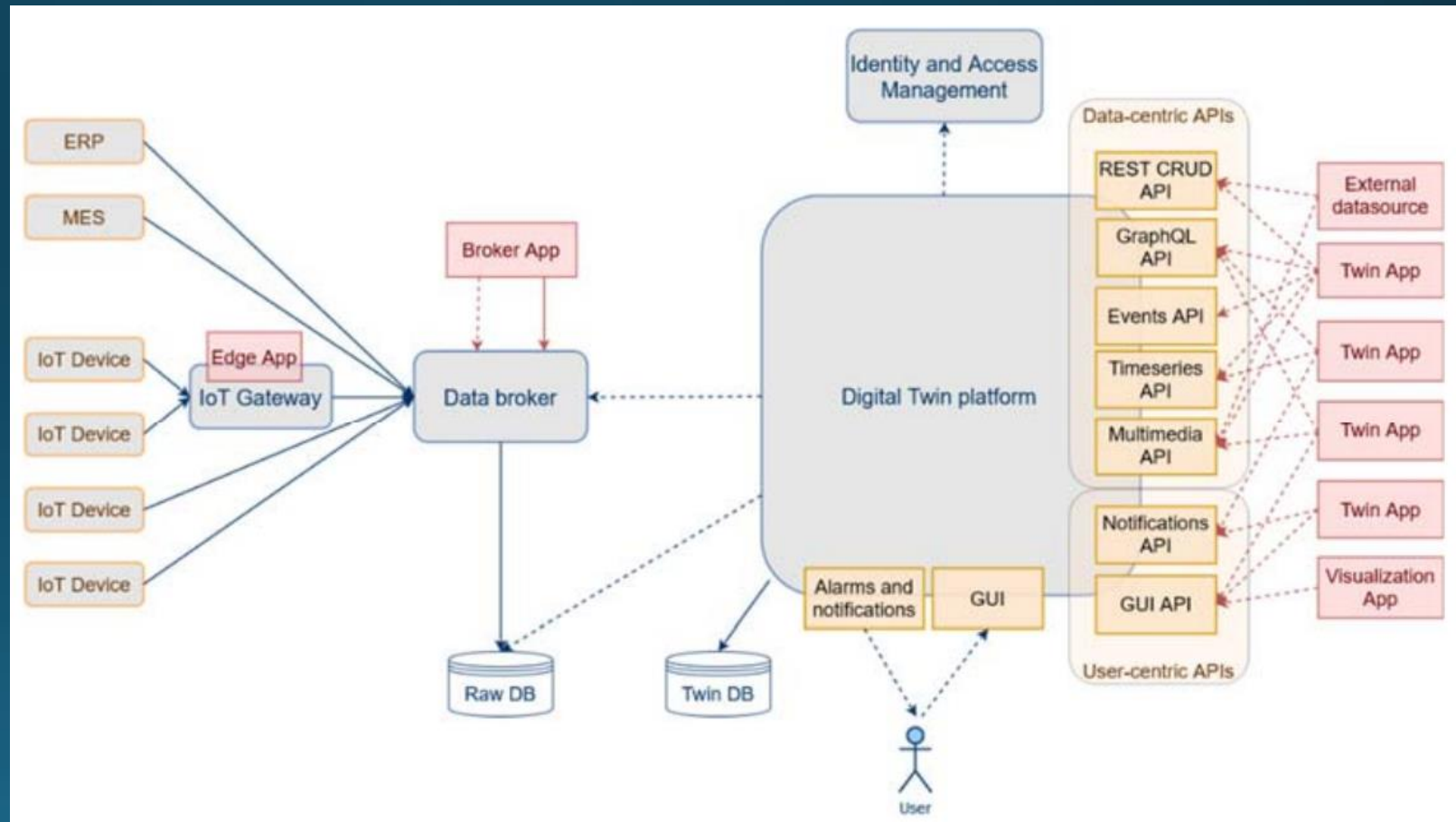
<https://grzpiotrowski.github.io/sitevisor-project/>

# Introduction

- Background
- **Problem statement:** Limited insight into status and condition of a building
- **Proposed solution:** Approachable tool for managing a continuous feed of environmental sensor data
- **Target users:** facility managers, environmental engineers
- **Use cases**
  - Environmental monitoring of buildings
  - Managing rooms and sensors
  - Future: Managing any assets (equipment, building components)

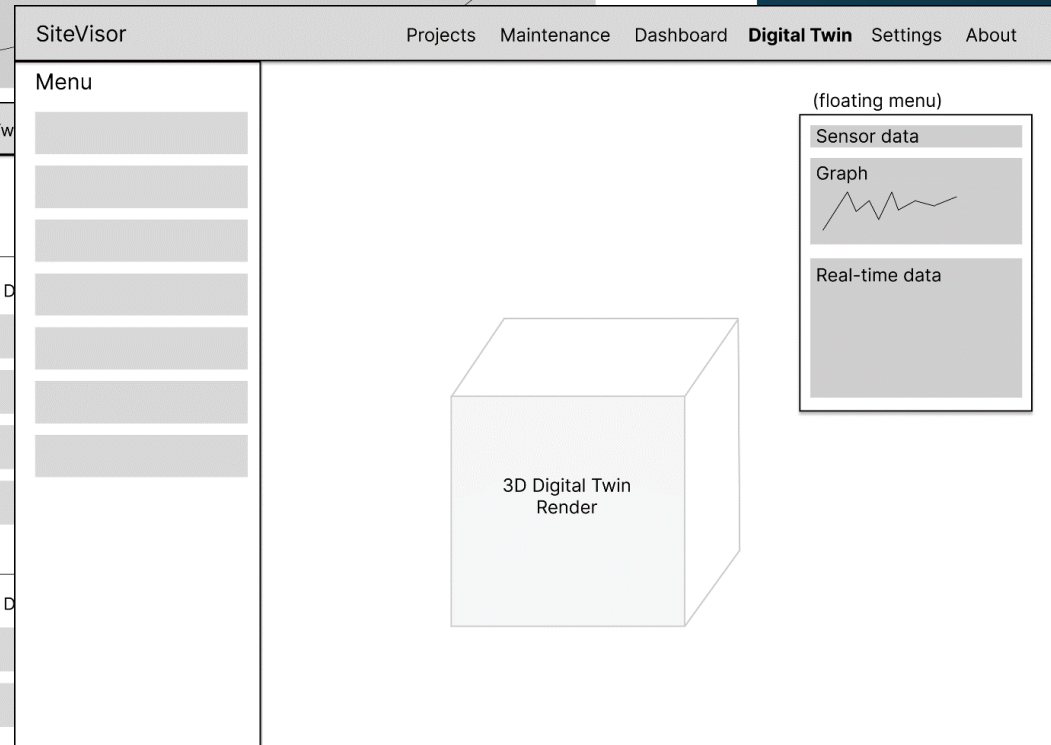
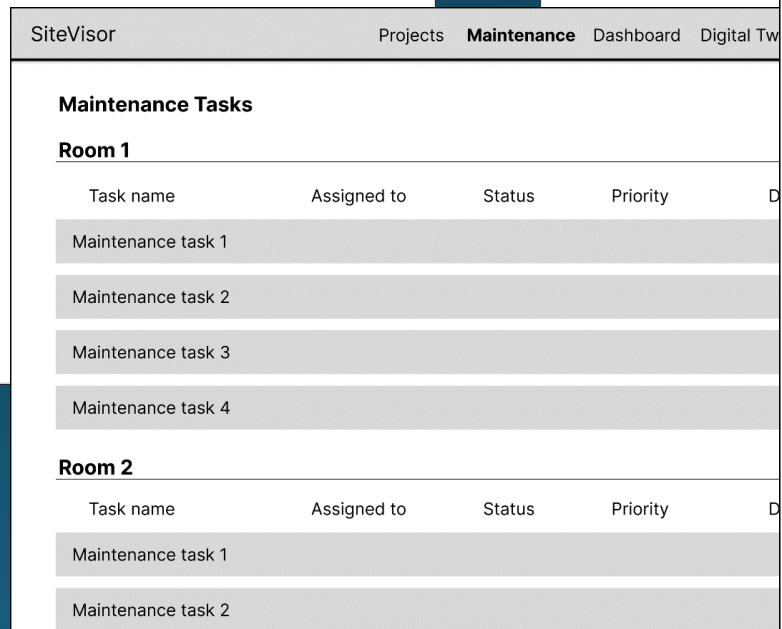
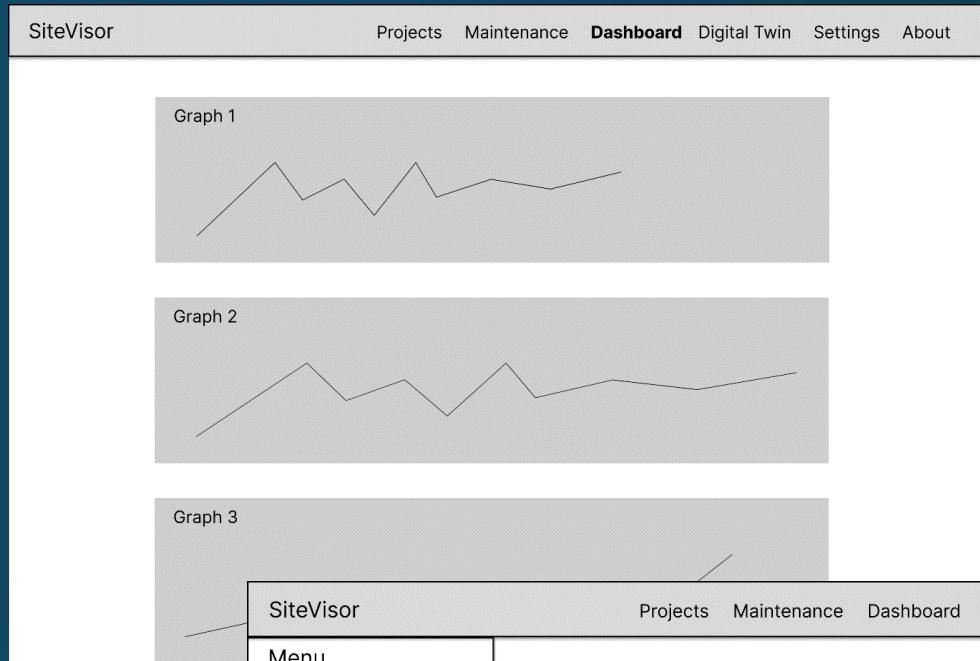
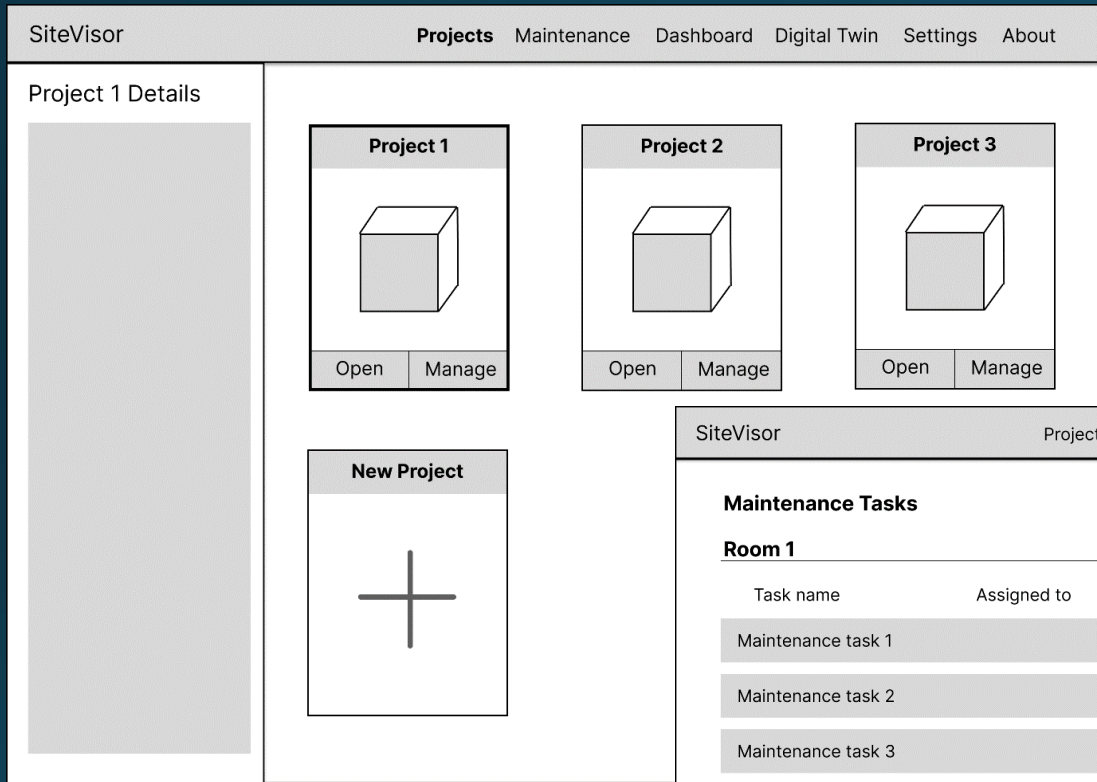
# Research & Analysis

- What is a Digital Twin?
- Technologies:
  - SvelteKit
  - TypeScript
  - WebSocket
  - PostgreSQL
  - Django
  - Docker
  - Strimzi
  - Kafka
  - Kubernetes



# Planning

- Wireframes



# Planning

- GitHub Project

SiteVisor

Overall Board | Issues | Roadmap | Current Board | + New view

is:issue 71 Discard Save

January 2024 February 2024 March 2024

2 milestones

Date	Task	ID
26	SiteVisor Frontend Views	#41
27	Add tests	#16
28	Helper Grid middle line colour	#45
29	Object creation dialog	#49
30	Create Room by dimensions	#46

Task details for SiteVisor Frontend Views #41:

- Add tests #16
- Helper Grid middle line colour #45
- Object creation dialog #49
- Create Room by dimensions #46
- Implement websocket client #21
- Implement POC Kafka Consumer #1
- Add Websocket server #2
- Implement sensor object interactivity #16
- Projects manager page #26

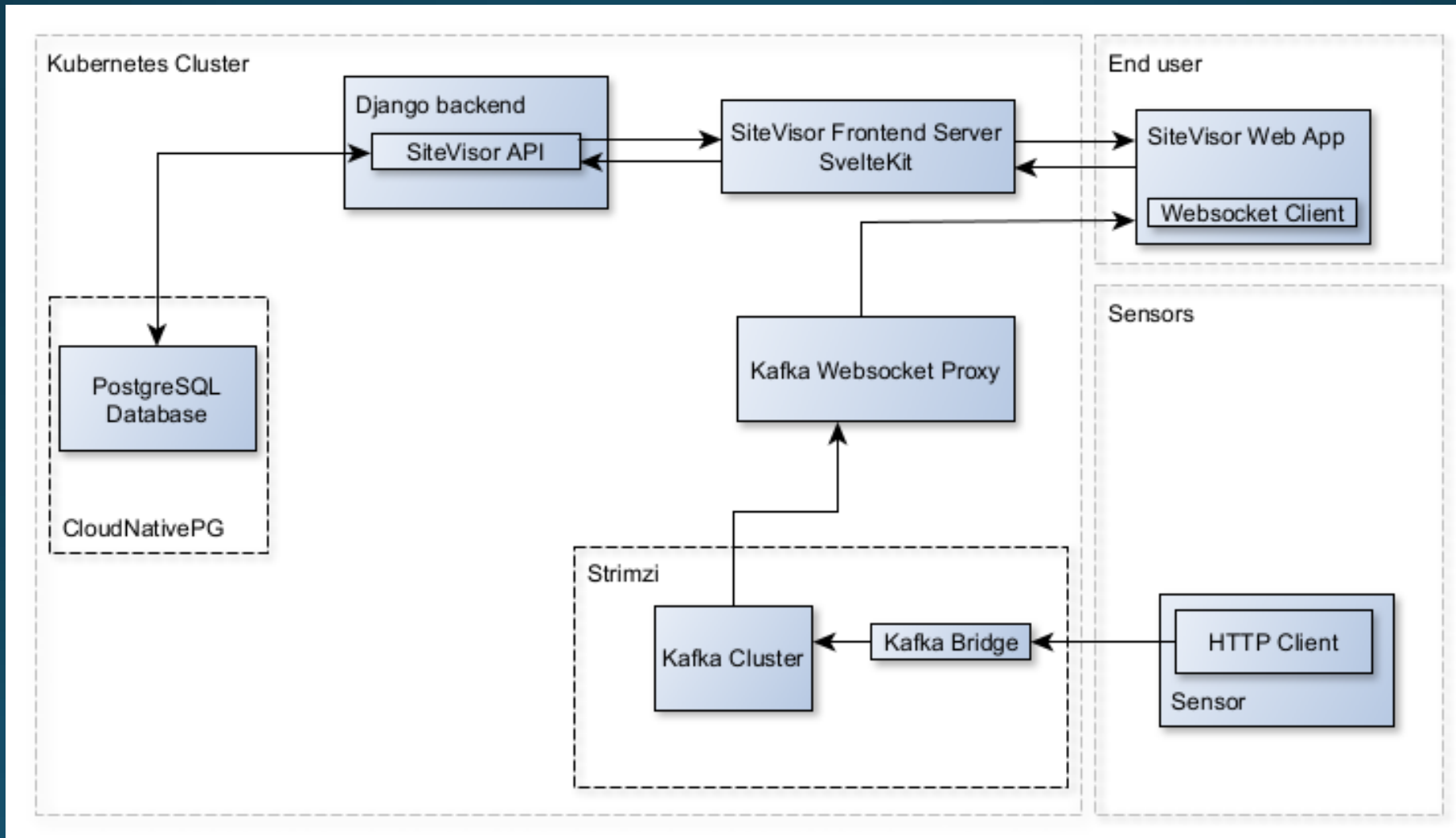
SiteVisor

Overall Board | Issues | Roadmap | Current Board | + New view

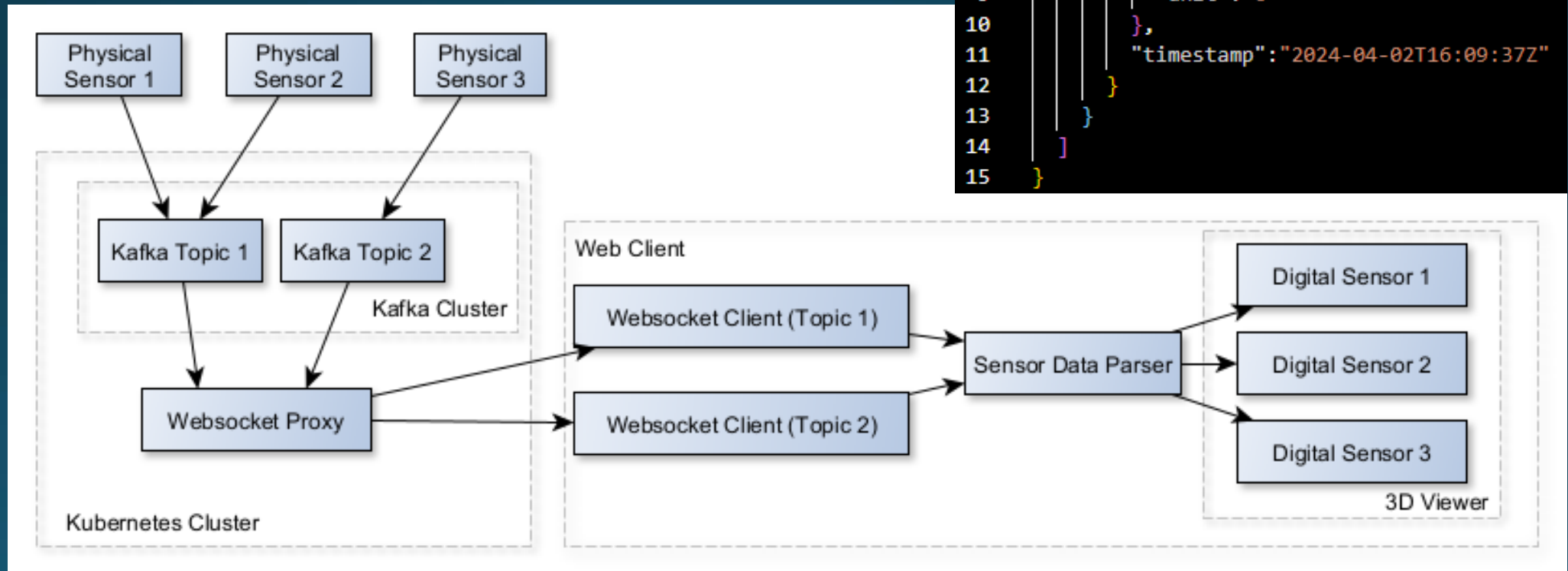
Filter by keyword or by field

- No Status (11)**
  - sitevisor #6: Add user accounts and projects
  - sitevisor #12: Add Asset management functionality
  - sitevisor #7: Share Projects between users
  - sitevisor #11: Enhance the UI with a CSS framework
  - sitevisor #14: Upload 3D model feature
  - sitevisor-project #2: Showcase Entry Part 1
  - sitevisor-project #3: Showcase Entry Part 2
  - sitevisor-project #4: Final Project Submission (code)
- Todo (5)**
  - sitevisor #3: Implement exporting the scene to the backend
  - sitevisor #4: Implement importing the objects from the backend
  - sitevisor #16: Implement sensor object interactivity
  - sitevisor-backend #3: Managing user access to topics
  - sitevisor-project #1: Interim Report
- In Progress (4)**
  - sitevisor #21: Implement websocket client
  - sitevisor #22: Include documentation from sitevisor-backend
  - sitevisor-backend #1: Implement POC Kafka Consumer
  - sitevisor-backend #2: Add Websocket server
- Done (8)**
  - sitevisor #2: Create a starter Three.js scene with basic interactivity
  - sitevisor #1: Establish a sustainable project structure
  - sitevisor #17: Dockerise the application
  - sitevisor #18: App deployment on kubernetes
  - sitevisor #10: Create developer documentation
  - sitevisor #9: Create user documentation
  - sitevisor #15: Add User and Dev documentation with MkDocs

# Implementation - Architecture

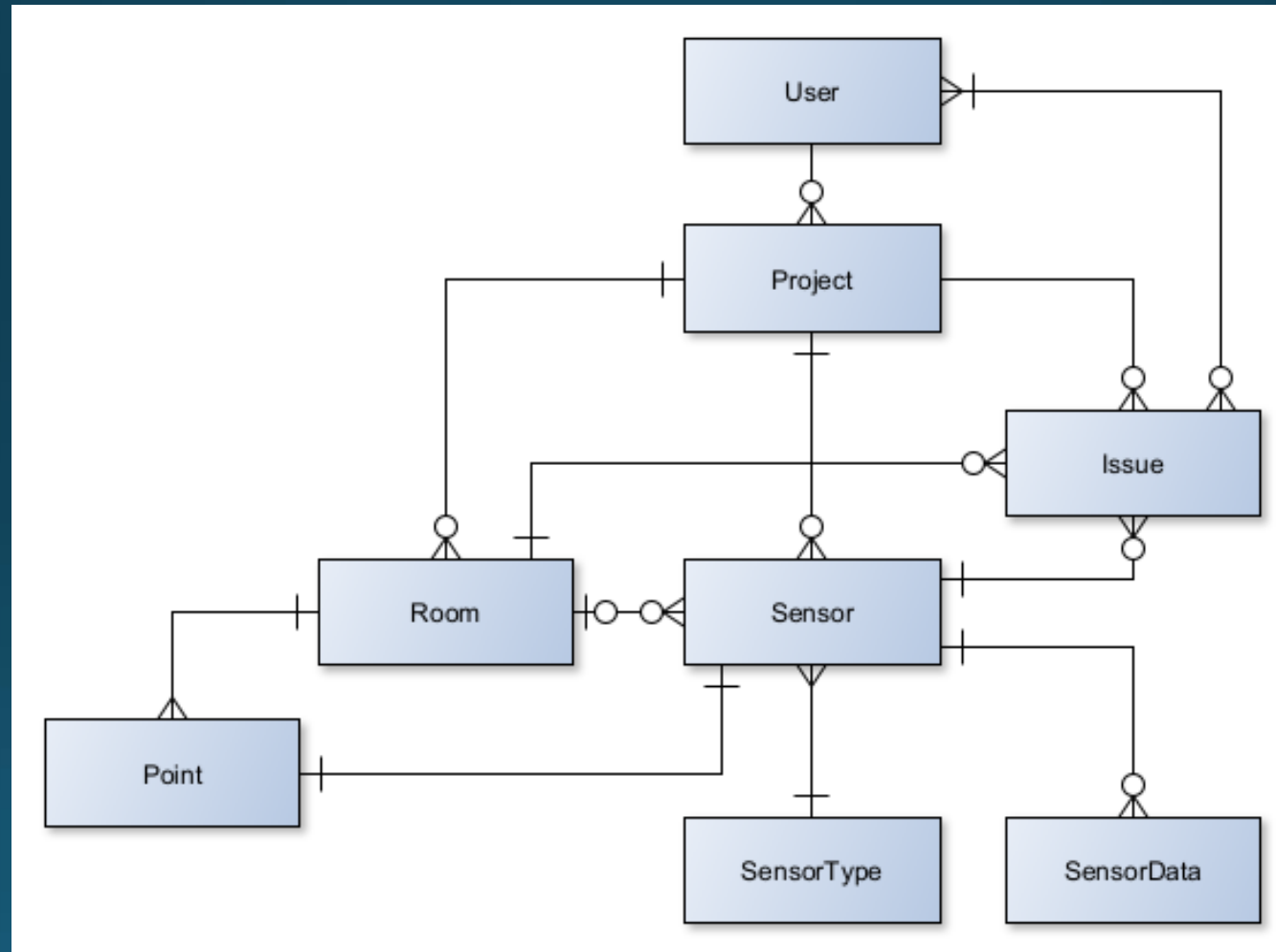


# Implementation – Sensor Data flow



```
1 {
2   "records": [
3     {
4       "value": {
5         "sensor_id": "sensor-123",
6         "sensor_type": "temperature",
7         "data": {
8           "value": 20.23,
9           "unit": "C"
10        }
11      },
12      "timestamp": "2024-04-02T16:09:37Z"
13    }
14  ]
15 }
```

# Implementation – Data model

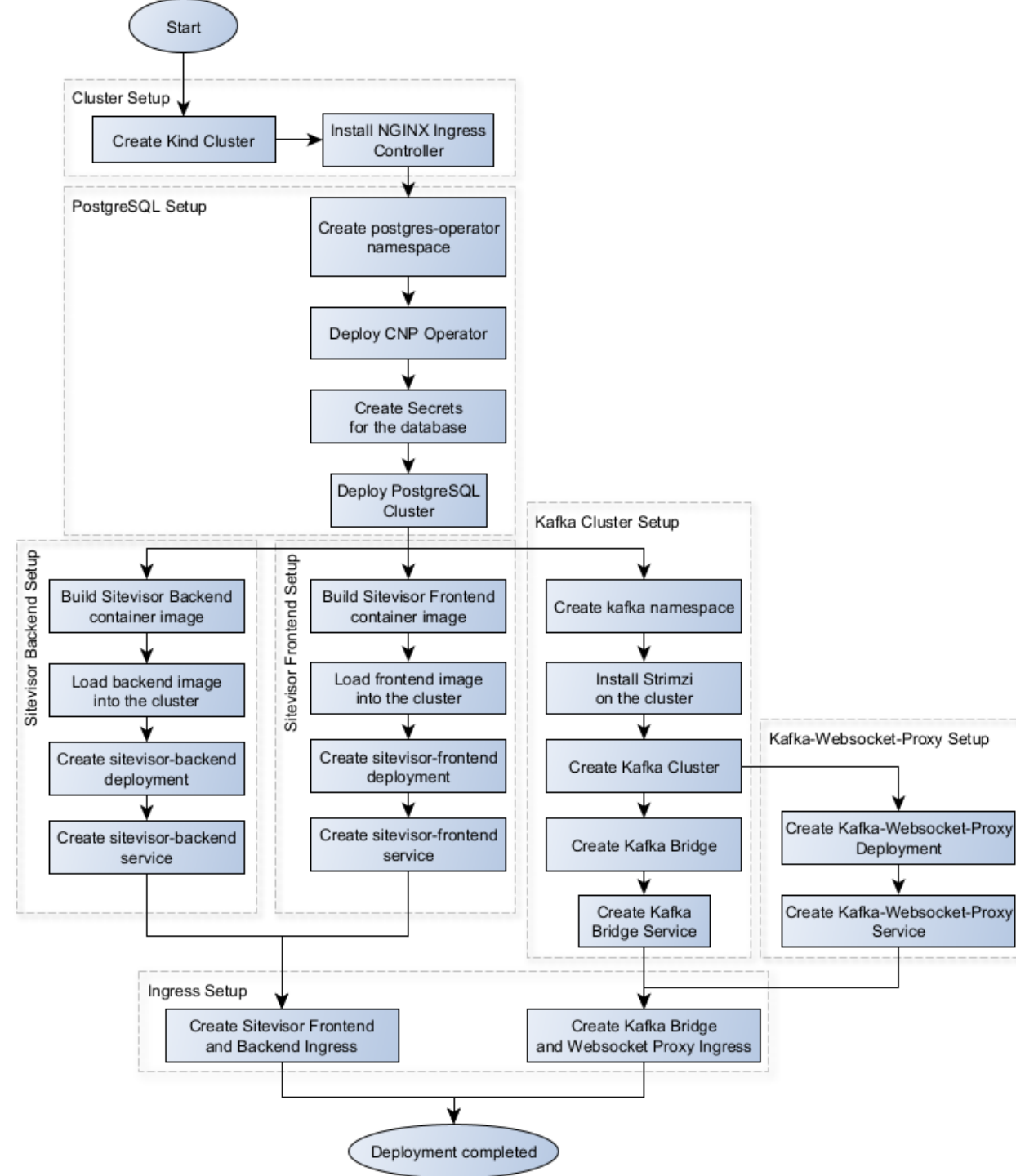




# Deployment

- Kind Cluster for development
- Strimzi: Kafka in Kubernetes

- Future:
  - CI/CD Pipeline
  - Cloud deployment



# Documentation

- Vital part of the project
- Describes k8s deployment

The screenshot shows the SiteVisor documentation page for PostgreSQL deployment. The page has a dark blue header with the SiteVisor logo, a search bar, and GitHub repository information. The left sidebar contains a navigation menu with categories like Overview, User Guide, Developer Guide, Architecture, and Frontend Overview. The main content area is titled 'PostgreSQL deployment' and includes an introduction, prerequisites, and code snippets for creating a namespace, deploying the operator, and creating a secret.

**SiteVisor** Search GitHub ☆ 0 🐙 0

SiteVisor  
Overview  
User Guide ▾  
  Project Management  
Developer Guide ▾  
  Local deployment  
  Docker deployment  
  Kind deployment ▾  
    Kind Cluster  
    PostgreSQL deployment  
  Backend deployment  
  Frontend deployment  
  Kafka deployment  
  Kafka Websocket Proxy deployment  
  Ingress configuration  
Architecture ▾  
  Architecture Overview  
  Frontend Overview

## PostgreSQL deployment

This section provides steps required to deploy the CloudNativePG operator which manages PostgreSQL workloads on Kubernetes cluster.

### Prerequisites

- Running Kind cluster

**Create a Namespace for the PostgreSQL Operator:**

```
kubectl create namespace postgres-operator
```

**Deploy the CNP Operator:**

```
kubectl apply -n cnpg-system -f https://github.com/cloudnative-pg/cloudnative-pg/r
```

**Create a Secret for the PostgreSQL Superuser:**

```
kubectl create secret generic postgres-superuser-secret --from-literal=username=po:
```

Table of contents  
Prerequisites  
Inspecting the database

# Overcoming Challenges

- Kafka – Websocket Connection
  - Not natively supported by Kafka or Strimzi.
  - Found an existing project as a fitting solution
- Websocket connection to Kubernetes
  - Wanted to use k8s Gateway API initially (WebSocket support only in experimental channel)
  - Switched to NGINX Ingress.
- Managing Websocket Clients
- Designing project architecture
- Structuring the codebase
- Database integration: MongoDB vs PostgreSQL

# Use of AI

- No AI used for the report, video, project ideation or docs
- ChatGPT used in code development:
  - Way of condensing lengthy internet searching
  - Getting exposed to tech, ideas, approaches
  - Quickly prototyping
  - Getting a very rough idea for implementation to kickstart work
  - Simple copy/paste was not an option
  - Always ended up needing own research and dev

# Project demonstration

# Future development

- Implement Sink Connector for sensor data (WIP)
- User Roles
- Focus on Security – Manage access to Kafka
- Move WebSocket clients to Web Workers
- Upload GLTF Files
- Alerts/Notifications
- Enhanced Digital Twin and 3D Viewer features
  - Object collisions
  - Generic Asset object
  - Send signals from Digital Twin to actuators/devices
  - Room properties derived from sensor data
  - Building levels
- Frontend tests (challenging for Three.js)

# Recap & Learnings

- More experience with Svelte/SvelteKit
- Expanded on my Three.js skills
- Learned Django
- Learning Kafka and Strimzi from scratch
- Kubernetes from Web App Developer's perspective
- Extensive use of Git/GitHub

Thank You