КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ
Кафедра кібербезпеки та комп'ютерної інженерії

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: Система моніторингу та керування IoT-пристроями(веб-додаток)

Виконав: Козлюк І.Ю.
Керівник: Шабала Є.Є.

Київ 2025р.

Актуальність проблеми фрагментації в екосистемі Інтернету речей

Масштаб явища

Кількісне зростання

Прогнозована кількість IoT-пристроїв у світі до 2030 року:

> 29 мільярдів

Джерело: Statista, 2023.

Економічне зростання

Прогнозований обсяг глобального ринку IoT до 2028 року:

> 2.4 трильйона доларів США

Джерело: Grand View Research, 2024.

Суть проблеми

«Зоопарк» технологій

Кожен виробник створює власну закриту екосистему **з несумісними протоколами та додатками.**

Наслідки для користувача

Результат: Необхідність використовувати **десятки різних додатків** та неможливість створення єдиних сценаріїв автоматизації.

Мета та завдання кваліфікаційної магістерської роботи

Мета дослідження

Розробка та дослідження гнучкої мікросервісної архітектури для універсальної веб-системи моніторингу та керування пристроями Інтернету речей.

Об’єкт та предмет дослідження

Об’єкт: Процес управління та моніторингу гетерогенними IoT-пристроями.

Предмет: Методи, моделі та програмні засоби для інтеграції різнорідних IoT-пристроїв в єдину хмарну веб-платформу.

Основні завдання

1. Провести аналіз предметної області, технологій та існуючих рішень.

2. Спроєктувати гнучку мікросервісну архітектуру системи.

3. Розробити програмний прототип, що реалізує ключові компоненти архітектури.

4. Провести функціональну валідацію розроблених проєктних рішень.

Аналіз ринку та визначення незайнятої ніші

Локальні Open-Source системи

Home Assistant

Технічні ентузіасти

- \+ Максимальна гнучкість та приватність даних.
- − Високий поріг входу: вимагає технічних знань та власного обладнання.

Enterprise Open-Source системи

ThingsBoard

Бізнес (від СМБ до enterprise)

- \+ Готові бізнес-функції (multi-tenancy, rule engine).
- − Надлишковість: складний в адмініструванні та надлишковий для індивідуальних потреб.

Глобальні хмарні платформи

AWS IoT Core

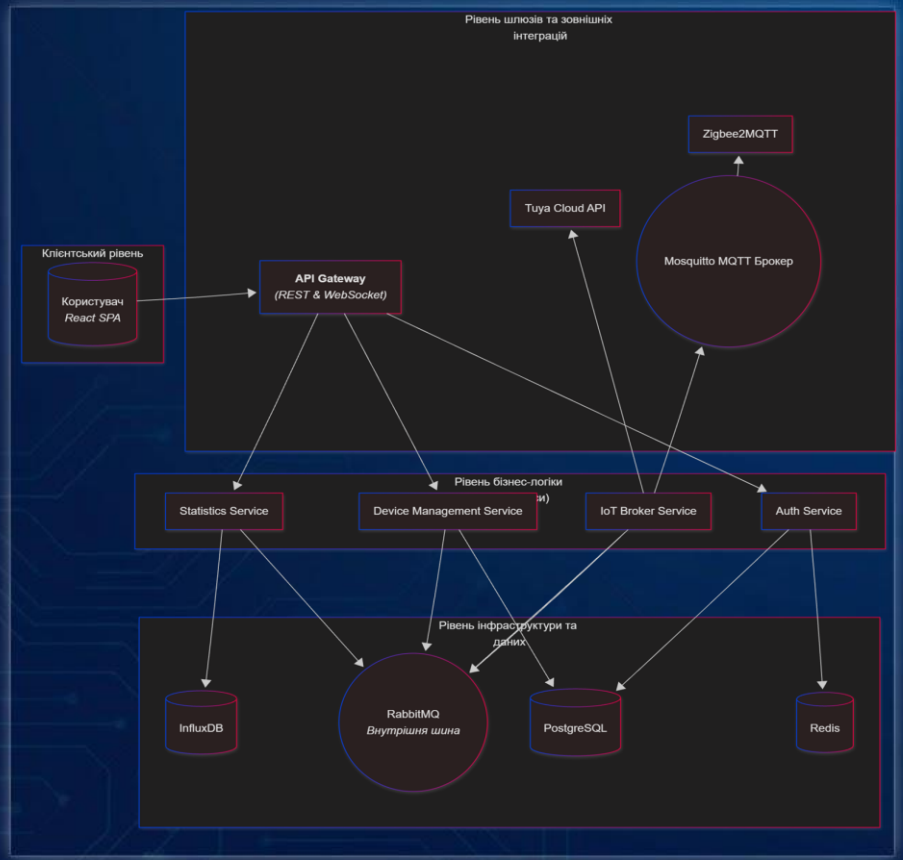
Великі корпорації

- \+ Нескінченна масштабованість та екосистема.
- − Дуже висока складність та непередбачувана вартість для малих проєктів.

Висновок: Незайнята ринкова ніша

Існує потреба в публічній хмарній SaaS-платформі, яка б поєднувала простоту використання хмарних сервісів з доступністю та орієнтацією на масового кінцевого користувача.

Розроблена компонентна архітектура системи



Ключові архітектурні рішення:

- **Мікросервісна архітектура**: Система декомпозована на незалежні, спеціалізовані сервіси.
- **Подієво-орієнтована взаємодія**: Сервіси спілкуються асинхронно через шину повідомлень RabbitMQ.
- **Єдина точка входу**: API Gateway виступає як "фасад", що інкапсулює внутрішню складність системи.

Проєктні рішення для серверної частини (Бекенду)

Основа мікросервісів

NestJS Framework

- Обрано як основу для всіх мікросервісів.
- Переваги:
- Надає потужну, структуровану архітектуру "з коробки" (модулі, сервіси, контролери).
- Вбудована підтримка TypeScript забезпечує надійність та масштабованість коду.
- Спрощує реалізацію складних шаблонів (напр., Dependency Injection).

Внутрішньосервісна комунікація

gRPC Protocol

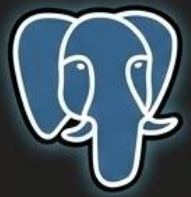
- Обрано для синхронної комунікації між сервісами (напр., API Gateway -> Auth Service).
- Переваги:
- Висока продуктивність завдяки бінарному формату Protobuf та HTTP/2.
- Строгі контракти (.proto файли) виключають помилки інтеграції між сервісами.
- Значно швидший за традиційний REST API для внутрішніх викликів.

Асинхронна взаємодія

RabbitMQ Message Broker

- Обрано як асинхронну шину повідомлень для подієво-орієнтованої взаємодії.
- Переваги:
- Забезпечує слабку зв'язаність: сервіси не залежать один від одного напряму.
- Підвищує відмовостійкість: діє як буфер, що запобігає втраті даних при тимчасових збоях.
- Гнучка маршрутизація повідомлень завдяки реалізації протоколу AMQP.

Підхід "Polyglot Persistence": вибір сховища для кожної задачі



PostgreSQL

- **Призначення:** Зберігання структурованих метаданих.
- **Сутності:** Користувачі (Account), Пристрої (Device), Групи (Group).
- **Чому обрано:** Гарантія цілісності даних (ACID) та реляційні зв'язки.

InfluxDB

- **Призначення:** Зберігання часових рядів (телеметрії).
- **Сутності:** Показники сенсорів, історія змін станів.
- **Чому обрано:** Висока швидкість запису та стиснення часових даних.

Redis

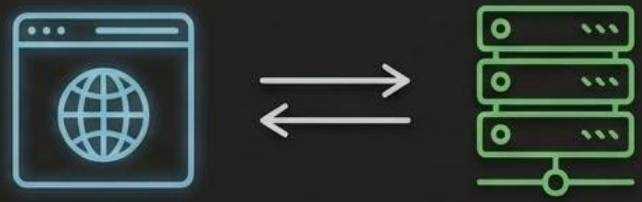
- **Призначення:** Кешування та швидкий доступ.
- **Сутності:** Активні сесії, JWT-токени, "гарячий" стан пристроїв.
- **Чому обрано:** Мінімальні затримки доступу (in-memory).

Проєктні рішення для клієнтської частини (Фронтенду)

Архітектура SPA (Single Page Application)

- **Принцип:** Додаток завантажується один раз, далі працює динамічно.
- **Переваги:** Плавний інтерфейс без перезавантаження сторінок (як у нативних додатків).
- **Структура:** Компонентний підхід (перевикористання віджетів).

Двоканальна взаємодія з сервером



🔄 **1. REST API (HTTP)**
Для "холодних" даних (запит списків, відправка команд, логін).

⚡ **2. WebSocket (WS)**
Для "гарячих" даних (миттєве оновлення дашборду в реальному часі).

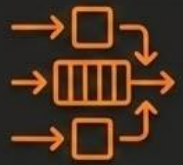
Наукова новизна отриманих результатів

Отримала подальший розвиток архітектура хмарних IoT-платформ

Створено рішення, що поєднує промислову надійність з простотою для кінцевого користувача.

Мікросервісна архітектура + gRPC

- Декомпозиція на незалежні сервіси.
- Високоефективна внутрішня комунікація.



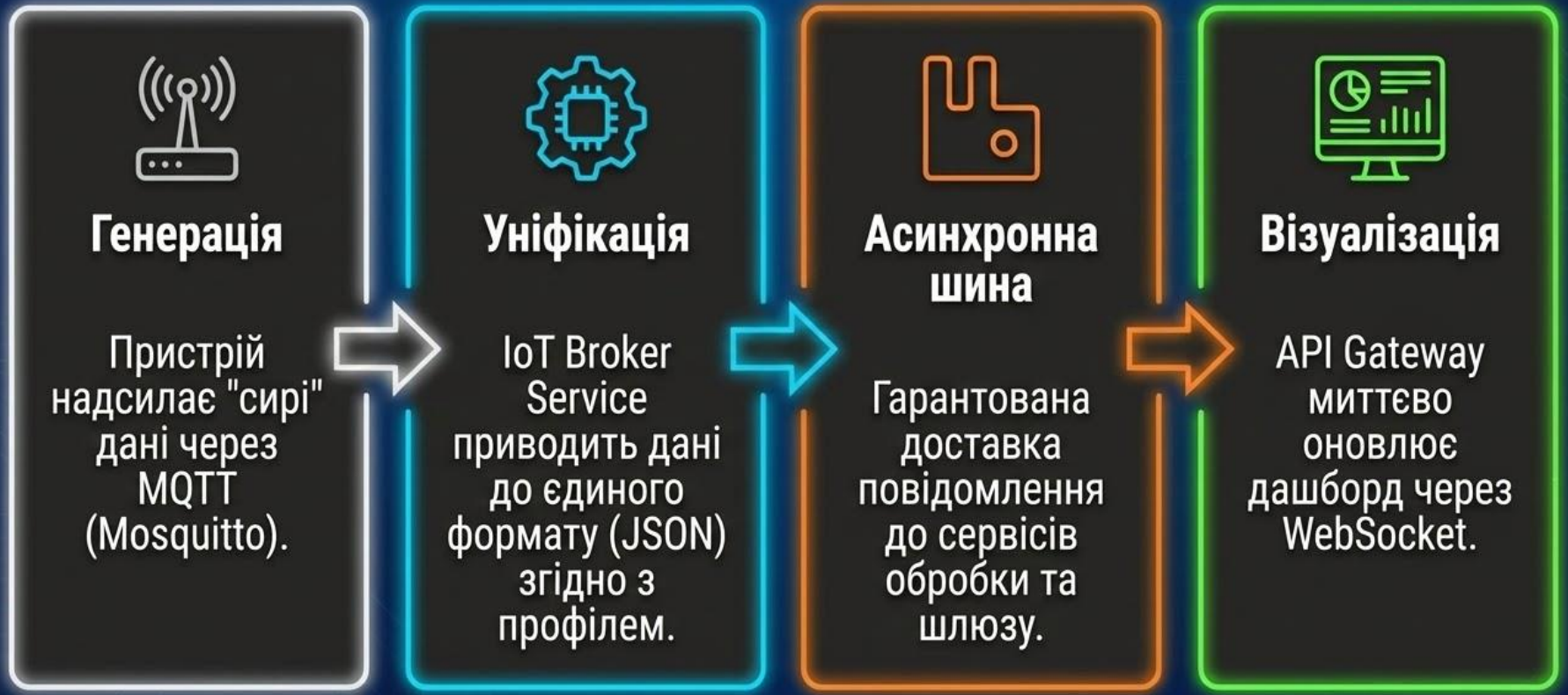
Асинхронна взаємодія (RabbitMQ)

- Слабка зв'язаність компонентів.
- Гарантована доставка даних (буферизація).

Уніфікація через "Профілі пристроїв"

- Гнучка інтеграція різнорідного обладнання.
- Декларативний опис правил мапінгу даних.

Схема потоку обробки даних у реальному часі



Демонстрація програмного прототипу

Сторінка авторизації

Реалізовано OAuth 2.0 (Google) та JWT-сесії.

Головний дашборд (SPA)

Компонентна архітектура на базі React.

Результати функціональної валідації архітектури

Асинхронна обробка

ПІДТВЕРДЖЕНО

Успішне проходження повідомлень через черги RabbitMQ від пристрою до шлюзу.

Відмовостійкість

ПІДТВЕРДЖЕНО

Відсутність втрати даних при імітації збою сервісу статистики (буферизація).

Механізм безпеки

ПІДТВЕРДЖЕНО

Коректна генерація та валідація JWT-токенів (Access/Refresh).

Практичне значення отриманих результатів

Архітектурний шаблон

Готовий "blueprint" для створення масштабованих IoT-систем. Економить час на проєктування архітектури для нових стартапів.

Прикладні сфери

- Системи "Розумний будинок".
- Моніторинг малого бізнесу (склади, серверні кімнати, агросектор).

Навчальна платформа

Використання як наочного посібника для вивчення мікросервісів, gRPC та RabbitMQ у навчальному процесі.

Висновки

- Проаналізовано проблему фрагментації ринку IoT та недоліки існуючих платформ.
- Розроблено та обґрунтовано гнучку мікросервісну архітектуру на базі сучасного стеку (NestJS, gRPC, RabbitMQ).
- Створено програмний прототип системи з веб-інтерфейсом (SPA) та підтримкою реального часу.
- Валідовано ключові архітектурні рішення (надійність, безпека, інтеграція) на практиці.

Мета кваліфікаційної магістерської роботи досягнута в повному обсязі.

Дякую за увагу!

Доповідь завершено.

Готовий відповісти на Ваші запитання

Козлюк Ілля Юрійович | 2025