Contents

Struktura Projektu Primus Inter Pares 2026

Przegląd

Projekt jest systemem zarządzania magazynem (Hybrid Event-Driven Modular Monolith) przygotowanym na zawody Primus Inter Pares 2026. Składa się z mikroserwisów (backend, frontend, mobile, IoT) orkiestrowanych przez Docker Compose.

Główne Katalogi

`/home/gabig/Primus2026` (Root)

Główny katalog projektu.

- **`primus-infra/`**: Konfiguracja infrastruktury.
  - `docker-compose.yml`: Definicja usług (db, redis, backend, worker, frontend, mqtt).
  - `.env`: Zmienne środowiskowe dla Dockera.
- **`primus-backend/`**: Logika biznesowa (Python, FastAPI).
  - `app/`: Kod źródłowy.
    - `api/`: Endpointy REST API.
    - `core/`: Konfiguracja, zabezpieczenia (auth).
    - `models/`: Modele SQLAlchemy (baza danych).
    - `schemas/`: Modele Pydantic (walidacja danych).
    - `services/`: Logika biznesowa (CRUD, serwisy).

    - `alembic/`: Migracje bazy danych.
    - `tests/`: Testy (Pytest).
    - `pyproject.toml`: Zarządzanie zależnościami (Poetry).
- **`primus-web-frontend/`**: Panel webowy (React, Vite, TypeScript).
    - `src/`: Kod źródłowy.
        - `components/`: Komponenty UI (Shadcn/UI).
        - `pages/` lub `views/`: Ekrany aplikacji.
        - `api/`: Integracja z backendem.
    - `vite.config.ts`: Konfiguracja Vite.
- **`primus-mobile/`**: Aplikacja mobilna (React Native, Expo).
    - `app/`: Routing oparty na plikach (Expo Router).
    - `components/`: Komponenty mobilne.
    - `assets/`: Obrazy, czcionki.
- **`primus-mock-sensor/`**: Symulator urządzeń IoT.
    - Skrypty Python generujące dane do MQTT.
- **`primus-mqtt-listener/`**: Usługa nasłuchująca MQTT.
    - Odbiera dane z czujników i przekazuje do systemu.
- **`.agent/workflows/`**: Definicje workflow dla agenta AI.
    - `kanban_workflow.md`: Zasady pracy z Kanbanem i GitHubem.

Tech Stack

- **Backend**: Python 3.11+, FastAPI, SQLAlchemy, Alembic, PostgreSQL, Redis, Celery.
- **Frontend**: React 18, TypeScript, Vite, TailwindCSS, Shadcn/UI.
- **Mobile**: React Native, Expo, NativeWind.
- **IoT**: MQTT (Mosquitto).
- **Infra**: Docker, Docker Compose.

Workflow Deweloperski

1. Uruchamianie Infrastruktury

W katalogu `primus-infra`:

```
docker-compose up -d --build
```

Uruchamia bazę danych, Redisa, brokera MQTT oraz serwisy backendowe/frontendowe zdefiniowane w compose.

2. Backend Development

Lokalne uruchamianie (jeśli nie przez Docker):

```
cd primus-backend
poetry install
poetry run uvicorn app.main:app --reload
```

Migracje bazy danych:

```
poetry run alembic upgrade head
```

3. Frontend Development

```
cd primus-web-frontend
npm install
npm run dev
```

4. Mobile Development

```
cd primus-mobile
npm install
npx expo start
```

Wybierz ‘a’ dla Androida lub ‘i’ dla iOS (wymaga symulatora) lub zeskanuj QR kod aplikacją Expo Go.

Kluczowe Pliki Dokumentacji

- `task.md`: Bieżące zadania i postęp (“pamięć” agenta).
- `project-context.md`: Kontekst architektoniczny (w pamieci agenta).

Ważne Komendy Agenta

- `/kanban_workflow`: Sprawdź zasady pracy z zadaniami.

Uwagi dla Agenta (Ty z Przyszłości)

- Wszystkie zmiany w kodzie powinny być odzwierciedlane w testach.
- Pamiętaj o aktualizacji modeli Pydantic (`schemas`) przy zmianach w modelach DB (`models`).
- Przy problemach z Dockerem sprawdź logi: `docker-compose logs -f [service_name]`.
- Pliki w `task.md` są priorytetem do śledzenia postępu.