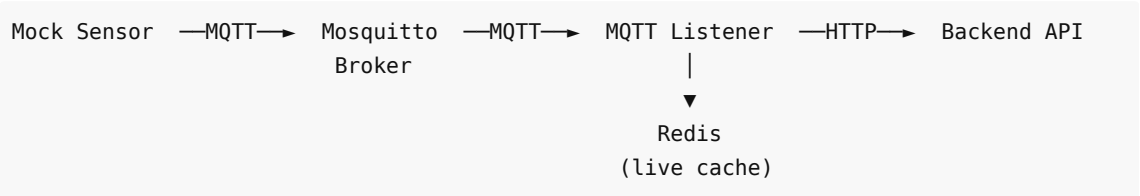
Primus MQTT Listener

Mikroserwis nasłuchujący dane z czujników IoT dla systemu **Primus Inter Pares 2026**. Odpowiada za przetwarzanie odczytów temperatury i wagi w czasie rzeczywistym oraz generowanie alertów.

Pełna dokumentacja projektu: Primus Docs

Architektura

Asynchroniczny serwis Python działający jako pomost między warstwą IoT a backendem.



Realizacja Wymagań Specyfikacji

5c. Monitoring Wagi (Nieautoryzowane Pobranie)

- **Implementacja:** `listener.py`
- **Logika:**
  1. Pobiera ostatnią prawidłową wagę z Redis (`Weight:{designation}:{row}:{col}`).
  2. Porównuje z aktualnym odczytem z sensora.
  3. Jeśli różnica > `TOLERANCE` (0.5 kg) i nie ma aktywnej transakcji (`ExpectedChange:...`), generuje alert `WEIGHT` .

5a/5b. Monitoring Temperatury

- **Logika:**
  1. Pobiera definicje regałów z backendu (zakresy `temp_min` , `temp_max`).
  2. Dla każdego odczytu sprawdza, czy temperatura mieści się w zakresie.
  3. Przy przekroczeniu generuje alert `TEMP` .

Funkcjonalności

Funkcja	Opis
MQTT Subscription	Nasłuchuje na topiku `sensors/+` (wszystkie regały)
Redis Cache	Zapisuje aktualne odczyty (`rack:{id}:live`) z TTL 60s
Alert Throttling	Nie wysyła tego samego alertu częściej niż raz na 60s
Auto-Login	Automatyczna autoryzacja do backendu z odświeżaniem tokena
Weight Sync	Przy starcie synchronizuje expected weights z backendu

Stos Technologiczny

Kategoria	Technologia

Język	Python 3.11
MQTT	Paho MQTT Client
Cache	Redis
HTTP	Requests
TLS	Obsługa certyfikatów dla MQTT i API

Uruchomienie

Docker (Zalecane)

Serwis uruchamiany automatycznie jako część stosu primus-infra:

```
cd ../primus-infra
docker compose up -d mqtt-listener
```

Lokalnie

Wymagane zmienne środowiskowe:

```
export MQTT_BROKER=localhost
export MQTT_PORT=1883
export REDIS_HOST=localhost
export BACKEND_URL=http://localhost:8000/api/v1
python listener.py
```