PZSP2 – Wirtualizacja/konteneryzacja, bezpieczeństwo

Zespół 11

Piotr Szmurło

Denys Savytskyi

Do Truong Giang

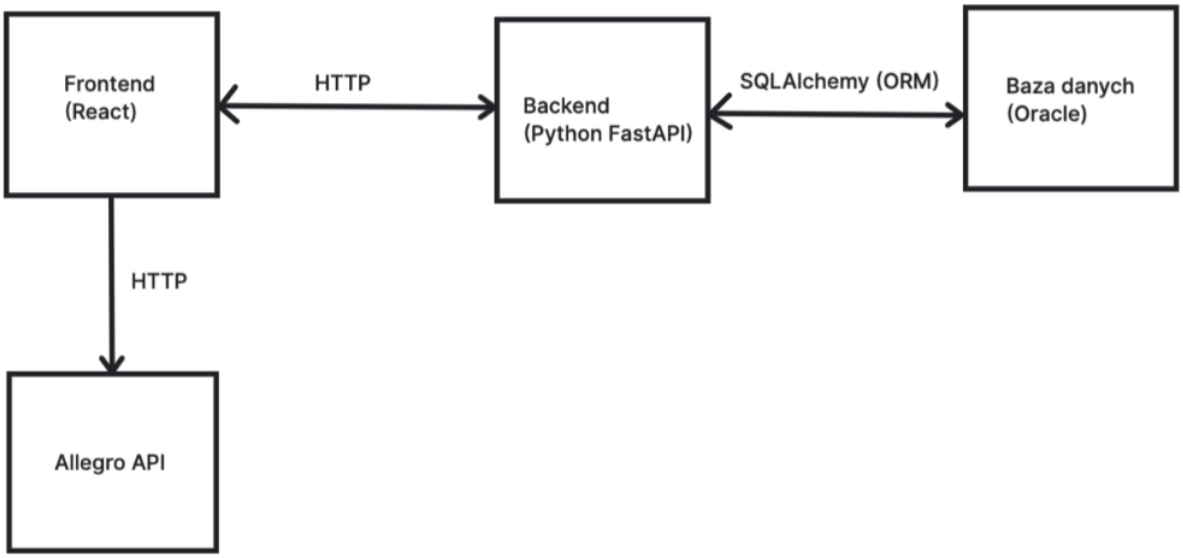
Piotr Kosmala

Krótki opis architektury

Nasz system składa się z trzech głównych elementów architektonicznych:

- API zrealizowane w frameworku FastAPI w języku Python
- Frontend napisany w technologii React
- Baza danych Oracle dostarczona nam przez uczelnię

Komunikacja przebiega za pomocą requestów http wysyłanych przez frontend do naszego API, które może wykonywać zapytania do bazy danych w celu zdobycia potrzebnych danych.



Dodatkowym elementem będzie komunikacja z API wystawionym przez środowisko Allegro Sandbox, ale jako że jest to podmiot zewnętrzny, to nie będziemy się na nim skupiać w tym dokumencie.

Zastosowane rozwiązania konteneryzacyjne

Projekt został wykonany z myślą o uruchomieniu go w kontenerach w środowisku Docker. W tym celu powstały poniższe pliki dockerfile:

backend > Dockerfile > ...

```
FROM python:3.9
WORKDIR /code
COPY ./requirements.txt /code/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
COPY . /code

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8080"]
```

frontend > Dockerfile > ...

```
FROM node:14-alpine
WORKDIR /
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]

```

Oraz następujący plik docker-compose.yml:

docker-compose.yml

```
version: "3.9"
services:
  api:
    build:
      context: backend
      dockerfile: Dockerfile
    environment:
      - DB_PASSWORD=${DB_PASSWORD}
    ports:
      - "8080:8080"

  app:
    build:
      context: frontend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    stdin_open: true

```

Takie rozwiązanie pozwala nam uruchamiać cały system jedną komendą terminalową, co jest bardzo wygodne podczas rozwoju aplikacji.

Jeśli chodzi o istotne zagadnienia bezpieczeństwa, widać tutaj, że hasło do bazy danych nie jest podane wprost nigdzie w projekcie – zamiast tego brane będzie z pliku .env, który musi być dodany przez każdą osobę pragnącą uruchomić system. W ten sposób ustawiamy zmienną środowiskową DB_PASSWORD, która następnie zostanie pobrana podczas próby utworzenia connection stringa:

```
def get_engine():
    with open("config.yaml") as config_file:
        config = yaml.safe_load(config_file)['database']
        db_password = os.getenv('DB_PASSWORD')
        if db_password is None:
            raise Exception("DB password not set")
        connection_uri = f"{config['dialect']}+{config['driver']}://{config['username']}:{db_password}@{config['host']}:{config['port']}/{config['dbname']}"
        return create_engine(connection_uri)
```

Potencjalne zagrożenia bezpieczeństwa

- Przechowywanie haseł użytkownika – w bazie danych przechowywane będą jedynie wartości funkcji haszującej wyliczonej na bazie hasła
- Uwierzytelnienie dostępu użytkownika do zasobu – podczas zalogowania zostanie wygenerowany JSON Web Token (jwt), na podstawie którego backend będzie zwracał odpowiednią zawartość lub błąd 403 (Unauthorised)
- Dane wrażliwe użytkowników – nasz system nie przechowuje danych wrażliwych poza adresem email
- Uwierzytelnienie w systemie Allegro Sandbox – przebiega za pośrednictwem tokenu generowanego przez API Allegro, zatem nie musimy przechowywać żadnych haseł do tego portalu
- Niepowołane osoby uzyskujące dostęp do naszego API, a tym samym bazy danych
    - Obecnie API przyjmuje jedynie zapytania z adresu *localhost:3000* (z możliwością łatwego dodania kolejnych adresów)
    - Dostęp do bazy danych jest możliwy jedynie po poznaniu hasła, którego nie ma nigdzie w repozytorium kodu

Dnia 13.12.2022 r. o godzinie 17:00 odbędzie się spotkanie z Panem dr. Inż. Tomaszem Krukiem w celu skonsultowania treści powyższego dokumentu.