Dokumentacja Projektu Video-Sent

Kacper Witek, Kacper Stasiak, Kacper Przybylski,
Julia Ruszer, Dawid Frontczak, Jakub Cendalski

2 grudnia 2025

Spis treści

1 Wstęp

W dokumencie przedstawiono budowę aplikacji Video-Sent. Projekt ten służy do analizy recenzji wideo pochodzących z serwisów takich jak YouTube. System składa się z dwóch głównych elementów: aplikacji klienckiej (**Frontend**) oraz serwera przetwarzającego dane (**Backend**). W dalszej części opisano wykorzystane technologie oraz zasadę działania systemu.

2 Ogólny opis systemu

Aplikacja funkcjonuje w modelu klient-serwer. Frontend komunikuje się z Backendem za pośrednictwem API. Część serwerowa została zaimplementowana w języku Python, co pozwoliło na wykorzystanie dostępnych bibliotek do przetwarzania danych. Zastosowano przetwarzanie asynchroniczne, dzięki czemu użytkownik nie oczekuje na zakończenie analizy w czasie rzeczywistym – proces ten realizowany jest w tle.

2.1 Użyte technologie

- **Frontend:** Biblioteka React 19 (Vite), wizualizacja danych przy użyciu Chart.js.
- **Backend:** Język Python 3.13 oraz framework FastAPI.
- **Baza Danych:** System zarządzania bazą danych PostgreSQL.
- **Analiza i AI:**
  - spaCy - narzędzie do analizy tekstu.
  - OpenAI API - usługa transkrypcji mowy na tekst.
- **Inne:** yt-dlp (pobieranie ścieżek audio), Pytest (testy jednostkowe).

3 Szczegóły implementacji

3.1 Frontend (Aplikacja webowa)

Warstwa prezentacji została zrealizowana jako SPA (Single Page Application) w technologii React. Główne elementy to:

- **Dashboard:** Widok prezentujący listę wykonanych analiz.
- **Widok szczegółów:** Ekran przedstawiający szczegółowe wyniki dla wybranego materiału wideo.
- **Warstwa komunikacji:** Moduły odpowiedzialne za wysyłanie żądań HTTP do serwera.

3.2 Backend API

Serwer udostępnia interfejs programistyczny (API) wykorzystywany przez frontend. Kluczowe punkty końcowe (endpoints) to:

- `POST /api/analysis/`: Przyjmuje adres URL filmu. System weryfikuje obecność filmu w bazie. W przypadku braku, tworzone jest nowe zadanie (`Job`) i inicjowany jest proces przetwarzania w tle.
- `GET /api/status/{job_id}`: Umożliwia monitorowanie postępu analizy.
- `GET /api/result/{film_id}`: Służy do pobrania finalnych wyników analizy.

3.3 Główna logika (Pipeline)

Procesem przetwarzania steruje moduł orkiestracji (plik `pipeline.py`). Realizuje on następujące etapy:

1. **Pobieranie:** Narzędzie `yt-dlp` pobiera ścieżkę audio ze wskazanego adresu URL.
2. **Transkrypcja:** Plik audio przekazywany jest do API OpenAI w celu uzyskania transkrypcji tekstowej.
3. **Analiza:** Moduł NLP przetwarza tekst, identyfikując cechy produktu (np. bateria, ekran) oraz określając sentyment wypowiedzi.
4. **Zapis:** Wyniki są zapisywane w bazie danych, a status zadania ulega aktualizacji.

3.4 Baza Danych

W projekcie wykorzystano system PostgreSQL oraz bibliotekę SQLAlchemy, co pozwala na mapowanie obiektowe relacyjnej bazy danych (ORM).

4 Struktura danych

Model danych opiera się na trzech głównych tabelach:

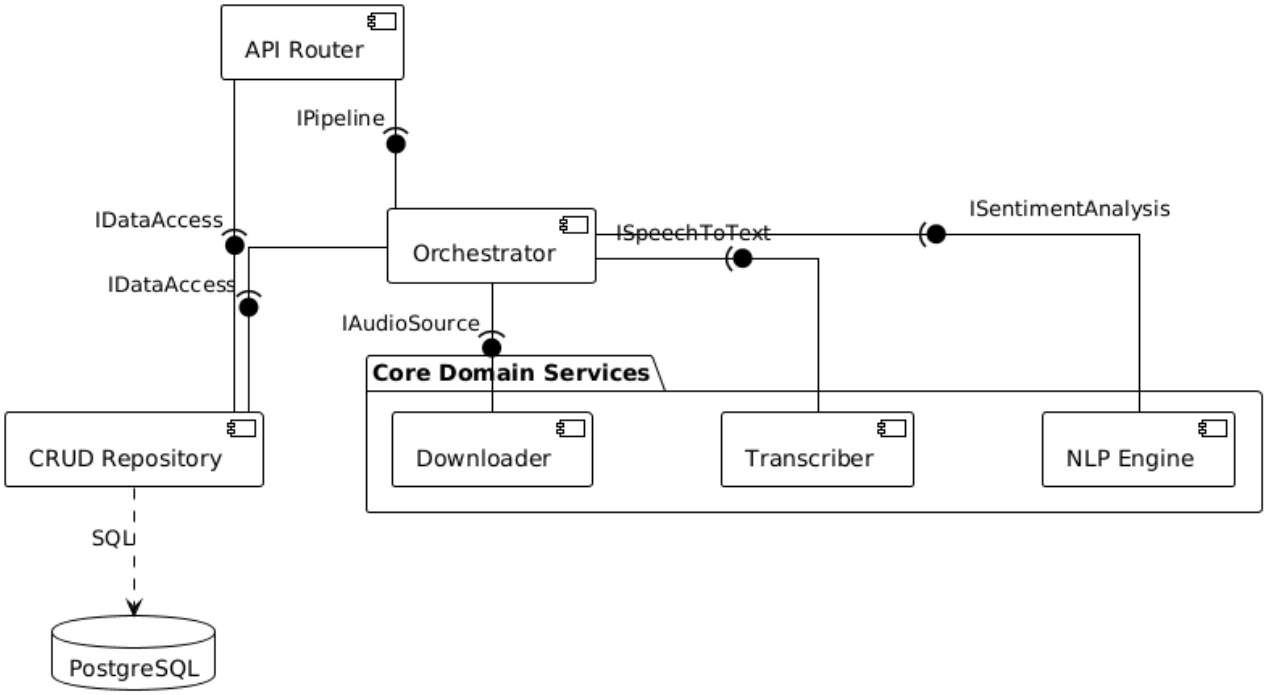
- **Jobs:** Przechowuje informacje o zadaniach, w tym ich aktualny status (np. w toku, zakończone, błąd).
- **Films:** Zawiera metadane filmu (tytuł, platforma) oraz pełną transkrypcję.
- **Analysis:** Gromadzi wyniki liczbowe analizy, takie jak oceny poszczególnych aspektów.

5 Diagramy

Poniżej przedstawiono schematy obrazujące strukturę i działanie systemu.

5.1 Architektura Backendu

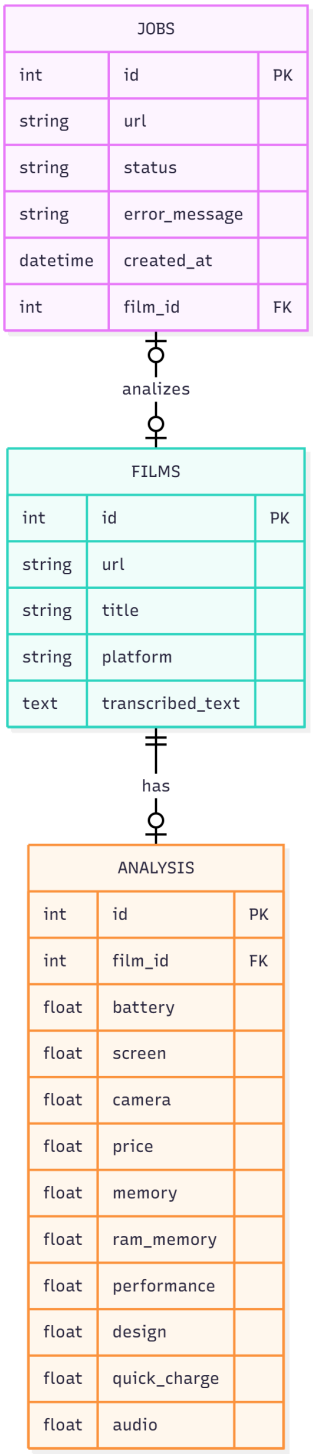
Diagram prezentuje strukturę modułową backendu, uwzględniając router API, logikę sterującą oraz poszczególne serwisy (pobieranie, transkrypcja, NLP).



Rysunek 1: Moduły Backendu

5.2 Schemat Bazy Danych (ERM)

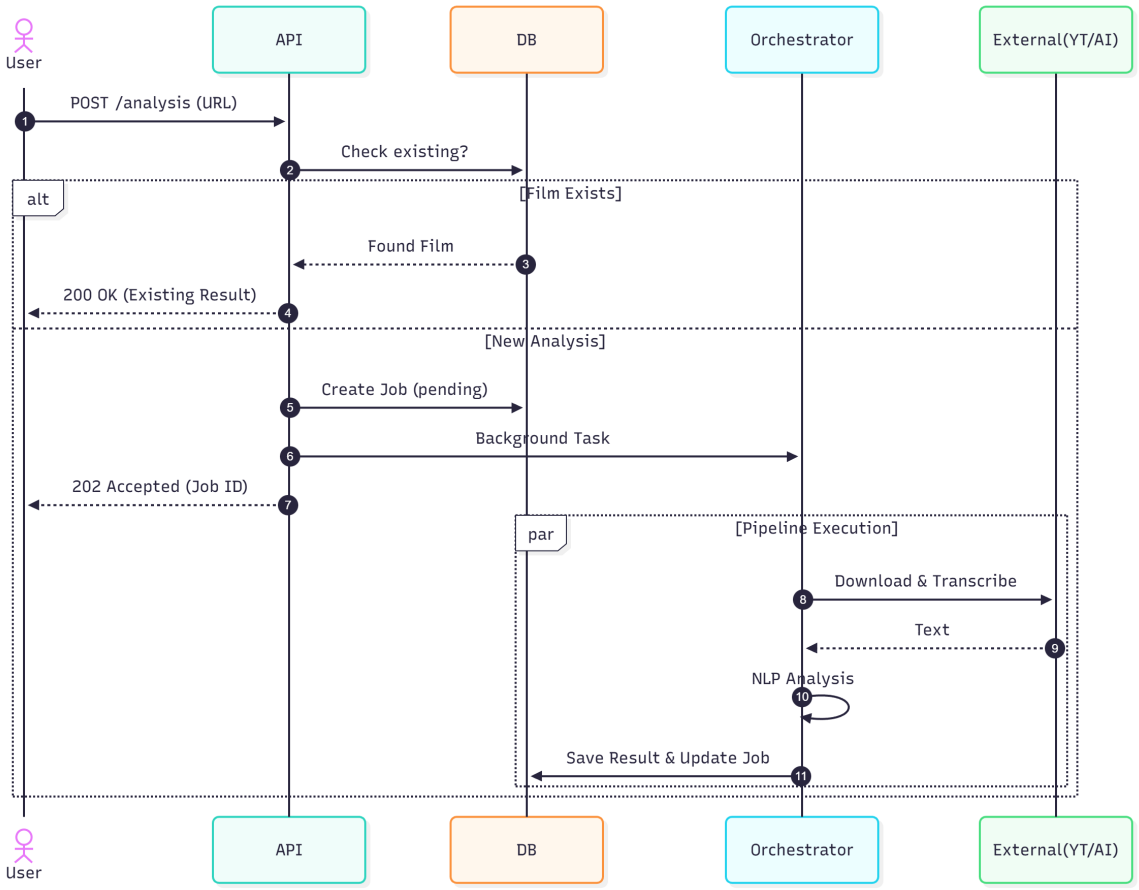
Schemat obrazuje relacje między tabelami w bazie danych. Tabela `Jobs` powiązana jest z `Films`, a wyniki analizy przechowywane są w tabeli `Analysis`.



Rysunek 2: Struktura bazy danych

5.3 Przebieg analizy (Sequence Diagram)

Diagram sekwencji ilustruje przepływ sterowania podczas procesu analizy wideo, wskazując na interakcje między komponentami systemu.



Rysunek 3: Przebieg procesu analizy

6 Testowanie

W celu weryfikacji poprawności działania systemu przygotowano testy:

- **Backend:** Testy funkcjonalne realizowane przy użyciu frameworka `pytest`.
- **Wydajność:** Testy obciążeniowe przeprowadzane za pomocą narzędzia `locust`.