Dokumentacja systemu zarzadzania zadaniami FastAPI + Keycloak + Express.js

Olha Yakymenko

1 Wprowadzenie

Aplikacja umożliwia tworzenie i zarzadzanie zadaniami przez użytkowników systemu. Uwierzytelnienie oraz autoryzacja odbywaja sie przy pomocy serwera Keycloak (OAuth2/OpenID Connect), a dane przechowywane sa w bazie PostgreSQL.

System składa sie z kilku backendów:

- FastAPI – główny backend REST API odpowiedzialny za zarzadzanie zadaniami.
- Express.js – dodatkowy backend obsługujacy zgłoszenia użytkowników o nadanie roli administratora oraz zarzadzanie tymi zgłoszeniami.

2 Architektura systemu

- **FastAPI** – backend REST API zarzadzajacy zadaniami i ich statusami.
- **Express.js** – backend do obsługi prośb o nadanie roli admina oraz zarzadzania tymi zgłoszeniami.
- **Keycloak** – serwer uwierzytelniania i autoryzacji (OAuth2).
- **PostgreSQL** – baza danych z tabelami: `tasks`, `task_status`, `admin_requests`.
- **Frontend (np. React)** – korzysta z API z autoryzacja Bearer Token.

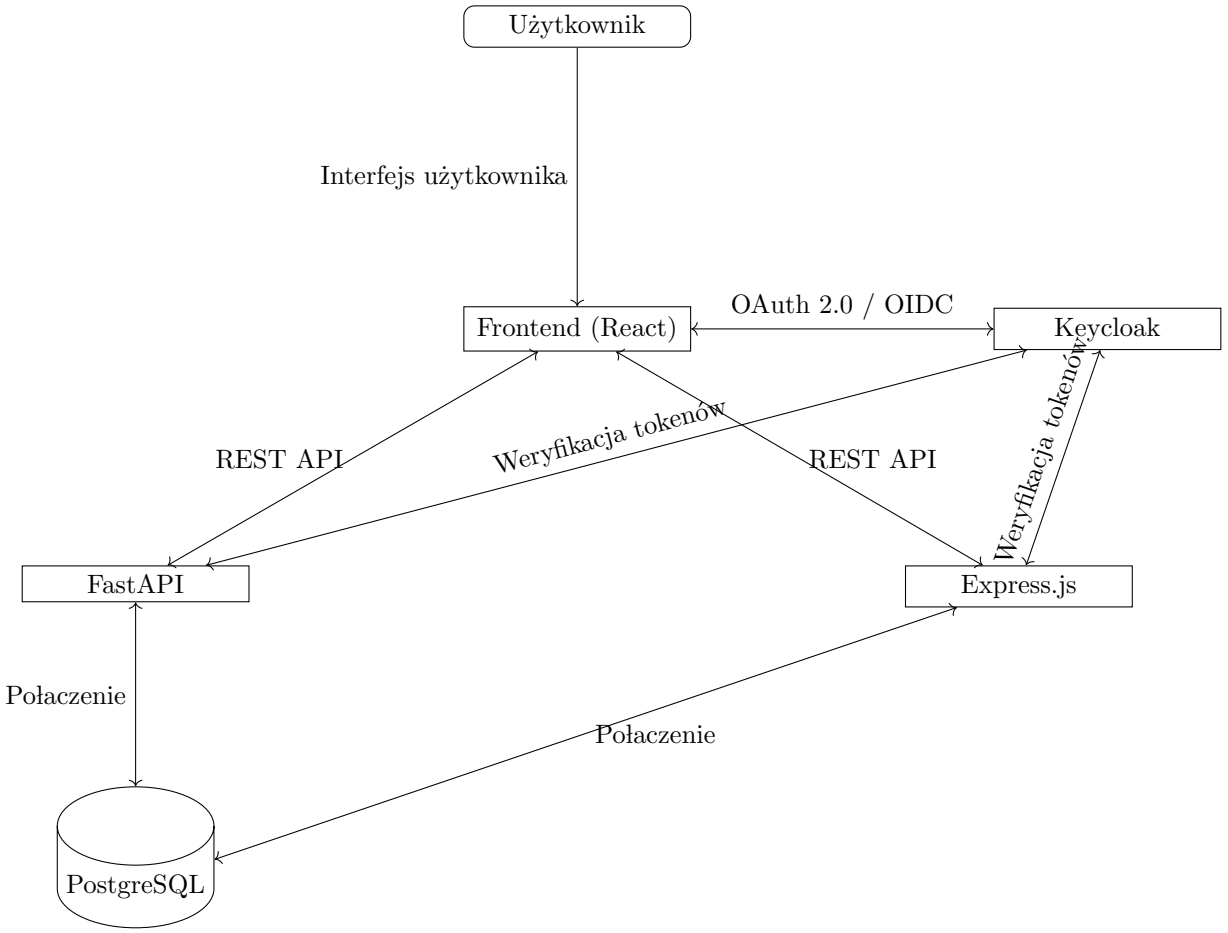
3 Architektura systemu

3.1 Komponenty

- Frontend: Aplikacja React.js

- Backend:
  - FastAPI (Python) - główna logika aplikacji
  - Express.js - obsługa próśb o uprawnienia admina
- Dostawca tożsamości: Keycloak
- Baza danych: PostgreSQL

3.2 Diagram architektury



4 Flow uwierzytelniania

System wykorzystuje OpenID Connect z Keycloak:

1. Użytkownik loguje sie przez Keycloak
2. Frontend otrzymuje tokeny (access token, refresh token)
3. Access token jest dołaczany do żadań API
4. Backend weryfikuje token przez endpoint introspekcji Keycloak
5. Dostep jest przyznawany na podstawie ról użytkownika

5 Autoryzacja

5.1 Role

- **admin** - Może tworzyć zadania, przegladać wszystkie zadania, zarzadzać prośbami o uprawnienia admina
- **user** - Może przegladać i aktualizować przypisane sobie zadania

6 Endpointy API

6.1 Endpointy FastAPI

Endpoint	Metoda	Opis
/api/task/assign	POST	Tworzenie nowego zadania (tylko admin)
/api/task	DELETE	Usuwanie zadania (tylko admin)
/api/admin/tasks	GET	Pobieranie wszystkich zadań ze statusami (tylko admin)
/api/task	PUT	Aktualizacja statusu zadania
/api/my-tasks	GET	Pobieranie zadań użytkownika
/api/summary	GET	Podsumowanie (inne dla admina i użytkownika)

6.2 Endpointy Express.js

Endpoint	Metoda	Opis
/api-ex/request-admin	POST	Prośba o uprawnienia admina
/api-ex/admin/requests	GET	Pobieranie próśb (tylko admin)
/api-ex/admin/approve	POST	Akceptacja prośby
/api-ex/admin/reject	POST	Odrzucenie prośby

7 Instalacja

7.1 Instrukcja instalacji

1. Sklonuj repozytorium:

```
git clone https://github.com/twoje-repo/manage-tasks-app.git
```

2. Skonfiguruj Keycloak:

```
cd keycloak-config
./setup-realm.sh
```

3. Uruchom usługi:

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

4. Dla wdrożenia Kubernetes:

```
kubectl apply -f k8s/
```

8 Konfiguracja

8.1 Zmienne środowiskowe

Kluczowe zmienne konfiguracyjne:

```
# Keycloak
KEYCLOAK_URL=http://keycloak:8080
REALM=manage-tasks
CLIENT_ID=manage-tasks-client
CLIENT_SECRET=twoj-sekret

# Baza danych
PGUSER=admin
PGPASSWORD_FILE=/run/secrets/db-password
PGDATABASE=tasks
PGHOST=postgres
```

9 Implementacja zabezpieczeń

9.1 Weryfikacja tokenów

System używa introspekcji tokenów Keycloak do weryfikacji:

```
def verify_token(authorization: Optional[str] = Header(None)) -> dict:
    token = authorization.replace("Bearer ", "")
    introspection_url = f"{keycloak_url}/realms/{realm}/protocol/openid-connect/

    response = requests.post(
        introspection_url,
        data={'token': token},
        auth=(client_id, client_secret)
    )
```

```
    if not response.json().get("active", False):
        raise HTTPException(status_code=401, detail="Nieprawid owy token")
```

9.2 Sprawdzanie ról

```
// Przyklad komponentu React
const { keycloak } = useKeycloak();
const isAdmin = keycloak.hasRealmRole('admin');
```

10 Backend Express.js – obsługa zgłoszeń o nadanie roli administratora

Dodatkowy backend napisany w Express.js pełni role systemu zarzadzajacego prośbami użytkowników o przyznanie im roli `admin`.

10.1 Kluczowe funkcjonalności

- **Obsługa zgłoszeń użytkowników** – użytkownicy moga zgłosić cheć zostania administratorem przez endpoint:

  ```
  POST /api-ex/request-admin
  ```

- **Zarzadzanie zgłoszeniami przez adminów** – administratorzy moga przegladać wszystkie oczekujace zgłoszenia:

  ```
  GET /api-ex/admin/requests
  ```

  oraz akceptować lub odrzucać zgłoszenia:

  ```
  POST /api-ex/admin/approve
  POST /api-ex/admin/reject
  ```

- **Automatyczne przypisywanie roli w Keycloak** – po zatwierdzeniu zgłoszenia użytkownikowi jest przypisywana rola `admin` bezpośrednio w Keycloak przez API administracyjne.

10.2 Baza danych

Tabela `admin_requests` zawiera:

- `id` – klucz główny,
- `username` – nazwa użytkownika,
- `status` – status zgłoszenia (`pending`, `approved`, `rejected`).

10.3 Mechanizm weryfikacji tokena

Backend Express.js używa introspekcji tokena JWT w Keycloak, podobnie jak backend FastAPI, aby potwierdzić ważność tokena oraz sprawdzić role użytkownika.

10.4 Integracja z Keycloak

Backend Express korzysta z konta administracyjnego Keycloak do:

- pobierania tokena administracyjnego,
- wyszukiwania użytkowników w Keycloak,
- przypisywania ról (np. roli `admin`) użytkownikom po zatwierdzeniu zgłoszenia.

11 Backend FastAPI – mikroserwis zarzadzajacy zadaniami i statystykami

Mikroserwis FastAPI odpowiada za zarzadzanie zadaniami i ich statusami, integracje z baza danych PostgreSQL oraz udostepnianie danych i statystyk poprzez REST API.

11.1 Funkcjonalności

- Pobieranie i zapisywanie danych o zadaniach oraz ich statusach w bazie PostgreSQL.
- Obsługa przypisywania zadań wielu pracownikom oraz monitorowanie statusu realizacji każdego zadania.
- Generowanie podsumowań i statystyk dotyczacych zadań, takich jak liczba zadań, ich stan wykonania oraz liczba zaangażowanych pracowników.
- Weryfikacja tokenów JWT oraz introspekcja tokenów w Keycloak dla zabezpieczenia dostepu do API.
- Kontrola dostepu oparta na rolach — niektóre endpointy dostepne tylko dla użytkowników z rola administratora.
- Obsługa CORS, umożliwiajaca komunikacje z frontendem działajacym na lokalnych serwerach.

12 Podsumowanie

System zarzadzania zadaniami wykorzystuje trzy backendy, które wspólnie realizuja funkcje aplikacji:

- FastAPI obsługuje zadania i statusy,
- Express.js zajmuje sie zgłoszeniami o nadanie ról administratora i ich zatwierdzaniem,

co zapewnia modularność, elastyczność i separacje odpowiedzialności.

Bezpieczeństwo i autoryzacja sa realizowane przez Keycloak, a trwałość danych gwarantuje PostgreSQL.