Struktura Organizacyjna Firmy

Jakub Mikusek

13 grudnia 2024

1 Wprowadzenie

W ramach projektu została stworzona aplikacja multipage application (MPA) przy użyciu framework'u Flask. Baza danych została stworzona przy użyciu bazy neo4j. Aplikacja wraz z bazą danych zostały stworzone przy pomocy dwóch kontenerów.

2 Cel projektu

Celem projektu było stworzenie systemu do tworzenia, modyfikacji oraz wizualizacji struktury organizacyjnej firmy. Aplikacja pozwala na stworzenie oddziałów o określonych zadaniach, które obejmują swoim działaniem, zespołów, które posiadają zadane obowiązki, pracowników, którzy pracują w przypisanych im zespołach oraz menadżerów, którzy zarządzają wybranym zespołem.

Aplikacja pozwala również na modyfikację parametrów menadżerów oraz pracowników, takich jak rodzaj umowy, czy przypisanie do zespołu.

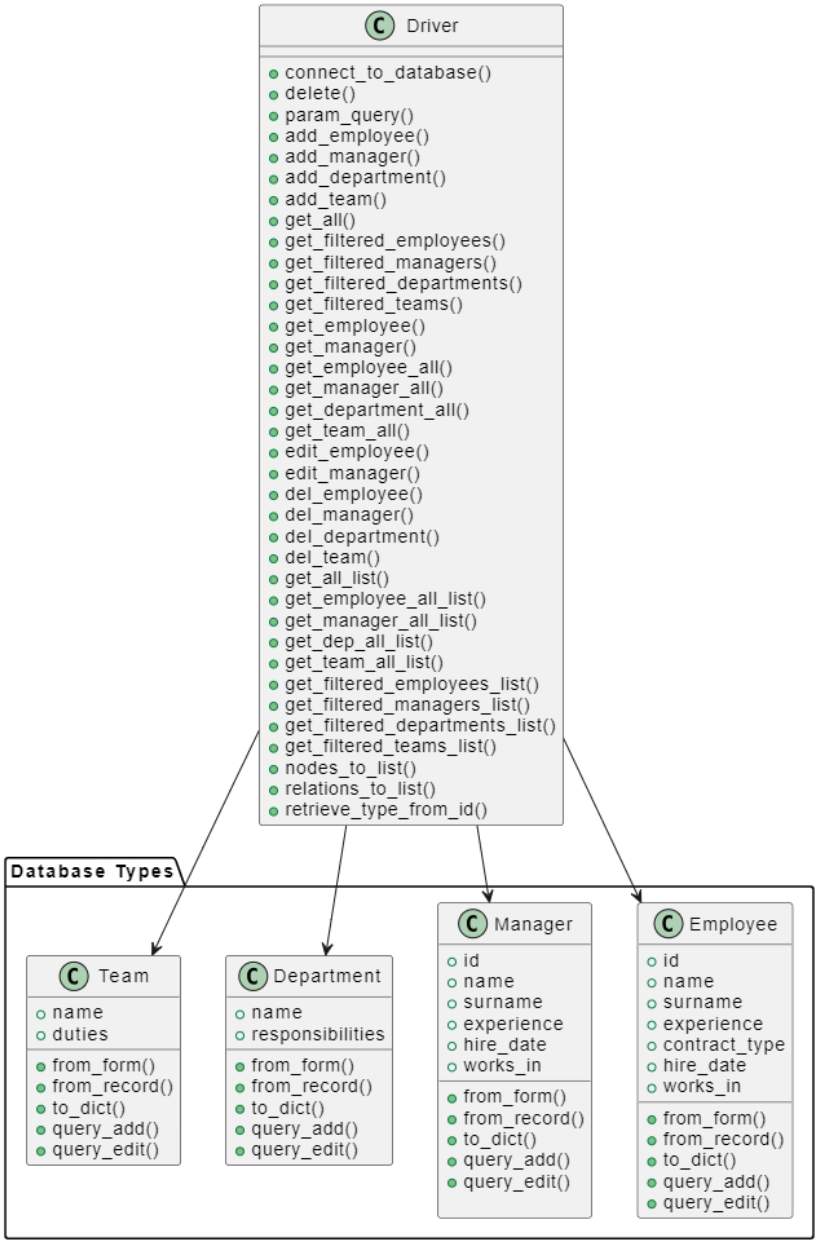
W ramach aplikacji został stworzony również widok pozwalający na filtrowanie pracowników, menadżerów, zespołów oraz oddziałów po określonych parametrach, które opisują wspomniane struktury. Dodatkowo została dodana prosta wizualizacja pozwalająca na podgląd stworzonych obiektów w bazie danych oraz relacji pomiędzy nimi.

3 Implementacja

Aplikacja użytkownika została napisana przy użyciu framework'u Flask, który pozwala przy użyciu szablonów stworzyć aplikację *MPA*. Następnie przy użyciu biblioteki *neo4j* została stworzona klasa pomocnicza odpowiadająca za połączenie z bazą danych.

Następnie przy użyciu oprogramowania *docker* zostały stworzone dwa kontenery, jeden zawierający aplikację użytkownika oraz drugi zawierający grafową bazę danych.

3.1 Diagram klas aplikacji użytkownika



Rysunek 1: Diagram klas wraz z możliwymi funkcjonalnościami oraz właściwościami struktur bazy danych

4 Grafowa baza danych

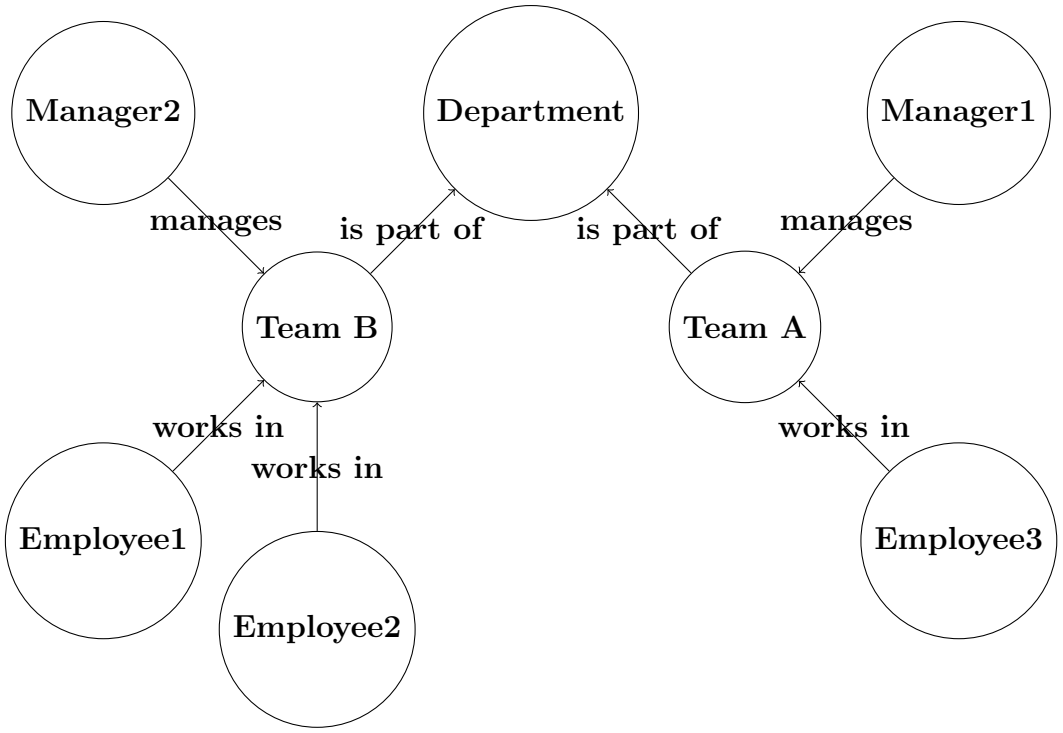
W ramach projektu została użyta grafowa baza danych *neo4j*, do której przy użyciu klasy pomocniczej *Driver* możliwe jest wykonywanie zapytań do tworzenia, modyfikacji, usuwania oraz wyszukiwania poszczególnych struktur.

4.1 Przykładowe zapytania

- **MATCH (e:Employee) RETURN e**
- **MATCH (d:Department {id: $id}) DETACH DELETE d**
- **CREATE (:Manager {id: $id, name: $name, surname: $surname, experience: $experience, hire_date: $hire_date}) WITH $id AS man_id MATCH (m:Manager {id: man_id}) MATCH (t:Team {id: $manages}) CREATE (m)-[:MANAGES]-(t)**
- **MATCH (e:Employee {: $id}) SET e.name = $name, e.surname = $surname, e.experience = $experience, e.contract_type = $contract_type WITH $id AS emp_id MATCH (e:Employee {id: emp_id})-[r:WORKS_IN]-() DELETE r WITH $id AS emp_id MATCH (e:Employee {id: emp_id}) MATCH (t:Team {id: $works_in}) CREATE (e)-[:WORKS_IN]-(t)**

Zapytania do bazy danych obejmujące wyszukiwanie tworzone są dynamicznie w zależności od parametrów wybranych przez użytkownika.

4.2 Schemat bazy danych



5 Jak uruchomić

Pierwsze uruchomienie W celu uruchomienia aplikacji wraz z bazą danych należy stworzyć plik *neo4j_auth.txt* zawierający nazwę użytkownika oraz hasło służące do zalogowania do bazy danych.

Przykładowa zawartość pliku: *nazwa/hasło*.

Po utworzeniu pliku konfiguracyjnego dane użytkownika z folderu głównego projektu należy uruchomić komendę *docker compose up -d* w celu zbudowania bazy danych wraz z aplikacją użytkownika. Po skonfigurowaniu kontenerów pod adresem *127.0.0.1:2137* zostanie uruchomiona aplikacja użytkownika.

Kolejne uruchomienia Aby ponownie uruchomić aplikację zawartość pliku *neo4j_auth.txt* musi być identyczna jak w przypadku pierwszego uruchomienia, następnie w celu uruchomienia aplikacji należy użyć komendy *docker compose up -d*.