Dokumentacja projektu z przedmiotu Przetwarzanie danych w chmurach obliczeniowych

Małgorzata Makieła

11 grudnia 2024

Spis treści

1 Wstęp

1.1 Cel projektu

Celem projektu było wykorzystanie grafowej bazy danych Neo4j oraz utworzenie interfejsu dostępu do tej bazy.

1.2 Zakres projektu

W ramach projektu wykorzystałam bazę grafową do przedstawienia zależności pomiędzy niektórymi elementami z uniwersum Wiedźmina. Natomiast w ramach interfejsu dostępu do bazy jest możliwość dodania węzła i relacji, wyszukanie relacji pomiędzy dwoma węzłami oraz wyszukanie wszystkich relacji dla węzła.

2 Użyte technologie

2.1 Baza danych

W projekcie wykorzystałam lokalną instancję bazy Neo4j, która pozwala przechowywać dane w postaci grafu - węzłów i relacji między nimi., dzięki czemu można dokładnie przedstawić zależności w świecie przedstawionym Wiedźmina.

2.2 Frontend

Do frontendu wykorzystałam język Javascript z frameworkiem React. Stworzyłam proste i intuicyjne UI umożliwiające użytkownikowy w łatwy sposób dodać i wyszukać informacje w bazie.

2.3 Backend

Do backendu wykorzystałam język Python i frmawork FastAPI ze względu na prostotę użycia i jego czytelność. Z jego użyciem utworzyłam połączenie między frontentem a bazą danych, co umożliwiło dodawanie węzłów i relacji oraz ich wyszukiwanie w bazie poprzez endpointy HTTP.

3 Funkcjonalność aplikacji

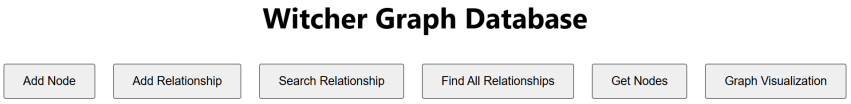
3.1 Rodzaje węzłów

Aplikacja ma zdefiniowane 6 rodzajów węzłów:

1. Postać (character), z polami takimi jak imię (i nazwisko), alias, rok urodzenia, narodowość, profesja, rasa, oraz status
2. Grupa (Group) z jednym atrubutem - nazwą
3. Umiejętność (Ability) - z nazwą oraz krótkim opisem
4. Znak (Sign) - określa znaki magiczne używane przez wiedźminów
5. Typ Potwora (MonsterType) - grupa określająca typ potworów
6. Potwór (Monster) - element należący do konkretnego typu potwora

Pomiędzy węzłami występują już różne relacje jak np FRIEND, HAS_ABILITY, BELONGS_TO, jest też możliwość tworzenia nowych rodzajów relacji.

3.2 Przewodnik po aplikacji

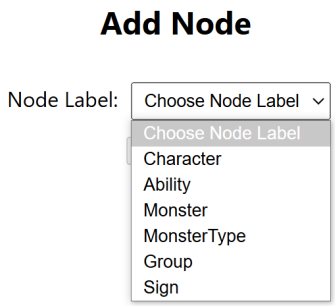


Rysunek 1: Główne okno aplikacji

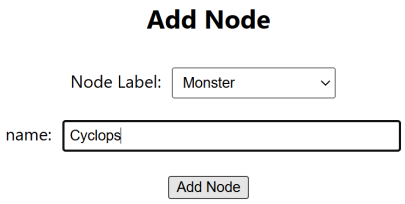
Aplikacja oferuje 6 funkcji, po kolei od lewej:

3.2.1 Dodawanie węzła

Po kliknięciu przycisku **Add Node** wyświetli się okno, gdzie wybrać należy rodzaj węzła jaki chcemy dodać. Spowoduje to rozwinięcie się formularza do wpisania atrybutów dla węzła.



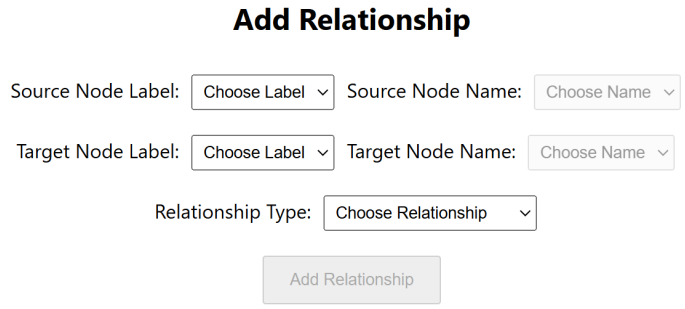
Rysunek 2: Dodawanie węzła



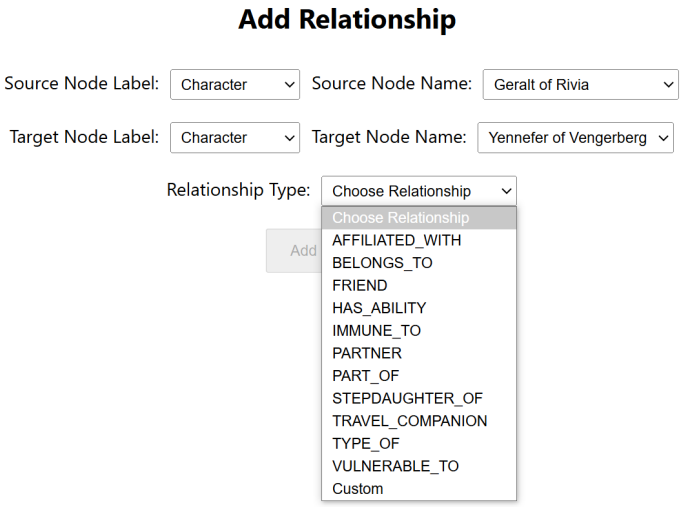
Rysunek 3: Formularz atrybutów dla węzła Monster

3.2.2 Dodawanie relacji

W celu relacji należy wybrać typ węzła dla 'początku' i 'końca' relacji oraz wybrać z dostępnych w bazie opcji konkretny węzeł. Następnie rozwiną się opcje wyboru dostępnych relacji, jest tez opcja dodania nowej relacji (Custom) o nowej nazwie.



Rysunek 4: Okno dodawania relacji

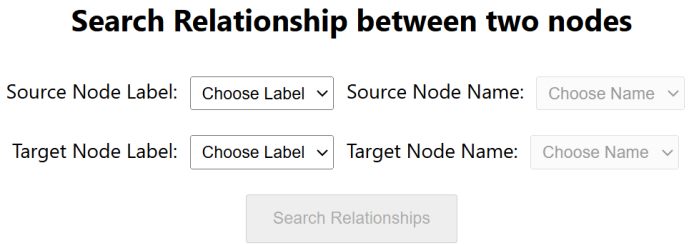


Rysunek 5: Przykładowe wypełnienie

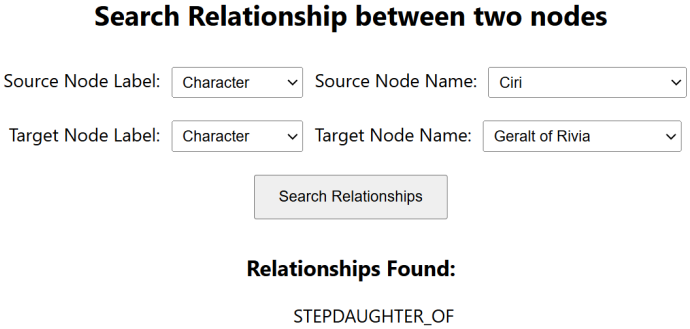
3.2.3 Wyszukanie relacji między węzłami

Do wyszukania relacji między dwoma węzłami, podobnie jak przy dodawaniu relacji należy wybrać rodzaj a następnie konkretny węzeł. Zwrócona zostanie

wyszukana relacja, lub komunikat o braku relacji, jeśli węzły nie są połączone.



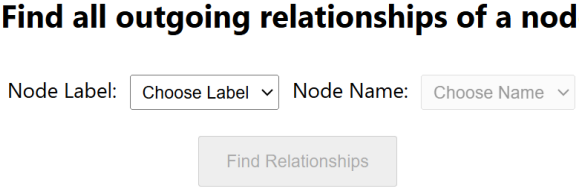
Rysunek 6: Okno wyszukiwania relacji między węzłami



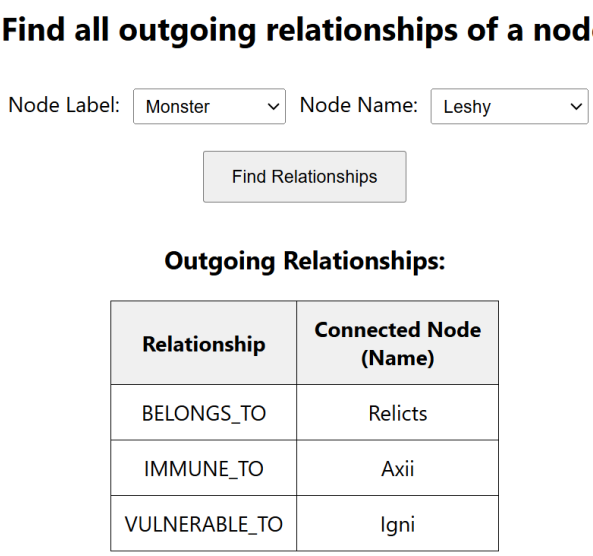
Rysunek 7: Przykładowe wypełnienie i wynik

3.2.4 Wyszukanie wszystkich relacji dla danego węzła

Aplikacje oferuje tez możliwość wyszukania wszystkich relacji wychodzących dla konkretnego węzła.



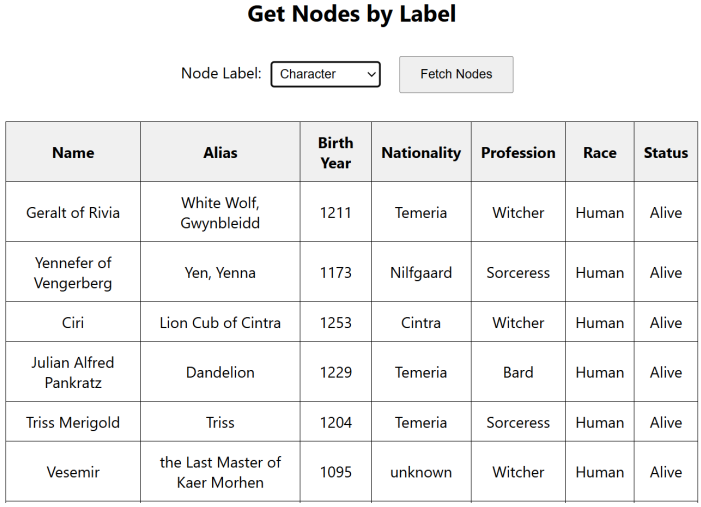
Rysunek 8: Okno wyszukiwania relacji dla węzła



Rysunek 9: Przykładowe wypełnienie i wynik

3.2.5 Pobranie danych o węzłach

W celach czysto informacyjnych jest mozliwość wyswietlenia aktualnie przechowywanych w bazie węzłów danego typu z ich atrybutami



Rysunek 10: Get Nodes

3.2.6 Wizualizacja grafu

Przedstawia ogólny obraz bazy uzyskany dzięki Neo4j browser.

4 Uruchomienie aplikacji

4.1 Przygotowanie środowiska

Do uruchomienia aplikacji potrzebne będą:

- Python 3.8 lub wyżej oraz pip
- fastapi (pip install fastapi)
- uvicorn (pip install uvicorn)
- neo4j-driver (pip install neo4j-driver)
- Node.js
- axios (npm install axios)
- react-router-dom (npm install react-router-dom)
- Neo4j

Następnie należy uruchomić instację bazy neo4j oraz uruchomić backend oraz forntend

1. Z poziomu głównego folderu projektu należy uruchomić wirtualne środowisko w Pythonie

   ```
   .\venv\Scripts\activate
   ```

   a następnie uruchomić backend

   ```
   uvicorn main:app --reload
   ```

   Działanie backendu można testować na http://127.0.0.1:8000

2. Następnie w celu uruchomienia frontendu należy przejść do folderu witcher-app\src

   ```
   cd witcher-app\src
   npm start
   ```

   komenda 'npm start' automatycznie otworzy okno aplikacji w przeglądarce pod adresem http://localhost:3000