

# Dokumentacja projektu z przedmiotu Przetwarzanie danych w chmurach obliczeniowych

Małgorzata Makiela

11 grudnia 2024

## Spis treści

<b>1 Wstęp</b>	<b>1</b>
1.1 Cel projektu . . . . .	1
1.2 Zakres projektu . . . . .	2
<b>2 Użyte technologie</b>	<b>2</b>
2.1 Baza danych . . . . .	2
2.2 Frontend . . . . .	2
2.3 Backend . . . . .	2
<b>3 Funkcjonalność aplikacji</b>	<b>2</b>
3.1 Rodzaje węzłów . . . . .	2
3.2 Przewodnik po aplikacji . . . . .	3
3.2.1 Dodawanie węzła . . . . .	3
3.2.2 Dodawanie relacji . . . . .	4
3.2.3 Wyszukanie relacji między węzłami . . . . .	4
3.2.4 Wyszukanie wszystkich relacji dla danego węzła . . . . .	5
3.2.5 Pobranie danych o węzłach . . . . .	6
3.2.6 Wizualizacja grafu . . . . .	7
<b>4 Uruchomienie aplikacji</b>	<b>7</b>
4.1 Przygotowanie środowiska . . . . .	7

## 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 użytkownikowi w łatwy sposób dodać i wyszukać informacje w bazie.

### 2.3 Backend

Do backendu wykorzystałam język Python i framework FastAPI ze względu na prostotę użycia i jego czytelność. Z jego użyciem utworzyłam połączenie między frontendem 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 atrybutem - 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

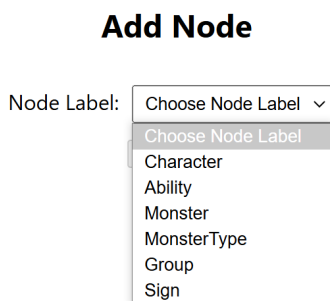


Rysunek 1: Główne okno aplikacji

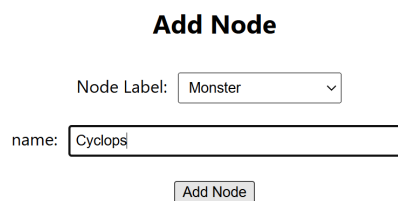
Aplikacja oferuje 6 funkcji, po kolei od lewej:

### 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 też opcja dodania nowej relacji (Custom) o nowej nazwie.

**Add Relationship**

Source Node Label:  Source Node Name:

Target Node Label:  Target Node Name:

Relationship Type:

Rysunek 4: Okno dodawania relacji

**Add Relationship**

Source Node Label:  Source Node Name:

Target Node Label:  Target Node Name:

Relationship Type:

Choose Relationship

AFFILIATED\_WITH

BELONGS\_TO

FRIEND

HAS\_ABILITY

IMMUNE\_TO

PARTNER

PART\_OF

STEPDAUGHTER\_OF

TRAVEL\_COMPANION

TYPE\_OF

VULNERABLE\_TO

Custom

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.

### Search Relationship between two nodes

Source Node Label:  Source Node Name:

Target Node Label:  Target Node Name:

Rysunek 6: Okno wyszukiwania relacji między węzłami

### Search Relationship between two nodes

Source Node Label:  Source Node Name:

Target Node Label:  Target Node Name:

#### Relationships Found:

STEPDAUGHTER\_OF

Rysunek 7: Przykładowe wypełnienie i wynik

#### 3.2.4 Wyszukanie wszystkich relacji dla danego węzła

Aplikacja oferuje też możliwość wyszukiwania wszystkich relacji wychodzących dla konkretnego węzła.

### Find all outgoing relationships of a node

Node Label:  Node Name:

Rysunek 8: Okno wyszukiwania relacji dla węzła

## Find all outgoing relationships of a node

Node Label:  Node Name:

### Outgoing Relationships:

Relationship	Connected Node (Name)
BELONGS_TO	Relicts
IMMUNE_TO	Axii
VULNERABLE_TO	Igni

Rysunek 9: Przykładowe wypełnienie i wynik

### 3.2.5 Pobranie danych o węzłach

W celach czysto informacyjnych jest możliwość wyświetlenia aktualnie przecho-  
wywanych w bazie węzłów danego typu z ich atrybutami

### Get Nodes by Label

Node Label:

Name	Alias	Birth Year	Nationality	Profession	Race	Status
Geralt of Rivia	White Wolf, Gwynbleidd	1211	Temeria	Witcher	Human	Alive
Yennefer of Vengerberg	Yen, Yenna	1173	Nilfgaard	Sorceress	Human	Alive
Ciri	Lion Cub of Cintra	1253	Cintra	Witcher	Human	Alive
Julian Alfred Pankratz	Dandelion	1229	Temeria	Bard	Human	Alive
Triss Merigold	Triss	1204	Temeria	Sorceress	Human	Alive
Vesemir	the Last Master of Kaer Morhen	1095	unknown	Witcher	Human	Alive

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 instancję bazy neo4j oraz uruchomić backend oraz frontend

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>