Grafowa Baza danych - Śledztwo

Dokumentacja Projektu

Jan Pazdan

12 Grudnia 2025

Spis treści

1 Wprowadzenie

Dokument opisuje funkcjonalność aplikacji webowej służącej do wizualizacji, tworzenia oraz zarządzania grafem śledczym przechowywanym w bazie Neo4j. System umożliwia użytkownikowi dodawanie węzłów określonych typów, tworzenie relacji między nimi, edycję węzłów oraz podgląd struktury grafu z poziomu wizualizacji. Zastosowana architektura opiera się na backendzie w Node.js (Express) oraz prostym frontendzie HTML/JS, korzystającym z biblioteki *vis-network* do rysowania grafu.

2 Struktura projektu

Projekt składa się z poniższych elementów:

- **server.js** – serwer aplikacji oparty na Express, udostępniający REST API do komunikacji z Neo4j.
- **public/index.html** – główny interfejs użytkownika.
- **public/script.js** – logika frontendowa: wysyłanie zapytań do API, aktualizacja wizualizacji, obsługa formularzy.
- **.env** – konfiguracja połączenia z Neo4j.
- **package.json** – deklaracja zależności i skryptów projektu.

Serwer udostępnia statyczne pliki z katalogu `public`, dzięki czemu aplikację otwiera się przez odwołanie do lokalnego adresu serwera.

3 Model danych w Neo4j

Aplikacja operuje na pięciu typach węzłów:

- **Victim**
- **Witness**
- **Suspect**
- **CrimeScene**
- **Evidence**

Każdy węzeł posiada właściwość `name` oraz `ID`, będące unikalną nazwą wyświetlaną w grafie oraz unikalnym identyfikatorem zaczynającym się od pierwszej litery typu oraz liczby porządkowej np. S1, W1 itp.

3.1 Relacje

System pozwala użytkownikowi na tworzenie wyłącznie relacji zgodnych z predefiniowanymi regułami:

- KILLED_AT: Victim → CrimeScene
- WITNESSED: Witness → Suspect
- WAS_AT: Witness → CrimeScene
- FOUND_AT: Evidence → CrimeScene
- EVIDENCE_OF: Evidence → Suspect

Dzięki temu użytkownik nie może utworzyć niepoprawnego połączenia.

3.2 Praktyczna realizacja

Relacje w przypadku śledztwa oznaczają:

- KILLED_AT: Osoba *Victim* została zabita w miejscu zbrodni *CrimeScene*
- WITNESSED: Osoba *Witness* świadczy o osobie *Suspect*
- WAS_AT: Osoba *Witness* była na miejscu zbrodni *CrimeScene*
- FOUND_AT: Dowód *Evidence* został znaleziony na miejscu zbrodni *CrimeScene*
- EVIDENCE_OF: Dowód *Evidence* należy do osoby *Suspect*

4 Funkcjonalność frontendowa

4.1 Dodawanie węzłów

Użytkownik wybiera typ węzła z listy rozwijanej oraz wpisuje nazwę, która stanie się wartością właściwości `name`. Po zatwierdzeniu:

1. wysyłane jest żądanie do API,
2. tworzony jest węzeł w Neo4j,
3. graf jest odświeżany,
4. nowy węzeł pojawia się w wizualizacji.

Kolor węzła zależy od jego typu; każdy typ ma przypisany unikalny kolor.

4.2 Tworzenie relacji

Po wybraniu rodzaju relacji użytkownikowi prezentowane są dwa pola pozwalające wpisać:

- ID węzła źródłowego,
- ID węzła docelowego.

Frontend pilnuje, aby użytkownik mógł wybrać relację tylko z dostępnej listy reguł. Następnie serwer:

1. weryfikuje poprawność typów,
2. tworzy relację w Neo4j,
3. zwraca wynik do aplikacji,
4. graf jest odświeżany.

4.3 Edycja węzłów

Dostępna jest funkcja edycji, która umożliwia zmianę nazwy istniejącego węzła. Użytkownik podaje:

- ID węzła,
- nową nazwę.

Po edycji interfejs odświeża wizualizację, a zmiana staje się widoczna natychmiast.

4.4 Widok grafu

Wizualizacja oparta jest na bibliotece `vis-network`. Umożliwia:

- dynamiczne przesuwanie grafu,
- przybliżanie i oddalanie,
- podświetlanie powiązań,
- odświeżanie po każdej operacji na węzłach lub relacjach.

Rys. 4.1: Ilustracja przedstawia diagram grafowy przykładowych danych.

5 Legenda grafu

Aplikacja generuje półprzezroczystą legendę przedstawiającą:

- każdy rodzaj relacji,
- kolor węzła źródłowego,
- kolor węzła docelowego.

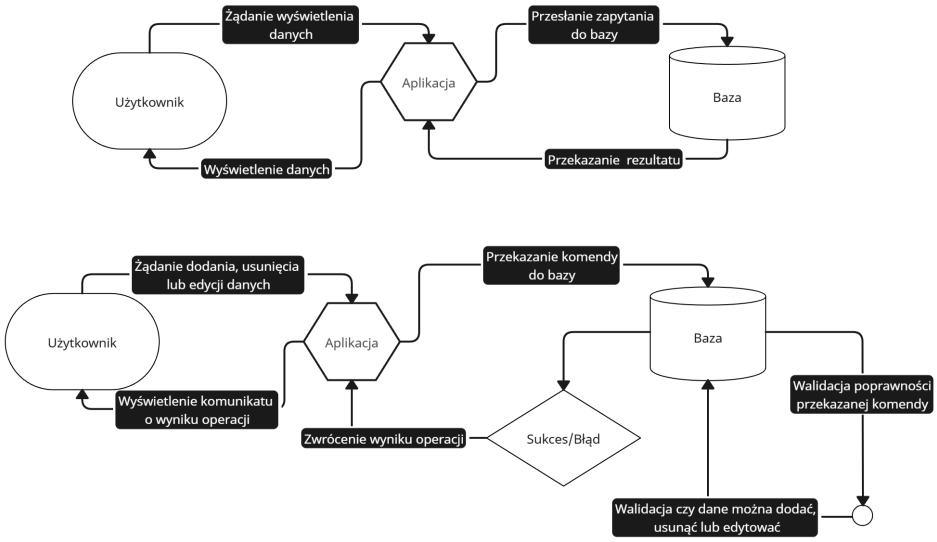
Legenda umieszczana jest bezpośrednio w obszarze wizualizacji, aby użytkownik miał natychmiastowy wgląd w strukturę i znaczenie kolorów. Każdy wpis legendy prezentuje małe kolorowe markery symbolizujące typy węzłów powiązanych relacją.

6 Funkcjonalność backendowa

Backend udostępnia zestaw endpointów REST API:

- tworzenie nowego węzła,
- tworzenie relacji,
- edycja węzła,
- pobieranie wszystkich węzłów i relacji,
- walidacja danych wejściowych.

Serwer łączy się z Neo4j przy użyciu oficjalnego sterownika `neo4j-driver`. Dane zwracane przez API są przesyłane w formacie JSON i następnie renderowane w grafie frontendowym.



Rys. 6.1: Diagram przepływu danych (DFD) realizowany w tym projekcie.

7 Podsumowanie

Projekt umożliwia pełne zarządzanie grafem śledczym przechowywanym w bazie Neo4j. Zapewnia:

- intuicyjne dodawanie węzłów,
- tworzenie tylko poprawnych relacji,
- edycję istniejących elementów,
- pełną wizualizację grafu,
- przejrzystą legendę relacji i typów węzłów.

System może stanowić podstawę bardziej rozbudowanych aplikacji analitycznych lub śledczych, wykorzystujących technologie grafowe do prezentacji złożonych zależności.