POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Kamil Krzysztof Kuźnicki

Nr albumu s21815

Interfejs wirtualnej rzeczywistości dla grafowych baz danych

Praca magisterska napisana

pod kierunkiem:

dr inż. Mariusz Trzaska

Warszawa, czerwiec 2022

Streszczenie

Niniejsza praca porusza temat interfejsu wirtualnej rzeczywistości dla grafowych baz danych. Na rynku jest wiele narzędzi ułatwiających obsługę i wizualizację danych pochodzących z tego typu baz. Niewiele z nich pozwala na wykorzystanie w tym celu wirtualnej rzeczywistości. Przeanalizowano dostępne na rynku aplikacje oferujące taką możliwość.

Rezultatem pracy jest wynik analizy wymagań i cech pożądanych w takich aplikacjach. Zaproponowano nowe rozwiązanie, które stara się implementować większość z nich. Prototyp został wykonany w technologiach webowych. Może być obsługiwany z użyciem urządzenia VR lub klawiatury i myszy. Domyślnie umożliwia połączenie z bazami danych Neo4J i ArangoDB. Dzięki generycznej implementacji obsługi bazy danych, lista wspieranych technologii może być łatwo rozszerzana przez programistów. Użytkownik może między innymi wykonywać zapytania do bazy danych, zapisywać je oraz dostosowywać sposób w jaki prezentowane są ich wyniki. Postarano się aby korzystanie z aplikacji w trybie wirtualnej rzeczywistości było komfortowe. W tym celu zaimplementowano trzy sposoby poruszania się oraz mechanizmy wspomagające wprowadzanie danych takie jak predykcja tekstu czy konwersja mowy na tekst.

Słowa kluczowe:

wirtualna rzeczywistość, grafowe bazy danych, wizualizacja 3D

Spis treści

1. Wstęp

Technologia wirtualnej rzeczywistości (ang. virtual reality – VR) w ostatnich latach bardzo dynamicznie się rozwija. Urządzenia VR śledzące ruch głowy i rąk rozszerzają możliwości interakcji użytkownika z aplikacjami względem płaskich ekranów. Dzięki temu technologia ta znajduje coraz więcej zastosowań w różnych dziedzinach. Niniejsza praca skupia się na wykorzystaniu wirtualnej rzeczywistości jako interfejsu do grafowych baz danych.

1.1. Interfejs VR dla grafowych baz danych

Grafowe bazy danych przechowują informacje w postaci grafu. Wierzchołki (węzły) są wykorzystywane do reprezentacji encji, a krawędzie łączące wierzchołki do zachodzących między nimi relacji. Węzły i relacje posiadają swoje identyfikatory i zbiory atrybutów.

Do zalet grafowych baz można zaliczyć szybsze wykonywanie zapytań dzięki unikaniu złączeń i uproszczone modelowanie danych. Aby znaleźć połączenia w bazach relacyjnych trzeba użyć operacji złączenia. Polega to na znalezieniu wartości z jednej tabeli w innej tabeli. Może to być pracochłonne, jeżeli złączane będą duże tabele. Ponadto w przypadku relacji wiele-do-wielu konieczne jest utworzenie tabeli pośredniej. W grafowej bazie danych połączenia są modelowane z wykorzystaniem krawędzi [1].

Ze względu na przechowywanie danych w postaci grafu, najwygodniejszym sposobem prezentacji danych przy analizie wyników zapytania jest forma wizualna. Na rynku istnieje wiele rozwiązań, które to umożliwiają. Niewiele z nich jednak wykorzystuje do tego wirtualną rzeczywistość.

Wirtualna rzeczywistość to komputerowa symulacja trójwymiarowych światów. Zamiast oglądać je przez ekran monitora, użytkownik korzystający z urządzenia VR zostaje w nich umieszczony. Urządzenie VR najczęściej składa się z wyświetlacza nagłownego (ang. head mounted display – HMD) i kontrolerów, za pomocą których użytkownik podejmuje interakcje z otoczeniem.

Wirtualna rzeczywistość pozwala na bardziej naturalną dla człowieka prezentację grafu w przestrzeni niż rzutowanie trójwymiarowego obrazu na płaski ekran. Wykorzystanie HMD umożliwia również szybsze lokalizowanie konkretnych fragmentów grafu.

W oparciu o analizę istniejących rozwiązań przedstawiono cechy jakimi powinny charakteryzować się aplikacje stanowiące interfejs VR dla grafowych baz danych. Zaproponowano rozwiązanie, które stara się spełniać założone wymagania.

1.2. Cel pracy

Celem pracy jest analiza problematyki interfejsów dla grafowych baz danych wykorzystujących wirtualną rzeczywistość.

1.3. Rezultaty pracy

Rezultatem pracy jest wynik analizy wymagań i cech pożądanych w aplikacjach stanowiących interfejs wirtualnej rzeczywistości dla grafowych baz danych. Zaproponowano prototyp rozwiązania, który przy właściwym rozwoju spełni wymagania stawiane przed tego typu aplikacjami.

Rozwiązanie umożliwia budowanie zapytań oraz przeglądanie wyników zapytania w formie grafu. Sposób w jaki są prezentowane dane może być dostosowany przez użytkownika. Prototyp został zaimplementowany w taki sposób, aby można go było łatwo rozszerzyć o obsługę kolejnych baz danych.

1.4. Organizacja pracy

W drugim rozdziale przedstawiono istniejące na rynku aplikacje o podobnym zastosowaniu. Pokreślono w nim możliwości, które oferują i ich słabe strony.

W trzecim rozdziale przedstawiono cechy pożądane w tego typu aplikacjach. Została również zawarta propozycja nowego rozwiązania.

Czwarty rozdział stanowi opis języków, bibliotek oraz technologii, które zostały wykorzystane w prototypie.

Piąty rozdział przedstawia architekturę rozwiązania. Zawiera również szczegóły dotyczące danych przechowywanych w bazie danych aplikacji oraz integracji z wizualizowanymi bazami danych.

W szóstym rozdziale wymieniono najważniejsze możliwości, które oferuje aplikacja kliencka. Wyjaśniono kwestię poruszania się w aplikacji, możliwych interakcji z grafem oraz przedstawiono elementy interfejsu użytkownika.

W rozdziale siódmym zawarto podsumowanie pracy. Wskazano wady i zalety rozwiązania oraz możliwe kierunki rozwoju.

2. Istniejące rozwiązania

W rozdziale zostały przedstawione istniejące na rynku rozwiązania, które pozwalają na wizualizację grafowych baz danych. Każde z nich oferuje w tym celu wykorzystanie technologii wirtualnej rzeczywistości.

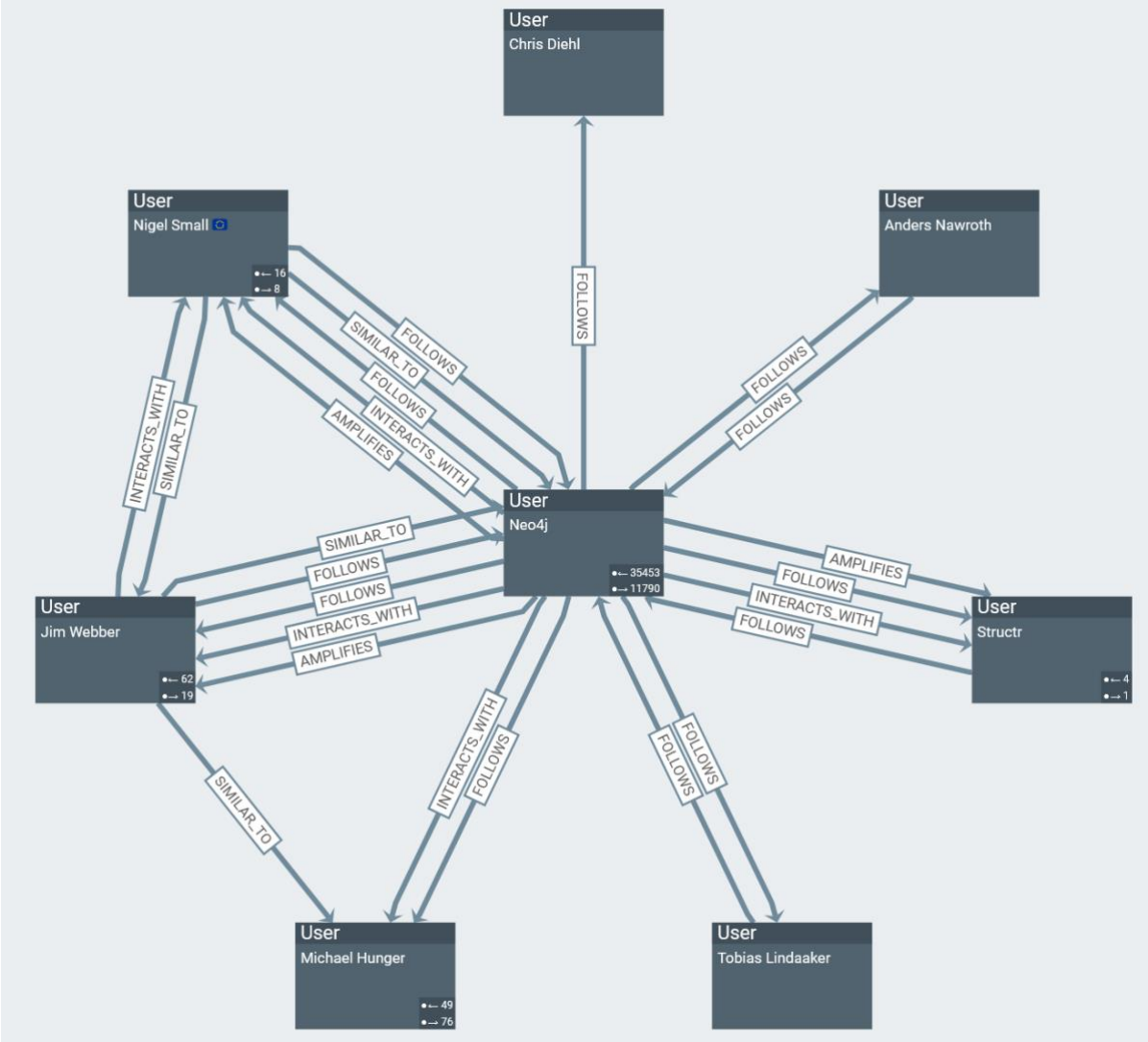
2.1. yWorks yEd Live

Firma yWorks ma w swoim portfolio szereg rozwiązań do wizualizacji grafów. Jednym z nich jest webowa aplikacja yEd Live [2]. Pozwala ona na szybkie tworzenie, eksport i udostępnianie grafów. Aplikacja umożliwia importowanie i zapis grafu w formacie .graphml (format pliku oparty o XML przechowujący informacje o grafie). Import jest też możliwy dla formatów takich jak:

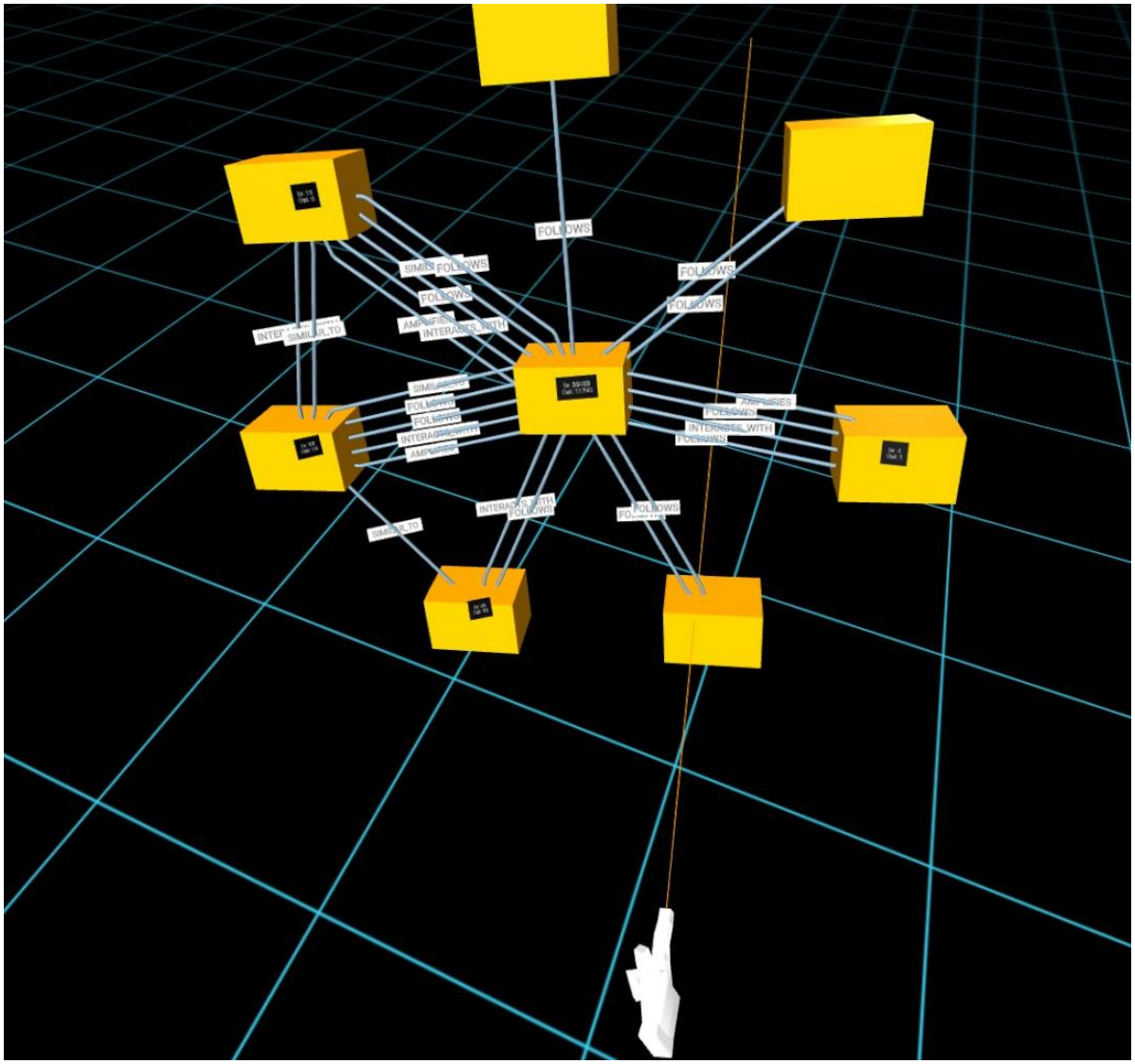
- .ygf – binarny format pliku przechowujący informacje o grafie utworzonym w yFiles [3] (rozwiązaniu firmy yWorks)
- .xls – rozszerzenie pliku utworzonego w arkuszu kalkulacyjnym Microsoft Excel
- .gml – plik zapisany w języku GML (Geography Markup Language). GML jest oparty o XML i przechowuje dane geograficzne,

Nie jest możliwe bezpośrednie połączenie się z bazą danych. Aby móc przeglądać zawartość bazy danych Neo4j trzeba skorzystać z kolejnego webowego rozwiązania od yWorks - Data Explorer for Neo4j [4]. Pozwala ono na nawiązanie połączenia z własną lub przykładową bazą danych i przeglądanie zawartości bazy z poziomu przeglądarki. Rysunek 1 przedstawia graf wygenerowany przez aplikację dla jednej z przykładowych baz danych – „Twitter". Po wyświetleniu grafu, który nas interesuje można skorzystać z opcji "Open Graph in yEd Live". Po kliknięciu opcji zostaje otworzona nowa karta z aplikacją yEd Live, w której jest ładowany poprzednio utworzony graf.

Rozwiązanie posiada eksperymentalną funkcjonalność wyświetlania grafu w 3D. Po otworzeniu tego trybu można wykorzystać urządzenie VR do przeglądania grafu (Rysunek 2). Rozwiązanie wspiera urządzenia Oculus i HTC Vive.



Rysunek 1. Graf utworzony w Data Explorer for Neo4j na podstawie danych z przykładowej bazy "Twitter"



Rysunek 2. Graf na podstawie danych z przykładowej bazy "Twitter" po eksporcie do yEd Live i włączeniu trybu VR

Niestety możliwości interakcji z grafem są bardzo ograniczone. Możliwe jest jedynie przemieszczanie, obracanie i skalowanie grafu. Ponadto tryb wirtualnej rzeczywistości w aplikacji został zbudowany w oparciu o API WebVR, które nigdy nie było domyślnie wspierane przez większość przeglądarek. W 2019 roku zostało zastąpione przez nowsze API WebXR.

Wszystkie interakcje, które są możliwe w trybie wirtualnej rzeczywistości, można było zobaczyć na stronie firmy yWorks już w 2017 roku. Na podstawie tych informacji, można założyć, że firma aktualnie nie skupia się na rozwoju funkcjonalności umożliwiającej wyświetlanie grafu w 3D i VR.

2.2. GraphXR

GraphXR [5] jest rozwiązaniem opracowanym przez firmę Kineviz. Pozwala ono między innymi na przeglądanie grafów w trójwymiarze, analizę danych, geoprzestrzenne wizualizacje, filtrowanie zawartości po osi czasu i wyświetlanie na grafach multimedialnych treści np. zdjęcia jako portrety osób. Aplikacja wspiera bezpośrednie połączenie z bazami Neo4j i wykonywanie zapytań w języku Cypher.

W aplikacji jest możliwe między innymi:

- Przeglądanie grafu w 3D
- Bezpośrednie połączenie się z bazą danych Neo4j (możliwa jest też integracja z innymi bazami w wersji Enterprise)
- Modyfikacja danych znajdujących się w bazie danych
- Wyświetlanie multimediów na grafie w postaci obrazów i wideo
- Modyfikacja i dodawanie wierzchołków i relacji do grafu
- Import i eksport danych w formacie CSV, GXRF (wersja Enterprise obsługuje również JSON, KMZ, GEXF i posiada integracje z Hunchly oraz Maltego)
- Zapis stanu grafu i udostępnienia go
- Kolaboracja w czasie rzeczywistym
- Dostosowanie kolorów wierzchołków i krawędzi według typu
- Wybór właściwości, która ma być użyta do wyświetlenia grafiki, podpisu i sterowania wielkością wierzchołka na grafie
- Ukrywanie wierzchołków i relacji poszczególnych typów
- Wizualizacje geoprzestrzenne
- Możliwość eksploracji grafu w wirtualnej rzeczywistości

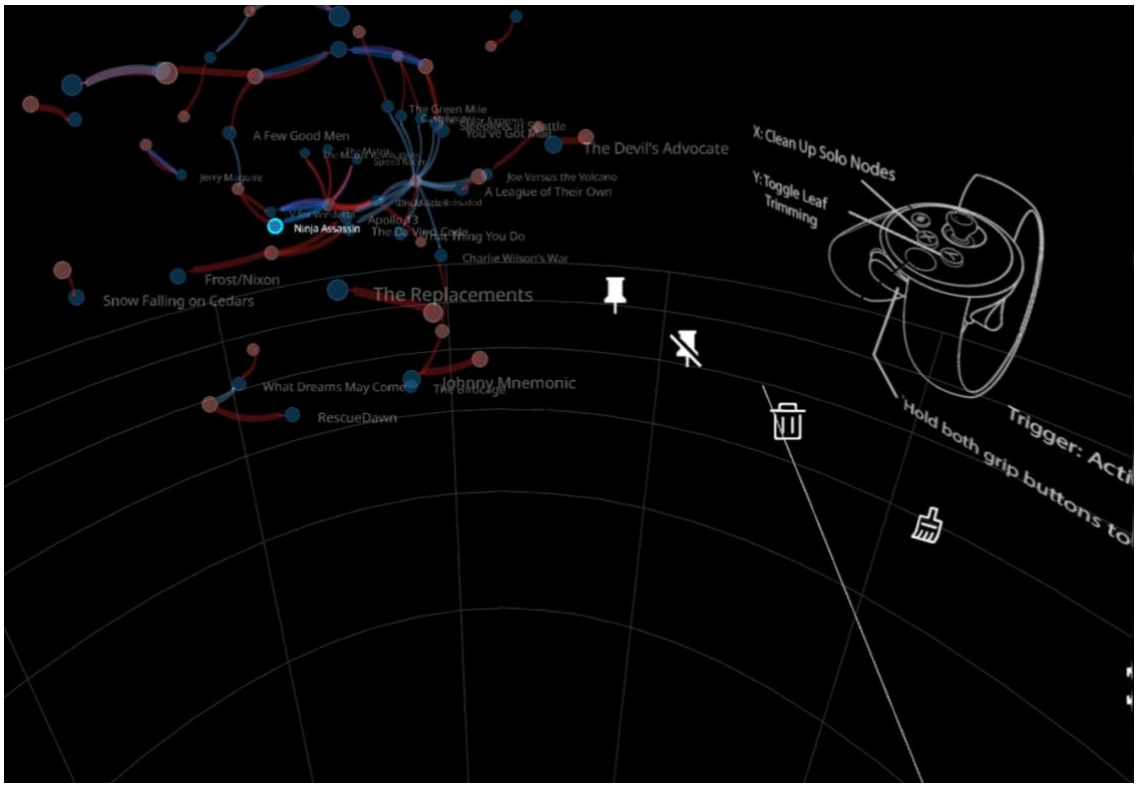
Aplikacja GraphXR pozwala również na przeglądanie danych w wirtualnej rzeczywistości. Wspierane są jedynie urządzenia firmy Oculus:

- Oculus Rift
- Oculus Rift S
- Oculus Quest 2

Tryb wirtualnej rzeczywistości (Rysunek 3) umożliwia m.in.:

- Skalowanie, przemieszczanie i zmianę orientacji grafu
- Zmianę pozycji wierzchołków
- Po zaznaczeniu danego wierzchołka możliwe jest szybkie zaznaczenie sąsiadujących wierzchołków

- Funkcjonalność przypinania wierzchołków w przestrzeni, która uniemożliwia im zmianę pozycji
- Funkcjonalność odcinania liści grafu



Rysunek 3. Zrzut ekranu z trybu wirtualnej rzeczywistości w aplikacji GraphXR

Niestety podczas testowania aplikacji z użyciem Oculus Quest 2 przyciski, które powinny umożliwiać bardziej zaawansowane interakcje nie działały, a po nacelowaniu przyciski nie były podświetlane. Zamiast nich podświetlane były niektóre z wierzchołków na grafie (dla każdego przycisku inny). Tryb wirtualnej rzeczywistości, podobnie jak w rozwiązaniu firmy yWorks, również został zbudowany w oparciu o przestarzałą technologię WebVR. Dodatkowo biorąc pod uwagę występujące błędy, aplikacja GraphXR także wygląda jakby nie była już rozwijana.

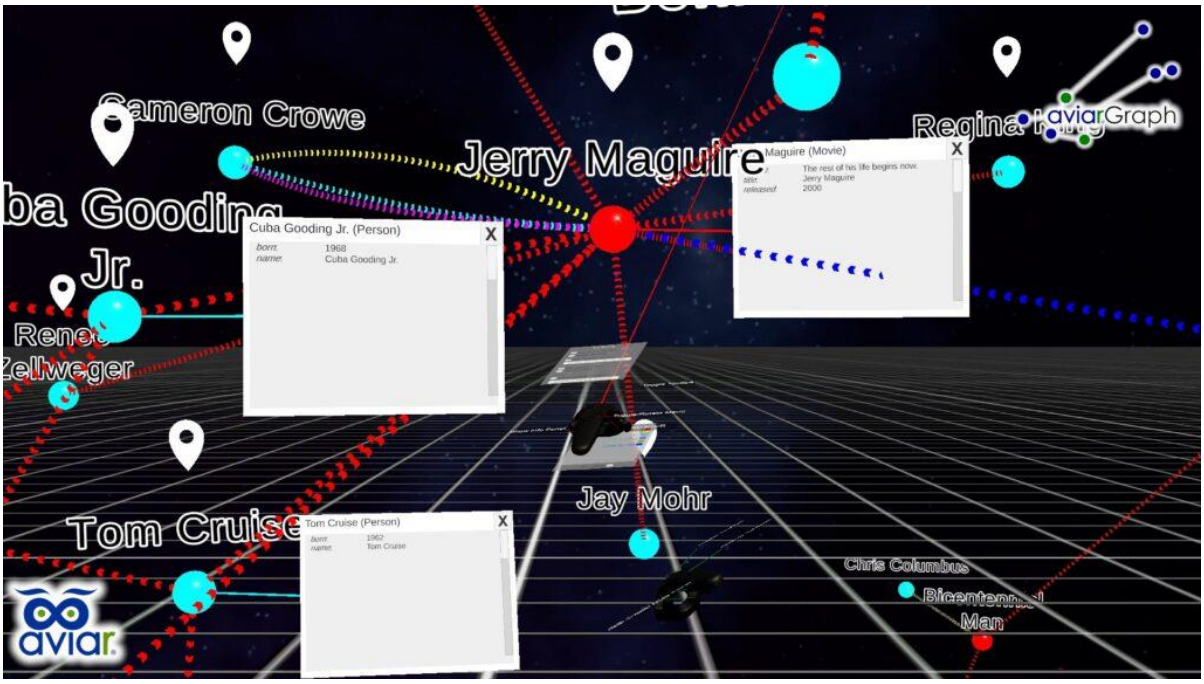
2.3. AviarGraph

AviarGraph [6] jest rozwiązaniem opracowanym przez Aviar Technology. Aplikacja jest zbudowana w oparciu o silnik Unity i pozwala na wizualizację grafu w wirtualnej rzeczywistości (Rysunek 4). Dane do utworzenia grafu są pobierane z bazy danych Neo4j dzięki bezpośredniemu połączeniu.

Cechy aplikacji:

- Poruszanie się po scenie za pomocą teleportacji
- Wierzchołki i krawędzie o tych samych typach są oznaczone tym samym kolorem.
- Możliwe jest przenoszenie wierzchołków i przypięcie ich w konkretnej pozycji w przestrzeni, aby uniemożliwić im dalszą zmianę pozycji.
- Ponadto możliwe jest filtrowanie, które relacje mają być wyświetlane
- Po kliknięciu na wierzchołek lub krawędź wyświetlane jest okno z właściwościami obiektu
- Możliwość ukrycia wierzchołków bez relacji

W odróżnieniu od poprzednich rozwiązań aplikacja AviarGraph umożliwia poruszanie się po scenie w trybie teleportacji. Ponadto pozwala na odfiltrowanie wierzchołków bez relacji i filtrowanie typów relacji wyświetlanych na grafie. Dzięki temu odnajdowanie i analizowanie konkretnych danych staje się łatwiejsze. Bardzo przydatną funkcjonalnością jest wyświetlanie właściwości obiektu po kliknięciu na nim.



Rysunek 4. Zrzut ekranu z aplikacji AviarGraph

2.4. Noda

Noda [7] to oprogramowanie dostępne na platformach Oculus, Steam i Viveport opracowane przez firmę Coding Leap. Rozwiązanie służy do tzw. “mind mappingu”. To czym wyróżnia się ta aplikacja na tle innych to prosta obsługa i dający duże możliwości poziom interakcji z elementami grafu.

Aby utworzyć wierzchołek wystarczy przez chwilę przytrzymać przycisk trigger. Aby utworzyć krawędź łączącą wierzchołki wystarczy przytrzymać trigger na jednym wierzchołku i upuścić go na drugim. Przyciskiem grip można przemieszczać elementy. Przytrzymując przyciski grip na obu kontrolerach możemy powiększać i pomniejszać wierzchołki wykonując gest na wzór uszczypnięcia znanego z gładzika laptopa czy ekranu smartfonu.

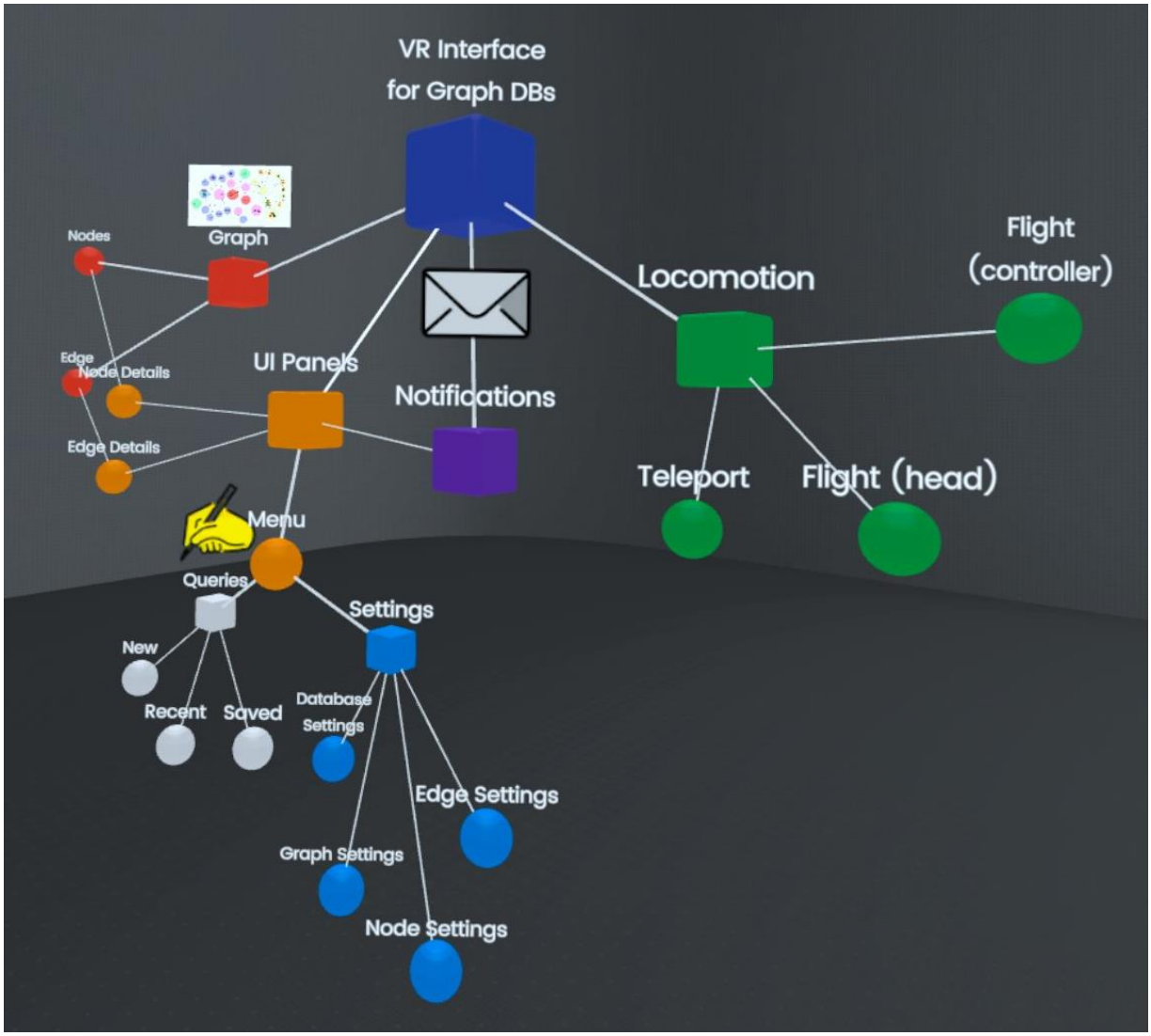
Klikając na element możemy zmodyfikować jego właściwości. Wierzchołkom możemy zmieniać tekst wyświetlany nad kształtem, kolory, przezroczystość, model 3D oraz dodawać obrazki wyświetlane nad kształtem i pisać notatki. Krawędziom możemy wybrać kolor (biały lub czarny), rodzaj (ciągła, przerywana, dwukolorowa) i grubość linii.

Ponadto aplikacja pozwala na:

- zaznaczanie obiektów w celu grupowego przemieszczania lub edycji
- przeglądanie Internetu za pomocą prostej przeglądarki wbudowanej w aplikację
- dodawanie obrazków z sieci przeciągając je z okna przeglądarki
- kolaborację w czasie rzeczywistym
- przemieszczanie się wokół grafu
- wprowadzanie tekstu używając konwersji mowy na tekst w wielu językach
- import i eksport z użyciem plików CSV

Na Rysunku 5 zaprezentowano przykładowy graf utworzony w aplikacji Noda. Wykorzystane zostały różne kształty i kolory wierzchołków. Przy węzłach „Notifications” i „Queries” zastosowano obrazki dostępne do wyboru w aplikacji. Węzeł „Graph” wyświetla obraz znaleziony w sieci za pomocą wbudowanej przeglądarki.

Noda jest prosta w obsłudze i sprawia wrażenie dopracowanego produktu. Aplikację cechują największe możliwości pod względem interakcji z grafem z wymienionych w Istniejące rozwiązania rozwiązań. Jako jedyna stara się ułatwić wprowadzanie danych. W tym celu zaimplementowano konwersję mowy na tekst. Ponadto wbudowana przeglądarka pozwala na szybkie wyszukanie informacji oraz kopiowanie tekstu i obrazów z Internetu.



Rysunek 5. Zrzut ekranu przedstawiający prosty graf utworzony w aplikacji Noda

3. Propozycja rozwiązania

W tym rozdziale zaprezentowano wynik analizy istniejących rozwiązań. Stanowi on listę funkcjonalności, które powinny posiadać interfejsy grafowych baz danych wykorzystujące wirtualną rzeczywistość. Przedstawiono również wymagania stawiane przed nowym rozwiązaniem.

3.1. Wnioski po analizie istniejących rozwiązań

Po analizie możliwości i mocnych stron dostępnych na rynku rozwiązań można wskazać pożądane funkcje dla tego typu aplikacji:

- Zdolność do nawiązania połączenia z bazą danych i wykonywania zapytań z poziomu aplikacji. Po otrzymaniu wyniku zapytania, powinien zostać wygenerowany graf zawierający informacje zwrócone przez bazę danych.
- Możliwość poruszania się w przestrzeni
- Możliwość przemieszczania i zmiany wielkości grafu
- Dostosowywanie sposobu w jaki są prezentowane elementy składowe grafu, np. możliwość zmiany podpisu pod elementem, jego kształtu, wielkości, koloru, możliwość wyświetlenia go w formie zdjęcia
- Filtrowania zawartości grafu
- Importowanie i eksportowanie grafów
- Możliwość interakcji użytkownika z grafem
- Ułatwienia przy wprowadzaniu danych w aplikacji, np. konwersja mowy na tekst
- Wieloplatformowość i kompatybilność z jak największą liczbą urządzeń
- Możliwość kolaboracji w czasie rzeczywistym

3.2. Wymagania

W tym podrozdziale przedstawiono wymagania stawiane przed prototypem nowego rozwiązania. Lista wymagań powstała w oparciu o cechy pożądane w tego typu oprogramowaniu wymienione w podrozdziale 3.1.

3.2.1. Wieloplatformowość

Aplikację powinno móc się uruchomić na jak największej liczbie platform. Aby to umożliwić rozwiązanie zostanie zaimplementowane z wykorzystaniem technologii webowych i będzie dostępne z poziomu przeglądarki internetowej.

3.2.2. Obsługa aplikacji bez urządzenia VR

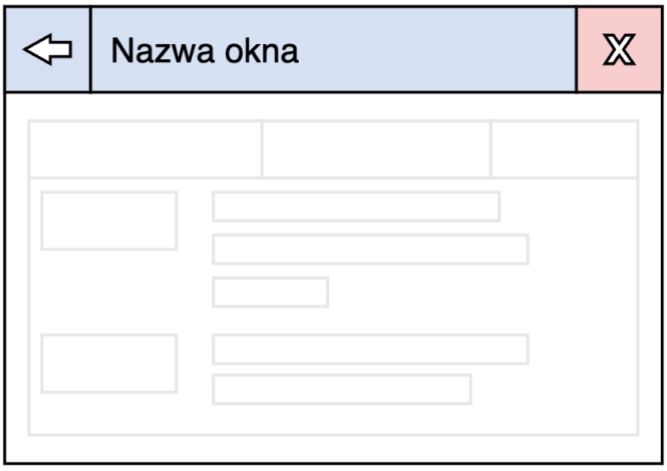
Sugerowanym sposobem podczas używania rozwiązania jest wykorzystanie urządzenia do wirtualnej rzeczywistości. Czasami jednak użytkownik chce coś szybko sprawdzić lub zmienić. Aby to ułatwić zostanie zaimplementowana możliwość obsługi aplikacji z wykorzystaniem klawiatury i myszy. Dzięki temu można sprawdzać wprowadzone do aplikacji zmiany bez wielokrotnego zakładania i zdejmowania urządzenia VR.

3.2.3. Interfejs użytkownika

Ze względu na rozbudowane możliwości jakie będzie oferować aplikacja konieczne jest zaimplementowanie interfejsu użytkownika. Zdecydowano, że większość elementów interfejsu użytkownika będą reprezentowane przez okna.

Wybór okien wydaje się odpowiedni ze względu na intuicyjność tego rozwiązania. Użytkownicy są przyzwyczajeni do okien, które są obecne na większości nowoczesnych systemów operacyjnych z graficznym interfejsem użytkownika. Ponadto taki interfejs będzie wygodny w obsłudze podczas korzystania z aplikacji bez urządzenia VR, z pomocą myszy.

Rysunek 6 przedstawia szkic planowanego układu okna. W górnej części znajduje się pasek z przyciskami i nazwą okna. Użytkownik będzie mógł je zamknąć (przycisk „X”) oraz cofnąć się do poprzednio otwartego okna (przycisk ze strzałką w lewo). W środku będzie znajdować się zawartość taka jak tekst, przyciski, pola tekstowe lub obrazy.



Rysunek 6. Szkic planowanego układu okna

3.2.4. Poruszanie się

Aplikacja powinna umożliwiać użytkownikowi poruszanie się po scenie. Dzięki temu będzie możliwe analizowanie danych z różnych pozycji i szybsze lokalizowanie interesujących danych.

Najprostszym do zaimplementowania sposobem poruszania wydaje się być lot. Ten tryb ruchu może zostać łatwo wdrożony do aplikacji. Kod wprowadzający ruch w wirtualnej rzeczywistości powinien mieć wiele części wspólnych z implementacją ruchu podczas obsługi bez urządzenia VR.

W wirtualnej rzeczywistości ten sposób poruszania się może nie odpowiadać każdemu. Jest to na tyle intensywne doświadczenie, że u niektórych osób może wywoływać nudności i ból głowy. Dlatego zdecydowano się udostępnić dodatkowo możliwość poruszania się za pomocą teleportacji.

3.2.5. Generyczna implementacja obsługi zapytań do bazy danych

W aplikacji będzie można wykonywać zapytania do bazy danych, a ich wynik będzie prezentowany na grafie. Użytkownik będzie mógł korzystać z wielu baz i przełączać się między nimi z poziomu klienta.

Zapytania będą budowane w języku zapytań używanym w danej technologii, na przykład dla Neo4j będzie to Cypher. Po ich zbudowaniu, będzie można je zapisać w celu późniejszego wykorzystania. Pozwoli to na szybsze wysyłanie najczęściej wykonywanych zapytań.

Aplikacja umożliwi korzystanie z różnych baz danych. Domyślnie wspierane będą Neo4j i ArangoDB. Będzie możliwe rozszerzanie listy wspieranych baz. W tym celu konieczne będzie zaimplementowanie klasy obsługującej połączenie z konkretną bazą danych.

3.2.6. Graf

Wyniki zapytań zwrócone do aplikacji klienckiej będą prezentowane w formie grafu. Użytkownik będzie miał możliwość dostosowania sposobu wyświetlania tych danych. Dla grafu będzie możliwa zmiana skali, pozycji, długości krawędzi łączących wierzchołki oraz czy ma być generowany w 2D (płaszczyzna równoległa do podłoża) czy w przestrzeni 3D.

Wierzchołkom posiadającym dane etykiety będzie można zmienić rozmiar, kolor, przezroczystość, kształt oraz podpis pod obiektem 3D. Będzie możliwość ustawienia aby zamiast obiektu 3D o danym kształcie był renderowany obraz. W tym celu wierzchołek powinien mieć adres URL prowadzący do obrazu jako wartość jednej z właściwości.

Po najechaniu na wierzchołek pojawi się okno ze szczegółowymi informacjami o nim. Zostanie wyświetlony identyfikator, etykiety wierzchołka oraz wartości wszystkich właściwości, które są do niego przypisane.

W trybie VR będzie możliwe również przenoszenie wierzchołków. Po wskazaniu na jeden z nich i przytrzymaniu przycisku, zacznie on podążać za kontrolerem.

Wygląd krawędzi reprezentujących relacje będzie mógł być zmieniany podobnie jak w przypadku wierzchołków. Krawędziom danego typu będzie można zmienić tekst podpisujący krawędzie, kolor, grubość linii oraz przezroczystość.

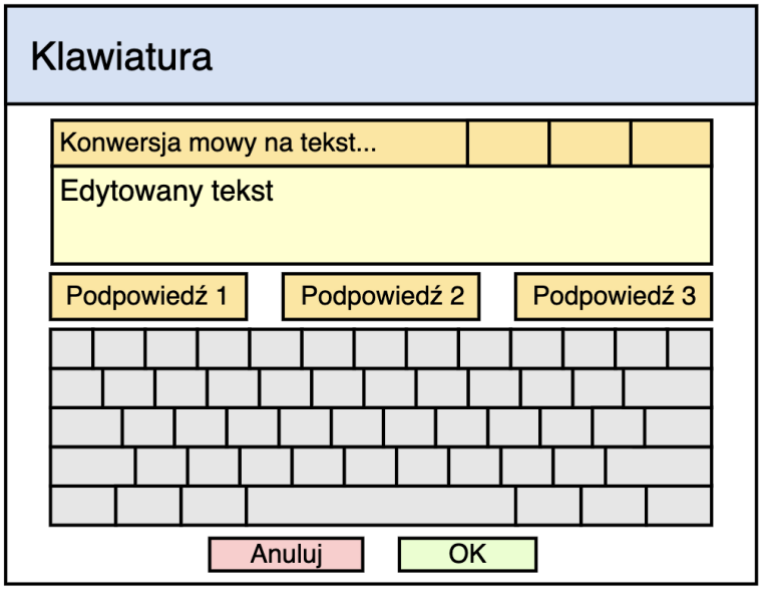
Po najechaniu na stożek wskazujący kierunek relacji, pojawi się okno ze szczegółami takimi jak identyfikator, typ oraz wartości właściwości przypisanych do relacji.

3.2.7. Wprowadzanie danych

Aplikacja powinna zapewniać wygodny sposób na wprowadzanie danych. Podczas obsługi z wykorzystaniem klawiatury i myszki nie stanowi to problemu. W takim przypadku będzie używana klawiatura podłączona do komputera.

Problem pojawia się podczas korzystania z aplikacji w wirtualnej rzeczywistości. Użytkownik z HMD na głowie nie widzi klawiatury. Wymusza to wdrożenie wirtualnej klawiatury. Układ klawiszy zostanie dostosowany tak, aby umożliwić szybkie budowanie zapytań. Dodatkowo zaimplementowane zostaną mechanizmy predykcji wprowadzanych słów oraz wprowadzania głosowego, dzięki wykorzystaniu wbudowanych w przeglądarki internetowe API.

Rysunek 7 przedstawia planowany układ klawiatury. W górnej części będzie znajdować się panel wykorzystywany przy konwersji mowy na tekst. Niżej będzie obszar tekstowy zawierający edytowany tekst. Pod nim wyświetlane będą podpowiedzi zaproponowane przez mechanizm predykcji. W dolnej części okna znajdować się będzie klawiatura oraz przyciski do anulowania edycji i akceptowania zmian.



Rysunek 7. Szkic układu okna z klawiaturą

4. Zastosowane technologie

W rozdziale przedstawiono technologie, które zostały wykorzystane do stworzenia prototypu rozwiązania.

4.1. HTML

HTML (HyperText Markup Language) jest językiem znaczników, który jest obecnie standardem przy pisaniu stron internetowych [8]. Został opracowany w latach 90. przez Tima Berners Lee w CERN. Początkowo służył jedynie do semantycznego opisu dokumentów dotyczących badań naukowych.

Został stworzony w oparciu o SGML (Standard Generalized Markup Language). Główna różnica w stosunku do SGML polega na istnieniu zbioru określonych znaczników oraz reguł ich stosowania.

Listing 1. Struktura prostego dokumentu HTML

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Sample page</title>
 </head>
 <body>
  <h1>Sample page</h1>
  <p>This is a <a href="demo.html">simple</a> sample.</p>
  <!-- this is a comment -->
 </body>
</html>
```

Listing 1 prezentuje przykładowy dokument HTML. Dokument zawiera drzewa składające się z elementów i tekstu. Większość elementów zaczyna się od znacznika początkowego (np. `<p>`) i kończy znacznikiem zamykającym (np. `</p>`). W niektórych z elementów można pominąć znacznik startujący lub zamykający.

Element można dodatkowo opisać atrybutami, które zawierają dodatkowe informacje niezbędne do poprawnego działania elementu lub ułatwiające pracę programistom. Przykładowo:

- Atrybut `href` w elemencie a służy do wskazania adresu, na który ma przekierować hiperłącze

- Atrybut `class` pozwala pogrupować elementy, które mają podobne działanie. Przy definiowaniu wyglądu elementów za pomocą styli CSS programiści używają zazwyczaj tego atrybutu.

Wartość atrybutu `id` stanowi unikalny identyfikator elementu. Może być przez programistę użyty do łatwego zlokalizowania elementu na stronie. Może też być użyty punkt nawigacji na stronie. Jeśli klikniemy na hiperłącze, które w atrybucie `href` ma wpisane id elementu poprzedzone znakiem '#', zostaniemy przeniesieni w miejsce na stronie, w którym występuje ten element.

Każdy atrybut jest definiowany w znaczniku początkowym i musi zawierać nazwę oraz wartość atrybutu oddzieloną '='.

Przeglądarki internetowe analizują dokument HTML i na jego podstawie tworzą drzewo DOM (Document Object Model), które przechowywane jest w pamięci. Takie drzewo może być modyfikowane przez skrypty, co umożliwia dynamiczną zmianę treści na stronie.

4.2. CSS

CSS, czyli kaskadowe arkusze stylów (ang. Cascading Style Sheets), jest językiem służącym do określania jak dokumenty mają być prezentowane użytkownikowi. Dokumenty stanowią zazwyczaj pliki tekstowe zapisane w języku znaczników, np. dokumenty HTML. Język jest jedną ze standardowych technologii używanych przy tworzeniu stron internetowych. Jest rozwijany przez grupę CSS Working Group utworzoną w 1997 roku przez World Wide Web Consortium (W3C) [9].

CSS jest językiem opartym na regułach zawierających style, które mają być zaaplikowane do ustalonych elementów na stronie. Przykład reguły CSS został przedstawiony na Listingu 2. Reguła zaczyna się od selektora. Następnie w klamrach deklarowane są style w postaci par składających się z właściwości i wartości, którą ma przyjąć właściwość elementu.

Listing 2. Reguła definiująca styl dla nagłówków reprezentowanych znacznikiem h1

```
h1 {
   color: red;
   font-size: 3em;
}
```

4.3. JavaScript

JavaScript**,** znany także jako JS, jest wieloplatformowym, wysokopoziomowym skryptowym językiem programowania zgodnym ze standardem ECMAScript. Został opracowany przez firmę Netscape w 1995 roku na potrzeby przeglądarki internetowej Navigator. Może być intepretowany lub kompilowany metodą JIT (ang. just-in-time compilation).

Jest wieloparadygmatowym językiem pozwalającym na programowanie imperatywne, funkcyjne i sterowane zdarzeniami. Jest zorientowany obiektowo i oparty na prototypach. Charakteryzuje się dynamicznym typowaniem i funkcjami pierwszej klasy.

JavaScript jest jedną z wiodących technologii wykorzystywanych przy tworzeniu stron internetowych. 98% wszystkich stron internetowych używa JS do wykonywania kodu po stronie przeglądarki internetowej [10]. JavaScript znajduje także zastosowanie w innych obszarach i może być wykorzystany do wykonywania kodu po stronie serwera lub tworzenia aplikacji desktopowych.

4.3.1. Webpack

Nowoczesne aplikacje webowe składają się z wielu plików, między którymi występują zależności. Webpack [11] jest narzędziem, które zajmuje się wczytywaniem plików w odpowiedniej kolejności. Na podstawie pliku konfiguracyjnego analizuje pliki występujące w aplikacji i buduje dla nich graf zależności. Na jego podstawie powstaje kod, który jest zapisywany do jednego lub więcej plików (tzw. bundle). Bundle zawiera tylko pliki, które rzeczywiście są używane w aplikacji, przez co zmniejsza ilość danych do pobrania. Ponadto konieczność pobrania tylko jednego pliku sprawia, że ładowanie aplikacji w przeglądarce przebiega szybciej.

Domyślnie Webpack rozumie jedynie pliki JavaScript i JSON. Loadery to narzędzia, które umożliwiają obsługę innych typów plików i utworzenie z nich modułów, które zostaną dodane do grafu zależności.

4.3.2. Babel

Babel [12] jest transpilatorem, który pozwala na konwersję kodu źródłowego JavaScript na wersję, która jest wspierana przez przeglądarki internetowe. Umożliwia to korzystanie z najnowszych funkcjonalności języka JavaScript niezależnie od przeglądarki, na której jest uruchamiana aplikacja. Może być użyty z Webpackiem przez zastosowanie loadera babel-loader.

4.3.3. Biblioteka three-forcegraph

Three-forcegraph [13] jest biblioteką, która umożliwia reprezentację danych grafu za pomocą instancji klasy `Object3D` z biblioteki Three.js. Obiekty tworzą graf w przestrzeni trójwymiarowej dzięki wykorzystaniu algorytmów opartych o symulację sił z jakimi oddziałują na siebie obiekty.

4.3.4. Biblioteka Notyf

Notyf [14] to prosta biblioteka, która pozwala na szybkie i łatwe tworzenie powiadomień prezentowanych użytkownikowi na stronie. Dla każdego powiadomienia oprócz treści można określić między innymi:

- pozycję w pionie i poziomie,
- czas trwania,
- kolor,
- ikonę,
- czy ma być możliwe do zamknięcia przez użytkownika przed upływem czasu trwania

4.4. Web Speech API

Web Speech API pozwala programistom na udostępnienie w swoich aplikacjach funkcjonalności wprowadzania głosowego i zamianę tekstu na mowę. API jest niezależne od implementacji i może obsługiwać rozpoznawanie i syntezę mowy na serwerze jak i w kliencie [15]. Składa się z dwóch części. Interfejs SpeechSynthesis umożliwia programiście odczytanie tekstu z pomocą syntezatora mowy. Interfejs SpeechRecognition pozwala na rozpoznanie mowy i konwersję na tekst. Zanim korzystanie z interfejsu będzie możliwe, użytkownik musi potwierdzić zgodę na użycie mikrofonu przez stronę.

Na chwilę obecną interfejs SpeechRecognition nie jest wspierany przez wszystkie przeglądarki. Wsparcia nie mają między innymi Firefox i Opera [16].

4.5. WebGL

WebGL jest wieloplatformowym, wolnym od opłat licencyjnych API używanym do tworzenia grafiki 3D w przeglądarce internetowej. Wykorzystuje element Canvas występujący w HTML5. Jest oparte na znanym i szeroko przyjętym standardzie OpenGL ES [17].

Do jego zalet można zaliczyć:

- kompatybilność z wieloma przeglądarkami i platformami
- ścisła integracja z zawartością HTML
- grafika 3D z akceleracją sprzętową w przeglądarce internetowej
- środowisko skryptowe, które ułatwia prototypowanie grafiki 3D – nie wymaga korzystania z kompilatora i linkera

4.6. Three.js

Three.js [18] to darmowa biblioteką open-source, która zapewnia abstrakcję dla API WebGL. Dzięki temu ułatwia proces programowania, sprawia, że jest mniej podatny na błędy oraz skraca ilość kodu wymaganą do uzyskania pożądanego efektu. Główne zalety biblioteki:

- Łatwe do opanowania podstawy
- Szczegółowa dokumentacja z dużą ilością przykładów
- Dobra wydajność
- Duża i aktywna społeczność skupiona wokół projektu
- Kompatybilność z większością formatów modeli 3D

4.7. WebXR

WebXR [19] pozwala na wprowadzenie do aplikacji webowych rzeczywistości rozszerzonej i wirtualnej (AR i VR). Połączenie tych technologii jest opisywane jako MR (z ang. Mixed Reality). XR (z ang. Extended Reality) jest bardziej ogólnym terminem, który określa wszystkie wyżej wymienione technologie (VR, AR, MR).

„Do głównych zadań WebXR należy:

- wykrycie czy urządzenie pozwala na uruchomienie doświadczenia XR
- zapytanie o możliwości urządzenia XR
- odpytywanie o stan urządzenia XR i powiązanych urządzeń wejścia
- wyświetlenie obrazu na urządzeniu XR w odpowiedniej liczbie klatek” [20]

4.8. Node.js

Node.js jest wieloplatformowym środowiskiem uruchomieniowym o otwartym kodzie. Został stworzony przez Ryana Dahla w 2009 roku. Pozwala na pisanie w języku JavaScript aplikacji działających po stronie serwera. Dzięki temu programiści mogą pracować nad aplikacjami webowymi wykorzystując ten sam język do implementacji logiki po stronie klienta i serwera.

Node.js działa w oparciu o silnik JavaScript V8, który jest używany w przeglądarce Google Chrome. Zamiast wielu wątków oczekujących na przetworzenie zapytań, Node.js przetwarza je w tym samym wątku, z wykorzystaniem modelu obsługi zdarzeń. To sprawia, że serwery działające na środowisku Node.js są dobrze skalowalne [21].

Instalacja, skonfigurowanie i programowanie w Node.js są bardzo proste. Listing 3 przedstawia jak mało kodu wymaga uruchomienie prostego serwera.

Listing 3. Prosty serwer utworzony z użyciem Node.js

```
const http = require('http');
const PORT = 3000;
const server = http.createServer((request, response) => {
   response.writeHead(200, { 'Content-Type': 'text/html' });
   response.write('<h1>Hello World!</h1>');
   response.end();
});
server.listen(PORT, () => console.log('Listening on port ' + PORT));
```

4.9. Express.js

Express.js [22] to jeden z najpopularniejszych frameworków Node.js. Rozszerza funkcjonalności Node.js. Dzięki temu ułatwia i przyspiesza pracę nad projektem. Express.js jest często wybierany przez programistów w celu zaoszczędzenia czasu potrzebnego na napisanie kodu, testowanie, szukanie oraz naprawę błędów występujących aplikacji.

Usprawnia między innymi:

- obsługę zapytań HTTP
- zarządzanie routingiem
- integrację z silnikami do generowania widoków po stronie serwera

4.10. MongoDB

MongoDB [23] – system do zarządzania bazą danych NoSQL o otwartym kodzie, opublikowany w 2009 roku. MongoDB zamiast przechowywać dane w postaci tabel i rekordów, wykorzystuje dokumenty i kolekcje. Zapytania są budowane z użyciem języka MQL (MongoDB Query Language).

Dokumenty są podstawową jednostką w MongoDB, stanowią zbiór kluczy i odpowiadających im wartości. Są tworzone i przechowywane w formacie BSON (od ang. Binary JSON).

Kolekcje to zbiory dokumentów i są odpowiednikiem tabeli z relacyjnych baz danych. Kolekcje charakteryzują się dynamicznymi schematami. Oznacza to, że dokumenty w tej samej kolekcji mogą posiadać inne klucze lub przechowywać inne typy wartości pod tymi samymi kluczami. Mimo to MongoDB pozwala na ustalenie walidacji schematów dla kolekcji, które są uruchamiane przy każdej operacji tworzenia lub aktualizacji dokumentu.

4.11. Mongoose

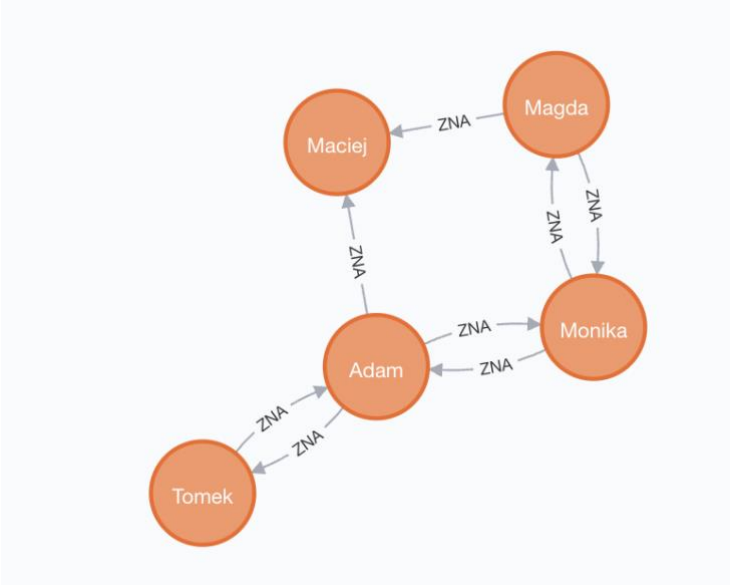
Mongoose [24] to biblioteka Node.js, która umożliwia modelowanie i walidację schematów danych dla bazy MongoDB. Ułatwia definiowanie logiki dotyczącej sprawdzania poprawności danych po stronie aplikacji. Dzięki temu nie ma konieczności definiowania walidacji dla kolekcji na poziomie bazy danych.

4.12. Neo4j

Neo4j [25] to system do zarządzania grafowymi bazami danych opracowany w 2007 roku przez Neo4j, Inc. Dane w Neo4j są przechowywane w formie grafów, czyli zbiorów wierzchołków (inaczej węzłów) i krawędzi je łączących.

Węzły w Neo4j są wykorzystywane do reprezentacji encji. Każdy węzeł posiada jedną lub więcej etykiet, które pozwalają pogrupować węzły. Przykładowe etykiety: Osoba, Film, Reżyser. Poza etykietą węzły przechowują dane w postaci właściwości w postaci par: klucz i odpowiadająca mu wartość

Węzły są połączone krawędziami, które reprezentują relacje pomiędzy węzłami. Relacja ma określony typ, który często jest wyrażony w formie czasownika i opisuje pewną zależność zachodzącą między węzłami, np. LUBI, ZNA, PRZECZYTAŁ. Podobnie jak węzły, relacje mogą przechowywać dane w postaci właściwości opisujących relację.



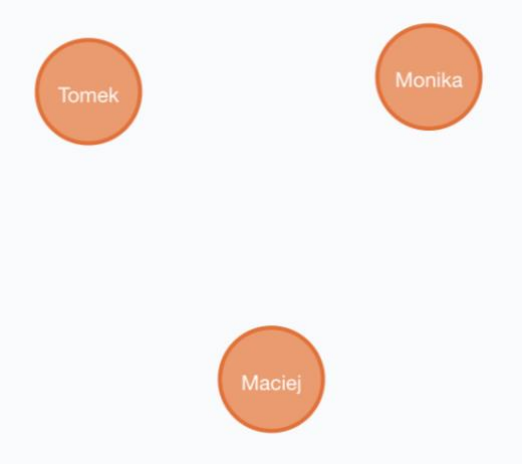
Rysunek 8. Przykładowy graf w Neo4j

Cypher to inspirowany SQL język do budowania zapytań w Neo4j. Aby zapewnić czytelność zapytań, Cypher w swojej składni używa ASCII-art. Dobrze to widać na Listingu 4. Węzły w zapytaniu są zapisane z wykorzystaniem nawiasów `(:Osoba {imie: "Adam"})` oraz `(osoba)`, a relacja i jej kierunek są reprezentowane za pomocą strzałki `-[:ZNA]->`.

Listing 4. Przykładowe zapytanie. Zwróci wszystkie węzły, które są połączone relacją „ZNA” z węzłami zawierającymi etykietę „Osoba” i właściwość „imie” o wartości „Adam”

```
MATCH (:Osoba {imie: "Adam"})-[:ZNA]->(osoba)
RETURN osoba
```

Zapytanie z Listingu 4 wysłane do bazy zawierającej dane z Rysunku 8 zwróci 3 węzły reprezentujące następujące osoby: Tomek, Monika i Maciej. Wynik zapytania został przedstawiony na Rysunku 9. Na rysunku nie widać relacji między węzłami, ponieważ zapytanie zostało skonstruowane tak, aby zwróciło jedynie węzły znajdujące się pod zmienną `osoba`.



Rysunek 9. Wynik zapytania z Listingu 4 dla grafu z Rysunku 8

4.13. ArangoDB

ArangoDB [26] jest systemem do zarządzania wielomodelową nierelacyjną bazą danych opracowanym przez triAGENS GmbH. Dane mogą być przechowywane jako pary kluczy i wartości, dokumenty JSON lub grafy. Niezależnie od modelu danych, dane mogą być pozyskiwane za pomocą jednego deklaratywnego języka zapytań – ArangoDB Query Language (AQL)

Dla Neo4j zostało przedstawione przykładowe zapytanie (Listing 4), którego wynikiem były wszystkie węzły reprezentujące osoby, które zna Adam. Listing 5 zawiera zapytanie o te dane napisane w języku AQL.

Listing 5. Zapytanie AQL którego wynik dla grafu przedstawionego na Rysunku 8. odpowiada wynikowi zapytania Cypher z Listingu 4.

```
FOR osoba in osoby
    FILTER osoba.imie == "Adam"
    FOR v IN OUTBOUND
        osoba zna
        FILTER IS_SAME_COLLECTION('osoby', v)
        RETURN v
```

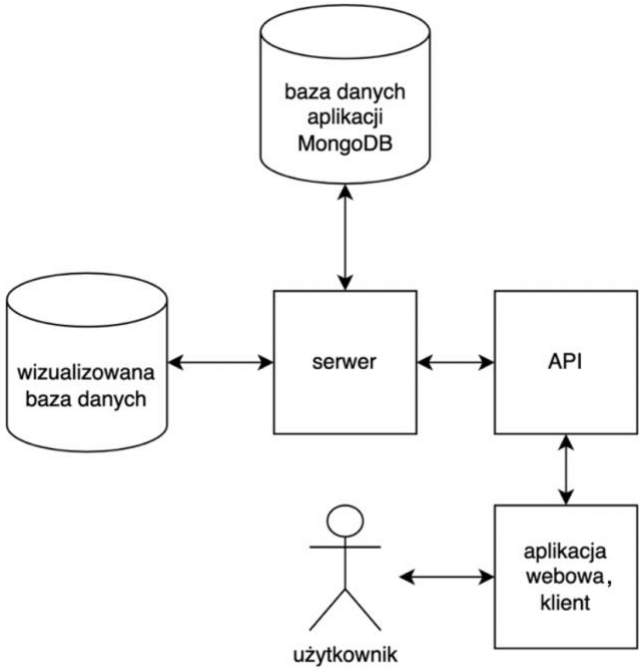
5. Implementacja

W niniejszym rozdziale opisano architekturę zastosowaną podczas implementacji oraz szczegóły dotyczące połączenia serwera z bazą danych aplikacji i z bazą do wizualizacji. Aplikacja kliencka została opisana w kolejnym rozdziale.

5.1. Architektura rozwiązania

W skład architektury rozwiązania wchodzą baza danych aplikacji, baza danych do wizualizacji, serwer, API oraz klient w postaci aplikacji webowej. Rysunek 10 przedstawia schemat architektury.

Serwer komunikuje się bazą danych aplikacji (MongoDB) i przechowuje w niej takie dane jak preferencje użytkownika oraz zapisane zapytania. Serwer nawiązuje również połączenie z bazą danych do wizualizacji. Aby zapewnić generyczność rozwiązania, istnieje możliwość połączenia z różnymi bazami danych. Niezbędne do tego jest zaimplementowanie klasy odpowiadającej za komunikację z daną bazą. Domyślnie zostały zaimplementowane klasy obsługujące bazy danych Neo4j i ArangoDB.



Rysunek 10. Architektura rozwiązania

5.2. Baza danych aplikacji

Aplikacja umożliwia użytkownikowi wykonywanie zapytań do bazy danych oraz dostosowanie ustawień prezentacji danych na grafie. Wprowadzanie ich za każdym razem byłoby uciążliwe, dlatego te informacje są przechowywane w bazie danych aplikacji.

Jako bazę danych dla aplikacji wybrano nierelacyjną bazę danych MongoDB. Jest to często wybierana baza danych przy serwerach działających na środowisku Node.js. Głównym powodem była łatwość obsługi tej bazy. Jednym ze składających się na to czynników jest wymiana danych w formacie JSON.

Zastosowano również Mongoose, bibliotekę Node.js, która ułatwia modelowanie i walidację schematów danych. Schematy są definiowane za pomocą klasy `Schema` i obiektu, który zawiera informację o nazwach pól i typie danych, które przechowują.

W projekcie utworzono dwa schematy:

1. Schemat `GraphSettings` definiujący strukturę danych dla rekordów, które dotyczą ustawień wyświetlania grafu wprowadzonych przez użytkownika. Listing 6 zawiera kod schematu, a pola wchodzące w skład schematu opisano w Tabeli 1.

Listing 6. Kod definiujący schemat `GraphSettings`

```
const mongoose = require('mongoose');
const graphSettingsSchema = new mongoose.Schema({
    databaseId: { type: String, required: true },
    type: { type: String, required: true },
    label: { type: String, required: false },
    data: { type: Map, required: true }
});
const GraphSettings = mongoose.model('GraphSettings', graphSettingsSchema);
module.exports = GraphSettings;
```

Tabela 1. Pola zdefiniowane w schemacie `GraphSettings`

Pole	Opis
`databaseId`	Identyfikator bazy, dla której zapisywane są ustawienia grafu zdefiniowane przez użytkownika
`type`	Typ obiektu, którego dotyczy ustawienie (`node` lub `edge`)
`label`	Jeśli pole `type` zawiera wartość `node`, pole `label` przechowuje etykietę węzłów. Ustawienie wpłynie na każdy węzeł zawierający daną etykietę. Jeśli pole `type` zawiera wartość `edge`, pole `label` przechowuje typ relacji.
`data`	Obiekt zawierający dane ustawień w postaci kluczy i ich wartości.

2. Schemat `SavedQuery` definiujący strukturę danych dla rekordów, które przechowują zapytania zapisane przez użytkownika. Kod schematu przedstawiono na Listingu 7, a w Tabeli 2 opisano zawarte w nim pola.

Listing 7. Kod definiujący schemat `SavedQuery`

```
const mongoose = require('mongoose');
const savedQuerySchema = new mongoose.Schema({
    databaseId: { type: String, required: true },
    queryName: { type: String, required: true },
    queryString: { type: String, required: true }
});
const SavedQuery = mongoose.model('SavedQuery', savedQuerySchema);
module.exports = SavedQuery;
```

Tabela 2. Pola zdefiniowane w schemacie `SavedQuery`

Pole	Opis
`databaseId`	Identyfikator bazy, dla której zapisywane jest zapytanie
`queryName`	Nazwa pod którą użytkownik chce zapisać zapytanie
`queryString`	Treść zapytania

5.3. Wizualizowana baza danych

Aby było możliwe wyświetlenie danych w postaci grafu w aplikacji klienckiej, serwer musi zwrócić dane w ustalonym formacie. Różne bazy danych zwracają dane w różnych formatach. To sprawia, że zanim odpowiedź z bazy danych zostanie przekazana na frontend, musi zostać przeprocesowana i przetłumaczona na format, który będzie zrozumiały dla aplikacji klienckiej.

5.3.1. Rozszerzanie listy obsługiwanych baz danych

W tym celu została zastosowana koncepcja implementowania klas do komunikacji z bazą danych. Takie podejście dodaje warstwę abstrakcji, która pozwala na współdzielenie kodu do obsługi różnych baz po stronie frontendu i backendu. Ponadto pozwala to na łatwe rozszerzanie listy obsługiwanych baz danych.

Domyślnie aplikacja potrafi obsłużyć połączenie z bazami danych Neo4j i ArangoDB. Zapewnienie wsparcia dla kolejnych baz danych może zostać zrealizowane przez dopisanie wtyczki w postaci klasy obsługującej połączenie z daną bazą. Po poprawnym zaimplementowaniu klasy należy zmodyfikować plik konfiguracyjny, dodając wszystkie dane wymagane do połączenia się z bazą oraz wskazując klasę, która ma to połączenie obsługiwać.

5.3.2. Konfiguracja połączeń z bazami danych

Plik GraphDatabaseHandlers.js przechowuje konfigurację połączeń z bazami danych. Konfiguracja z pominięciem niektórych właściwości jest wykorzystywana również w aplikacji klienckiej. Przykładowy kod dla pliku zaprezentowano na Listingu 8.

Konfigurację stanowi obiekt, w którym klucze są identyfikatorami połączeń do baz danych. Wartościami są obiekty z dwoma właściwościami:

- `constructor` – przechowuje referencję do klasy, która obsługuje połączenie z bazą danych danego typu.
- `config` – przechowuje obiekt, który jest przekazywany jako argument przy tworzeniu instancji klasy obsługującej połączenie. Jego zawartość jest także zwracana do aplikacji klienckiej. Wyjątkiem są zagnieżdżone obiekty zawierające wartość **true** dla właściwości **hide**. Dzięki temu przesyłane są jedynie dane, które są przydatne dla użytkownika i nie stwarzają zagrożenia dotyczącego bezpieczeństwa.

Listing 8. Przykładowy kod konfiguracji połączeń z bazami danych

```
const ArangoDb = require('./ArangoDb');
const Neo4j = require('./Neo4j');
module.exports = {
    neo: {
        constructor: Neo4j,
        config: {
            name: { label: 'Name', value: 'Neo4J' },
            address: {
                order: 1,
                label: 'Database Address',
                value: 'bolt://localhost:7687'
            },
            user: { order: 2, label: 'User', value: 'neo4j' },
            password: { value: '123456', hide: true }
        }
    },
    arango: {
        constructor: ArangoDb,
        config: {
            name: { label: 'Name', value: 'ArangoDB' },
            address: {
                order: 1,
                label: 'Database Address',
                value: 'bolt://localhost:8529'
            },
            database: { order: 2, label: 'Database', value: 'Trains' },
            user: { order: 3, label: 'User', value: 'root' },
            password: { value: '1234', hide: true }
        }
    }
}
```

5.3.3. Implementacja klasy obsługującej połączenie z bazą danych

Każda klasa obsługująca połączenie z bazą danych musi implementować zestaw metod, które są wywoływane z określonymi argumentami i zwracać odpowiedź w ustalonej strukturze (Listing 11). Metody wymagające implementacji oraz przyjmowane przed nie argumenty zostały opisane w Tabeli 3.

Tabela 3. Metody wymagające implementacji w klasie obsługującej połączenie z bazą danych

Metoda	Argumenty	Opis
`constructor`	`config` – konfiguracja dla połączenia z bazą danych, jest przechowywana w pliku GraphDatabaseHandlers.js	Metoda konstruktora, która jest odpowiedzialna za inicjację obiektu klasy. Powinna zawierać podstawową walidację konfiguracji, która jest przekazywana jako argument funkcji
`query`	`queryString` – tekst zapytania	Funkcja używana do wykonywania zapytań do bazy danych. Powinna zwracać wynik w o ustalonej strukturze (Listing 11).
`queryNodeById`	`nodeId` – identyfikator węzła	Metoda używana do wykonywania zapytań o konkretny węzeł. Powinna zwracać wynik o ustalonej strukturze (Listing 11).
`getKeywords`	brak	Metoda powinna zwracać tablicę ze słowami kluczowymi występującymi w języku, w którym budowane są zapytania do bazy danych. Wynik tej funkcji używany jest przy predykcji tekstu w aplikacji klienckiej.
`getNodesMetadata`	brak	Metoda zwraca metadane na temat węzłów znajdujących się w bazie danych. Wynikiem funkcji powinna być tablica obiektów o ustalonej strukturze. Na Listingu 9 został przedstawiony przykładowy wynik metody.
`getEdgesMetadata`	brak	Metoda zwraca metadane na temat krawędzi znajdujących się w bazie. Wynikiem funkcji powinna być tablica obiektów o ustalonej strukturze. Na Listingu 10 został przedstawiony przykładowy wynik metody.

Metoda `getNodesMetadata` powinna zwracać tablicę obiektów. Przykładowy obiekt przedstawiono na Listingu 9. Obiekt powinien posiadać właściwości:

- `labels` – tablica zawierająca zestaw etykiet węzłów
- `count` – ilość węzłów w bazie posiadających dany zestaw etykiet
- `properties` – tablica zawierająca właściwości występujące w węzłach o danym zestawie etykiet. Każdy element tablicy jest obiektem o dwóch właściwościach:
    - `name` – nazwa właściwości przypisanej do węzła
    - `count` – ilość węzłów posiadających daną właściwość

Listing 9. Przykładowy obiekt z tablicy zwracanej przez metodę `getNodesMetadata`

```
{
  "labels": ["Osoba", "Student"],
  "count": 21,
  "properties": [
    { "name": "imie", "count": 21 },
    { "name": "nazwisko", "count": 21 },
    { "name": "wiek", "count": 18 },
    { "name": "email", "count": 17 }
  ]
}
```

Metoda `getEdgesMetadata` powinna zwracać tablicę obiektów. Przykładowy obiekt przedstawiono na Listingu 10. Każdy obiekt powinien posiadać właściwości:

- `type` – typ relacji
- `count` – ilość krawędzi danego typu w bazie danych
- `properties` – tablica zawierająca właściwości występujące w krawędziach o danym typie. Każdy element tablicy jest obiektem o dwóch właściwościach:
    - `name` – nazwa właściwości przypisanej do relacji
    - `count` – ilość relacji posiadających daną właściwość

Listing 10. Przykładowy obiekt z tablicy zwracanej przez metodę `getEdgesMetadata`

```
{
  "type": "PRACUJE_W",
  "count": 21,
  "properties": [
    { "name": "od", "count": 21 },
    { "name": "stanowisko", "count": 21 },
    { "name": "do", "count": 7 }
  ]
}
```

5.3.4. Format danych dla wyników zapytań

Metody wykonujące zapytania do bazy danych powinny zwracać obiekt posiadający dwie właściwości: `graph` oraz `table`.

Pod pierwszą z nich znajdują się informacje, które są wykorzystywane przez aplikacje kliencką do stworzenia grafu. Wartością powinien być obiekt zawierający tablice dla właściwości `links` oraz `nodes`.

Pod właściwością `table` powinien znajdować się obiekt, zawierający informacje wykorzystywane przez klienta do prezentacji wyniku zapytania w formie tabeli. Obiekt powinien zawierać dwie właściwości: `headers` i `rows`. Właściwość `headers` zawiera tablicę z nagłówkami kolumn, a właściwość `rows` tablice, których liczba elementów odpowiada liczbie nagłówków, a każdy element jest zawartością komórki tablicy.

Struktura obiektu zwracanego przez metody odpytujące bazę danych została przedstawiona na Listingu 11.

Listing 11. Struktura danych odpowiedzi zwracanej do klienta po wykonaniu zapytania do bazy danych

```
{
  "graph": {
    "links": [
      {
        "id": "<id_węzła>",
        "source": "<id_węzła_źródłowego>",
        "target": "<id_węzła_docelowego>",
        "type": "<typ_relacji>",
        "properties": { "<klucz_właściwości>": "<wartość_właściwości>", ... }
      },
      ...
    ],
    "nodes": [
      {
        "id": "<id_węzła>",
        "labels": ["<etykieta_węzła_1>", "<etykieta_węzła_2>", ...],
        "properties": { "<klucz_właściwości>": "<wartość_właściwości>", ... }
      },
      ...
    ]
  },
  "table": {
    "headers": ["<nagłówek_1>", ..., <nagłówek_n>],
    "rows": [
      ["<zawartość_kolumny_1>", ..., < zawartość_kolumny_n>],
      ...
    ]
  }
}
```

6. Klient

Rolę klienta w rozwiązaniu spełnia aplikacja webowa. Aplikacja została napisana z wykorzystaniem języków HTML, CSS i JavaScript.

Kod HTML aplikacji nie jest skomplikowany (patrz Listing 12). W znaczniku `head` ładowane są 2 arkusze stylów. Jeden z nich zawiera ogólne reguły CSS używane w aplikacji, drugi jest używany przez elementy tworzone przez bibliotekę Notyf. W znaczniku `body` poza znacznikiem `script` znajduje się tylko jeden element `div`. Referencja do niego jest użyta w skrypcie odpowiadającym za inicjalizację renderera. Skrypt /dist/main.js zawiera całą logikę aplikacji. Jest on wynikiem transpilacji wykonanej przez transpilator Webpack.

Listing 12. Kod HTML aplikacji

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Graph Database VR Interface</title>
    <link rel="stylesheet" href="/css/style.css">
    <link rel="stylesheet" href="/css/notyf.css">
    <link rel="icon" type="image/png" href="/images/favicon196.png"/>
</head>
<body>
    <div id="renderer" style="user-select: none;"></div>
    <script src="/dist/main.js"></script>
</body>
</html>
```

6.1. Podsumowanie możliwości:

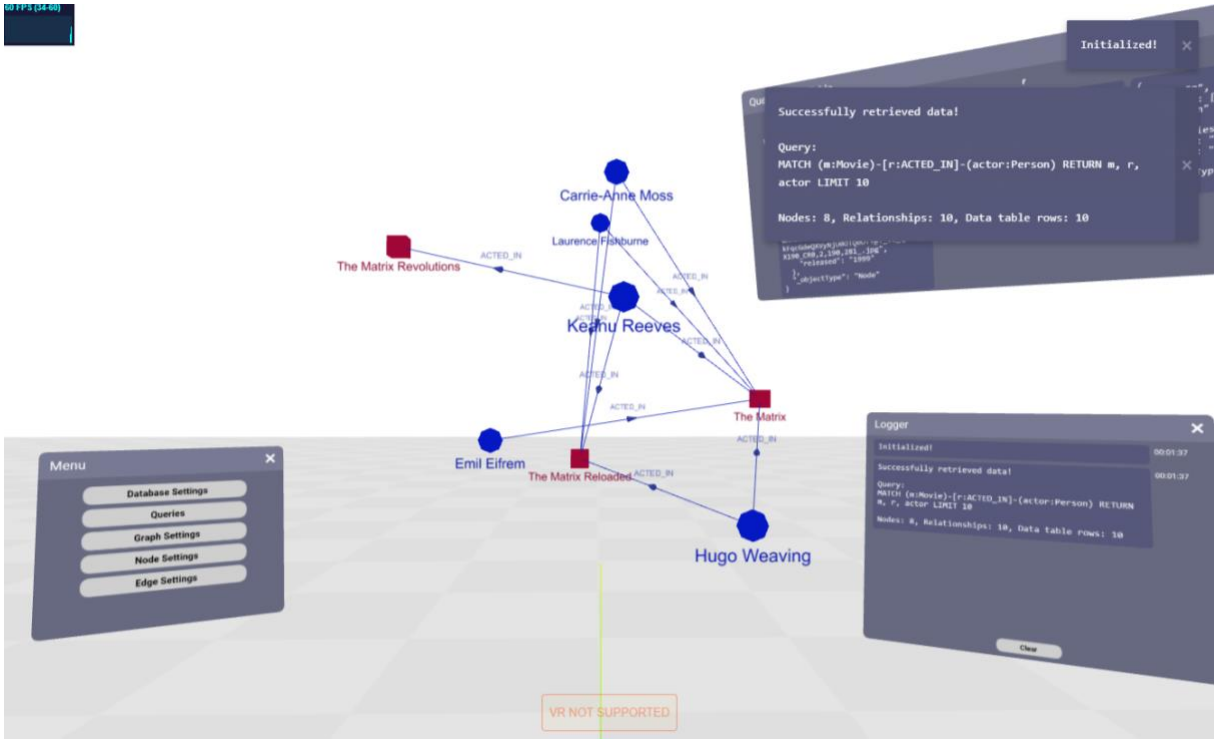
W kolejnych podrozdziałach szczegółowo przedstawiono funkcjonalności aplikacji klienckiej. Najważniejsze z nich można podsumować następująco:

- Aplikacja może być obsługiwana na dwa sposoby. Pierwszym z nich, który gwarantuje lepsze wrażenia, jest korzystanie z użyciem urządzenia VR (tryb VR). Drugim sposobem jest obsługa z pomocą myszy i klawiatury (tryb Desktop).
- Niezależnie od trybu, użytkownik może zmieniać swoje położenie na scenie. Oba tryby pozwalają na poruszanie się w trybie lotu. Tryb VR pozwala dodatkowo na zmianę pozycji z wykorzystaniem teleportacji, co może być wygodniejszym doświadczeniem dla wielu użytkowników.
- Po wyborze bazy danych, z którą użytkownik chce się komunikować, można konstruować zapytania w języku używanym przez daną bazę. Wprowadzanie wspomaga predykcja tekstu na podstawie słów kluczowych występujących w danym języku i metadanych z bazy. Ponadto możliwa jest konwersja mowy na tekst.
- Użytkownik może przeglądać ostatnio wysłane zapytania i zapisywać te, które chciałby użyć ponownie w przyszłości.
- Wynik zapytania można analizować w formie tabeli po przywołaniu odpowiedniego okna lub w formie grafu.
- Użytkownik może dostosować, w jaki sposób dane są prezentowane na grafie. Zmiana ustawień może wpłynąć m. in. na kształt, wielkość i kolor węzłów, właściwości wykorzystywane przy wyświetlaniu etykiet krawędzi i węzłów na grafie, skalę grafu czy przestrzeń, w której się znajduje (2D lub 3D).
- Aplikacja umożliwia podjęcie interakcji z grafem. Po najechaniu na węzeł lub krawędź zostaną wyświetlone szczegóły dotyczące obiektu. Węzły można przesuwać w przestrzeni, a po ich kliknięciu i przytrzymaniu zostanie wysłane zapytanie, którego wynikiem jest graf zawierający wybrany węzeł i wszystkie węzły, z którymi ma wspólne krawędzie.

6.2. Start

Po załadowaniu aplikacji całość strony wypełnia renderer, który wyświetla scenę. Na dole strony jest widoczny przycisk. Jeśli do komputera jest podpięte urządzenie VR i przeglądarka internetowa wspiera API WebXR, przycisk zawiera tekst „ENTER VR”. Po jego kliknięciu rozpoczyna się sesja WebXR i wyświetlanie obrazu w urządzeniu VR. W innym przypadku przycisk zawiera tekst „VR NOT SUPPORTED” i nie ma możliwości jego kliknięcia. Wtedy pozostaje obsługa w alternatywnym trybie Desktop.

Jeśli użytkownik korzystał już wcześniej z aplikacji to w obiekcie `localStorage`[1] mogą znajdować się dane zawierające informacje o ostatnio używanym połączeniu z bazą danych oraz ostatnio wykonanym zapytaniu. W takim przypadku zapytanie zostanie wykonane automatycznie i na podstawie zwróconych danych zostanie wygenerowany graf. Rysunek 11 przedstawia aplikację po uruchomieniu i wykonaniu takiego zapytania.



Rysunek 11. Widok po uruchomieniu aplikacji

[1] `localStorage` – jeden z mechanizmów Web Storage API. Właściwość `localStorage` wskazuje na obiekt pozwalający witrynom internetowym na przechowywanie danych w postaci kluczy i odpowiadających im wartości w postaci tekstu. Przechowywane w nim dane są dostępne nawet po zamknięciu przeglądarki.

6.3. Poruszanie się

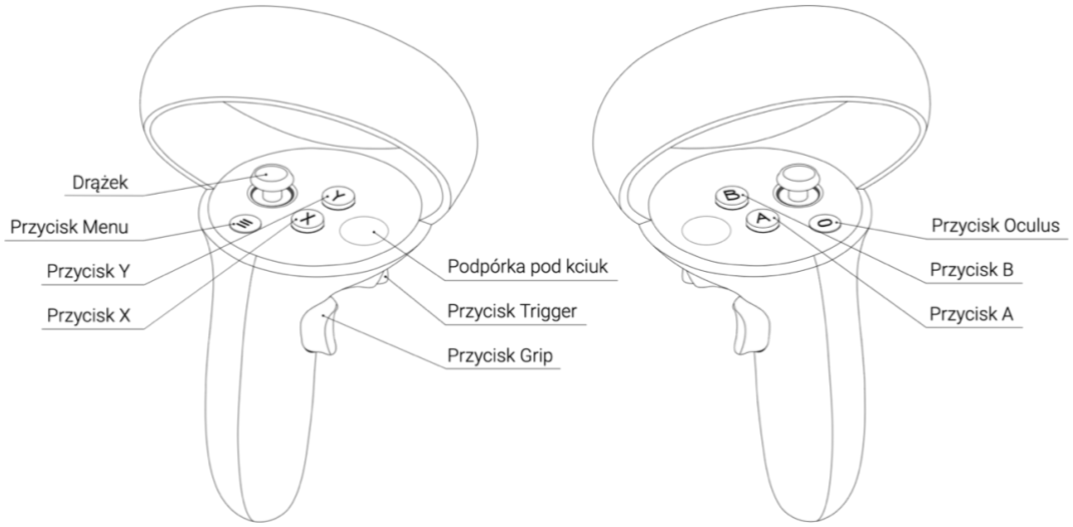
W rozwiązaniu jest możliwe poruszanie się po scenie. Pozwala to na zmianę pozycji, z której jest analizowany graf zawierający wynik zapytania. Ponadto w przestrzeni mogą być porozmieszczane okna interfejsu użytkownika. Zamiast zamykania okien aby otworzyć kolejne, można się przemieszczać w ich pobliże.

Przy obsłudze aplikacji w trybie Desktop, poruszanie odbywa się przez sterowanie pozycją i orientacją kamery. Tabela 4 przedstawia schemat sterowania z wykorzystaniem klawiatury i myszy.

Tabela 4. Schemat sterowania w trybie Desktop

Urządzenie	Klawisz / Przycisk	Akcja
Klawiatura	W	Ruch kamery do przodu
	S	Ruch kamery do tyłu
	A	Ruch kamery w lewo
	D	Ruch kamery w prawo
Mysz	lewy przycisk myszy	Interakcja z elementami interfejsu użytkownika
	prawy przycisk myszy	Zmiana orientacji kamery

Podczas korzystania w trybie VR wykorzystywane są HMD oraz kontrolery. Aplikacja została napisana z myślą o goglach Oculus Quest 2 i zgodnych z nimi kontrolerami Oculus Touch. Rysunek 12 przedstawia schemat kontrolerów wraz z podpisanymi przyciskami.



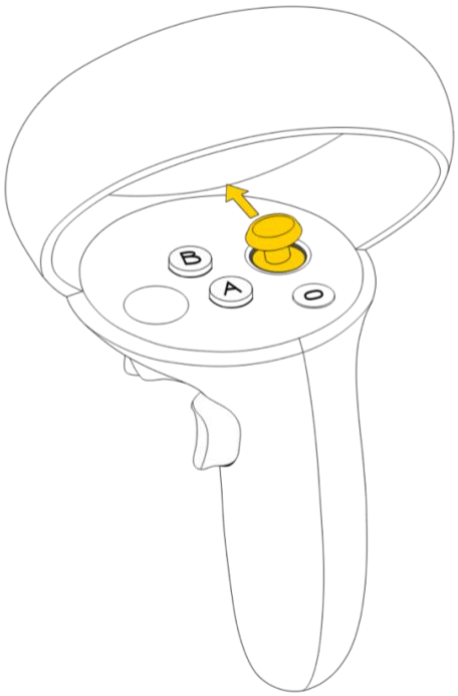
Rysunek 12. Schemat kontrolerów Oculus Touch zgodnych z Oculus Quest 2

Podczas poruszania się w wirtualnej rzeczywistości część użytkowników może doznać objawów choroby symulatorowej, takie jak [27]:

- ból oczu,
- senność,
- znużenie,
- apatia,
- ból i zawroty głowy o charakterze oszołomienia

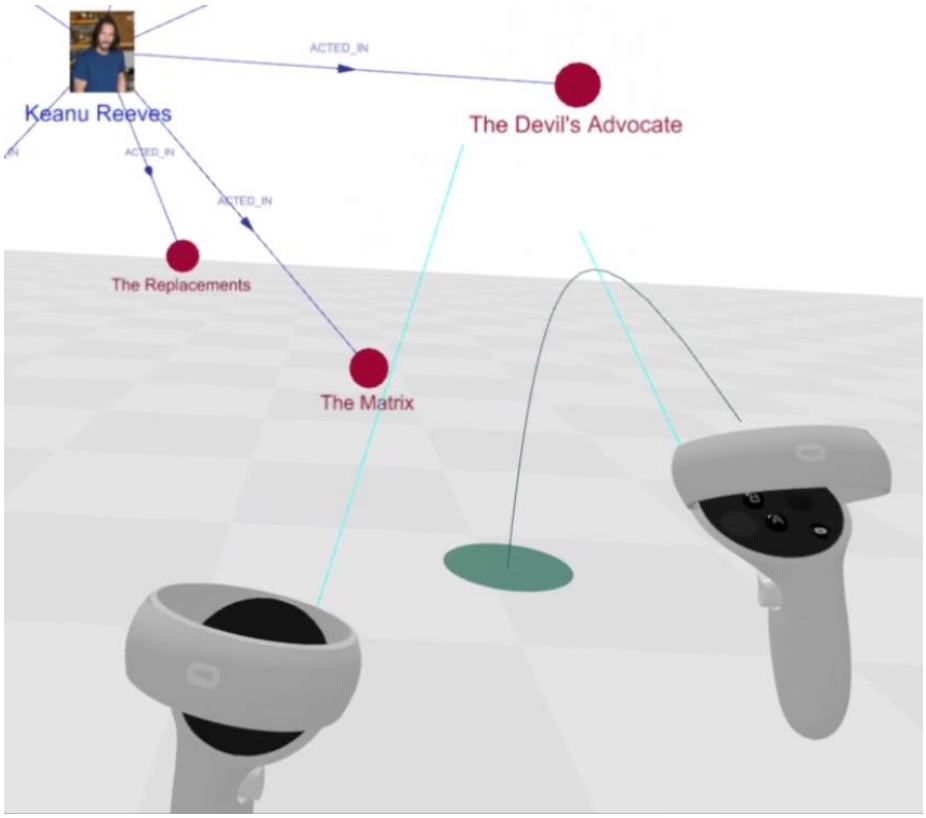
Badania [28] wskazują, że ryzyko wystąpienia objawów choroby symulatorowej jest uzależniane od sposobu poruszania się w wirtualnej rzeczywistości. Wynika z nich, że poruszanie się przy pomocy drążka na kontrolerze jest bardziej intensywnym doznaniem dla użytkownika niż teleportacja.

Biorąc pod uwagę komfort użytkownika zdecydowano się na implementację trzech trybów poruszania się: teleportacja, lot z wykorzystaniem orientacji głowy użytkownika oraz lot z wykorzystaniem orientacji kontrolera. Użytkownik ma możliwość przełączania się między nimi w dowolnym momencie. Domyślnym trybem jest teleportacja. Do poruszania się jest używany prawy kontroler.



Rysunek 13. Kontroler z wyróżnionym drążkiem, który jest wykorzystywany do poruszania się w trybie teleportacji i lotu

Teleportacja umożliwia szybką i komfortową dla użytkownika zmianę pozycji w przestrzeni. Aby przeteleportować się w dane miejsce, należy wychylić drążek na prawym kontrolerze do przodu i przytrzymać go w tej pozycji (patrz Rysunek 13). Następnie należy wskazać miejsce docelowe za pomocą zielonego wskaźnika tak jak przedstawiono na Rysunku 14 i puścić drążek, aby wrócił do swojej naturalnej pozycji. W momencie puszczenia drążka następuje zmiana lokalizacji użytkownika na scenie.



Rysunek 14. Zrzut ekranu uchwycony chwilę przed zmianą pozycji z użyciem teleportacji. Zielony wskaźnik na podłożu wskazuje miejsce docelowe.

Poza teleportacją możliwe jest poruszanie się w trybie lotu dostępnym w dwóch wariantach:

- pierwszy bazuje na kierunku, w którym zwrócona jest głowa użytkownika. Aby zmienić kierunek lotu użytkownik musi spojrzeć na miejsce, w którym chce się znaleźć. Wychylenie drążka do przodu powoduje przemieszczanie się użytkownika w kierunku, w którym jest zwrócony. Wychylenie drążka do tyłu pozwala na ruch w przeciwnym kierunku.
- drugi wariant działa bardzo podobnie. Jedyna różnica polega na tym, że do określenia kierunku lotu wykorzystywana jest orientacja prawego kontrolera. Pozwala to użytkownikowi na jednoczesne poruszanie się i rozglądanie się po scenie. Wadę może stanowić fakt, że jest to

najbardziej intensywny dla użytkownika ze sposobów poruszania się dostępnych w aplikacji. Może szybko doprowadzić do objawów choroby symulatorowej.

6.4. Elementy interfejsu użytkownika

Niezależnie od trybu używanego do obsługi aplikacji, niemal wszystkie interakcje podejmowane są z obiektami 3D. W trym rozdziale zostaną przedstawione elementy interfejsu użytkownika, z którymi użytkownik ma kontakt podczas korzystania z aplikacji.

6.4.1. Okna

Większość z elementów interfejsu użytkownika ma postać okien reprezentowanych przez oteksturowane płaszczyzny. Źródłem dla tekstury jest obraz wygenerowany z pomocą Canvas API. Każde okno może być zamknięte po kliknięciu ikony „X" na pasku tytułu. Po kliknięciu „strzałki w lewo" na pasku tytułu obecnie otwarte okno zostaje zamknięte i otwierane jest poprzednio otwarte okno.

Okna dostosowują swoją orientację w przestrzeni, w taki sposób, aby zawsze być zwrócone do użytkownika. Wyjątek stanowi sytuacja, kiedy użytkownik znajduje się jego pobliżu. W takim przypadku okno pozostaje nieruchome, aby ułatwić użytkownikowi interakcję, zwłaszcza z małymi elementami.

W trybie Desktop do interakcji z nimi używa się myszy. Po najechaniu na element, który można kliknąć kursor myszy zmieni się na rękę z wyciągniętym palcem wskazującym (zmiana właściwości `cursor` na wartość `pointer` w CSS). Do klikania elementów wykorzystywany jest lewy przycisk myszy, a do przewijania zawartości okna kółko myszy.

W trybie VR do interakcji są wykorzystywane promienie wychodzące z kontrolerów, które pełnią rolę wskaźników. Aby wybrać element najpierw należy wskazać na niego promieniem. Po najechaniu na element, który można kliknąć promień zmieni kolor. Następnie należy przycisnąć trigger na kontrolerze, który wskazuje element. Do przewijania zawartości używane są drążki na kontrolerach.

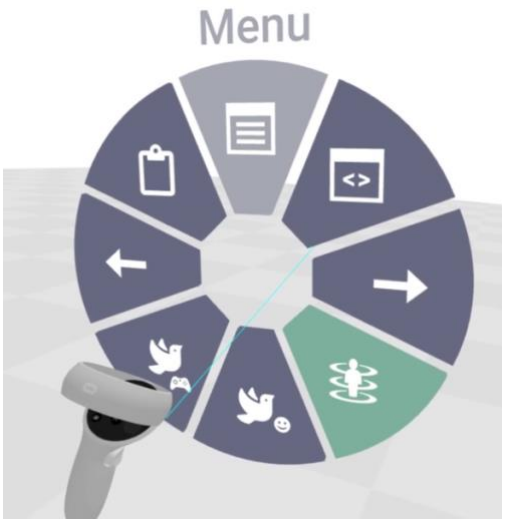
Ponadto w trybie VR okna mogą być przenoszone przez użytkownika. Aby to zrobić należy nacelować na dane okno i przytrzymać przycisk grip. Okno zacznie podążać za ruchem kontrolera. Gdy okno osiągnie pożądana pozycję wystarczy puścić przycisk, aby pozostawić je w obecnej pozycji.

6.4.2. Menu wywoływane za pomocą lewego kontrolera

W celu zapewnienia wygodnego dostępu do podjęcia takich działań jak: zmiana trybu poruszania się, nawigacja po wykonanych zapytaniach czy możliwość przywołania niektórych elementów interfejsu użytkownika zaimplementowane zostało specjalne menu. Ma ono kształt koła, które podąża

za kontrolerem. Każda opcja jest reprezentowana przez wycinek koła z odpowiadającą jej ikoną. Wygląd menu został zaprezentowany na Rysunku 15.

Menu może zostać wywołane poprzez wychylenie drążka na lewym kontrolerze w dowolnym kierunku. Powrót drążka do swojej naturalnej pozycji skutkuje schowaniem menu. Opcje dostępne do wyboru zostały opisane w Tabeli 5. Aby wybrać konkretną opcję należy wychylić drążek w kierunku, w którym dana opcja się znajduje. Spowoduje to zmianę koloru modelu 3D reprezentującego opcję z szarego na jasny szary. Następnie należy potwierdzić wybór wciskając przycisk trigger na lewym kontrolerze. Kolor modelu 3D zostanie zmieniony na jasny zielony (patrz Rysunek 16), a opcja zostanie aktywowana.



Rysunek 15. Menu wywołane z pomocą lewego kontrolera. Obecnie zaznaczona opcja jest wyróżniona jaśniejszym odcieniem szarego. Kolor zielony wskazuje obecnie aktywny tryb poruszania.

Rysunek 16. Menu po potwierdzeniu wyboru opcji

Tabela 5. Opis działania opcji dostępnych w menu wywoływanym lewym kontrolerem

Ikona	Działanie
	Zmiana trybu poruszania się na tryb latania według orientacji kontrolera
	Zmiana trybu poruszania się na tryb latania według orientacji głowy użytkownika
	Zmiana trybu poruszania się na tryb teleportacji
	Wykonuje poprzednie zapytanie do bazy danych
	Wykonuje następne zapytanie do bazy danych
	Przywołuje okno z menu głównym
	Przywołuje okno z logami
	Przywołuje okno z tabelą zawierającą wynik zapytania

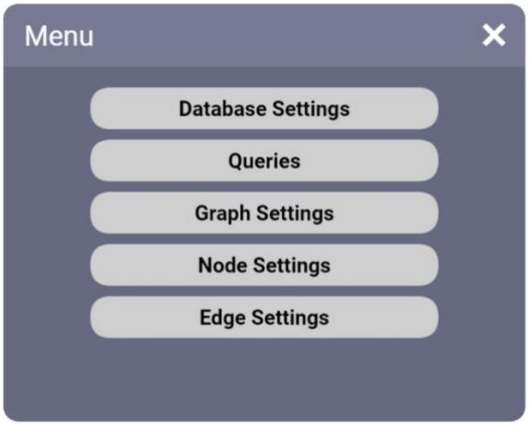
W trybie Desktop nie ma możliwości wywołania powyższego menu. Wszystkie czynności, poza zmianą trybu poruszania się, która nie jest możliwa w tym trybie, można wykonać za pomocą klawiatury używając kombinacji klawiszy z Tabeli 6.

Tabela 6. Kombinacje klawiszy uruchamiające czynności dostępne w menu wywoływanym lewym kontrolerem

Kombinacja klawiszy	Działanie
M	Przywołuje okno z menu głównym
K	Przywołuje okno z tabelą zawierającą wynik zapytania
L	Przywołuje okno z logami
Ctrl + Z	Wykonuje poprzednie zapytanie do bazy danych
Ctrl + Shift + Z	Wykonuje następne zapytanie do bazy danych

6.4.3. Okno Menu

Okno Menu pozwala na dostęp do wielu innych okien odpowiedzialnych między innymi za ustawienia dotyczące grafu, bazy danych i budowanie zapytań. Wygląd okna przedstawiono na Rysunku 17.

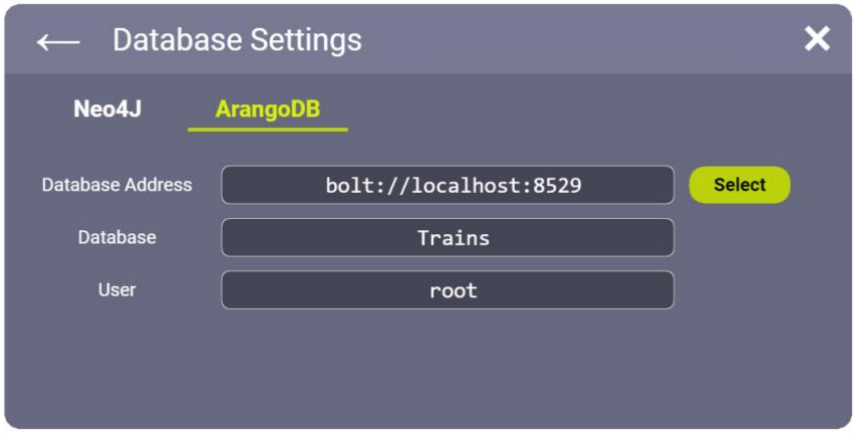


Rysunek 17. Okno Menu

6.4.4. Ustawienia bazy danych

W oknie widoczne są wszystkie dane konfiguracyjne dla połączenia z bazą danych. Na Rysunku 18 widoczne są nazwy dla dostępnych połączeń: „Neo4J” oraz „ArangoDB”. Klikając na nie można przełączać się między zakładkami. Każda zakładka zawiera dane, które są wykorzystywane przy połączeniu z bazą danych. Widoczne są wszystkie elementy z konfiguracji zaprezentowanej na Listingu 8 poza hasłem, ponieważ dla hasła wartość właściwości `hide` została ustawiona na `true`. Dane wykorzystywane do połączenia z bazą danych nie są edytowalne z poziomu aplikacji klienckiej.

Ponadto w zakładce może być wyświetlony przycisk Select. Jest on wyświetlany w zakładkach połączeń, które nie są obecnie wykorzystywane. Umożliwia zmianę połączenia, z którego chcemy korzystać. W tym samym czasie może być aktywne tylko jedno połączenie z bazą danych.



Rysunek 18. Okno ustawień bazy danych

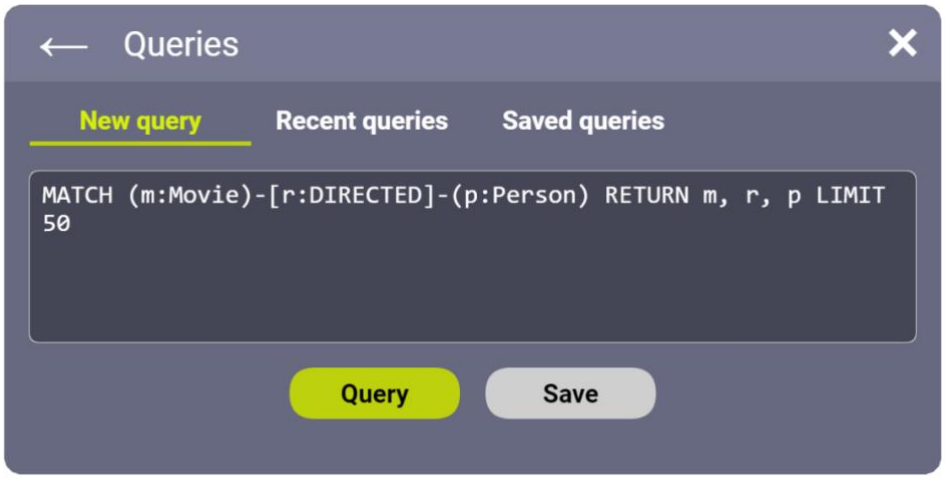
6.4.5. Zapytania

Zapytanie (inaczej kwerenda) służą do komunikacji z bazą danych. Za ich pomocą można pobierać i modyfikować dane przechowywane w bazie. Zapytania muszą być budowane w oparciu o składnię języka używanego w konkretnej bazie danych.

Okno Zapytania (Queries) zostało podzielone na 3 zakładki:

1. Nowe zapytanie (New query)

   W tej zakładce użytkownik może wykonać zapytanie do bazy danych oraz je zapisać. Wygląd okna z aktywną zakładką przedstawiono na Rysunku 19. Na środku widoczny jest obszar tekstowy, którego zawartość może być edytowana. W tym celu należy kliknąć w dowolnym miejscu na obszarze tekstowym, co spowoduje otworzenie okna z klawiaturą. Na dole widoczne są dwa przyciski: „Query” oraz „Save”. Po kliknięciu przycisku „Query”, zapytanie zostanie wysłane na serwer, który przekieruje je do bazy danych. Po otrzymaniu wyniku w aplikacji klienckiej, na jego podstawie jest generowany graf. Przycisk „Save” służy do zapisywania zapytań.

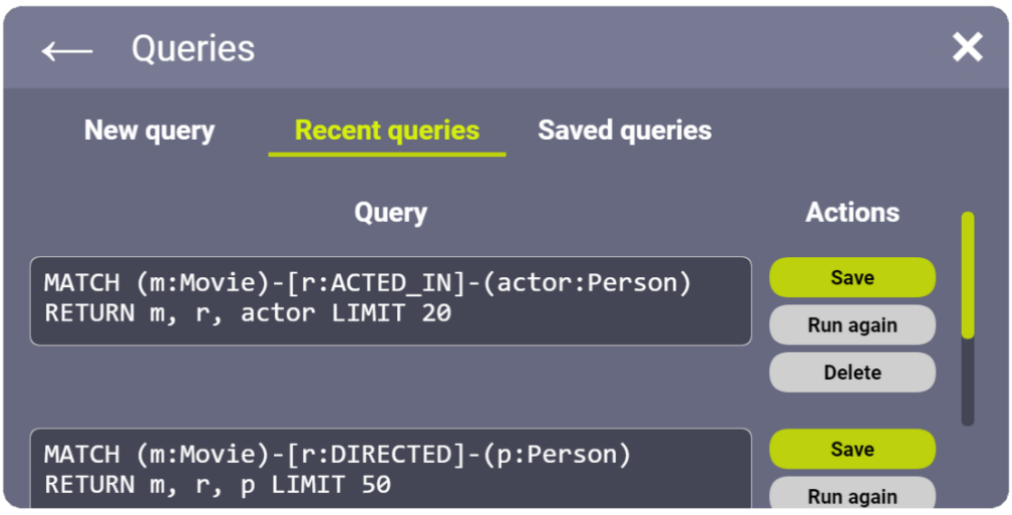


Rysunek 19. Okno Zapytania z aktywną zakładką Nowe zapytanie

2. Ostatnie zapytania (Recent queries)

   Zakładka zawiera listę ostatnio wykonanych zapytań i jest podzielona na dwie kolumny (patrz Rysunek 20). Pierwsza kolumna zawiera nieedytowalny obszar tekstowy zawierający treść zapytania. Druga kolumna zawiera przyciski z akcjami, które użytkownik może wykonać:

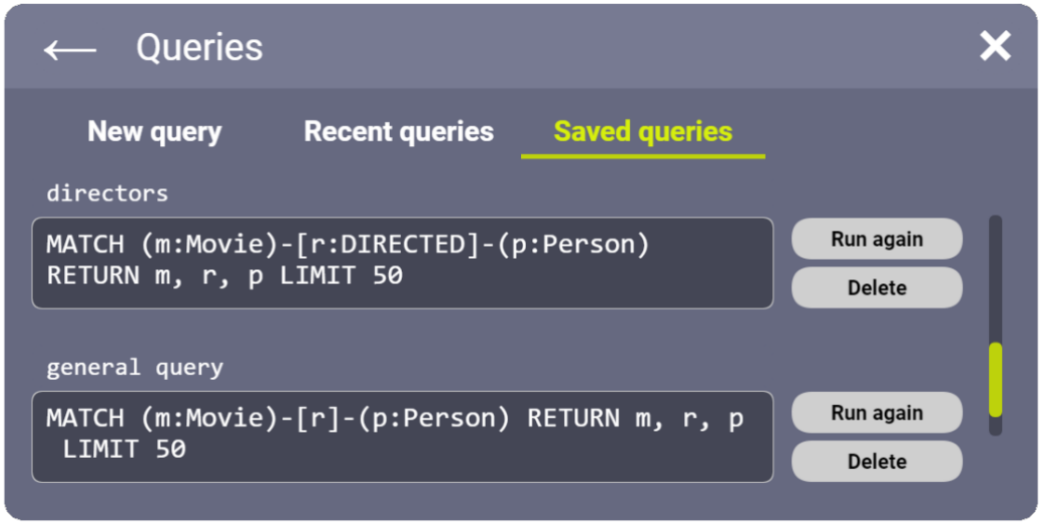
   - Przycisk „Save” zapisuje zapytanie
   - Przycisk „Run again” kopiuje zawartość zapytania, otwiera zakładkę „New query” i wkleja zawartość zapytania do obszaru tekstowego.
   - przycisk „Delete” usuwa wiersz z listy



Rysunek 20. Okno Zapytania z aktywną zakładką Ostatnie zapytania

3. Zapisane zapytania (Saved Queries)

   Wygląd zakładki przedstawiono na Rysunku 21. Zawiera ona listę zapisanych zapytań i jest podzielona na dwie kolumny. Pierwsza kolumna zawiera dwa edytowalne obszary tekstowe dla każdego zapytania. Pierwszy z nich zawiera nazwę zapisanego zapytania, a drugi jego zawartość. Domyślną nazwą dla każdego nowo zapisanego zapytania jest informacja, kiedy zostało zapisane, na przykład: „Query saved at 16.01.2022, 16:24:48”. Przyciski w kolumnie akcji działają analogicznie do przycisków akcji w zakładce „Recent Queries”.



Rysunek 21. Okno Zapytania z aktywną zakładką Zapisane zapytania

6.4.6. Klawiatura

Aby umożliwić budowanie zapytań, aplikacja musi pozwalać na wprowadzanie tekstu. W trybie Desktop oczywistym wyborem jest wykorzystanie klawiatury podłączonej do komputera. W trybie VR użytkownik ma na sobie HMD, więc nie widzi klawiatury. Sprawia to problemy przy jej lokalizowaniu i wprowadzaniu tekstu bez pomyłek, kiedy osoba nie jest dobrze zaznajomiona z układem klawiatury.

Celem zapewnienia wygodnego pisania w trybie VR zdecydowano się na dodanie wirtualnej klawiatury. Stian Kongsvik w swojej pracy [29] porównał cztery metody wprowadzania tekstu w wirtualnej rzeczywistości:

- klawiatura wykorzystująca rzutowanie promieni (ang. ray casting) – klawisze są wskazywane za pomocą promieni, na wzór wskaźnika laserowego
- klawiatura przypominająca perkusję – użytkownik trzyma w rękach pałeczki perkusyjne i za ich pomocą wciska klawisze na klawiaturze

- klawiatura z wprowadzaniem wykorzystującym orientację głowy użytkownika do wskazania klawiszy
- klawiatura podzielona na dwie części, których klawisze są wskazywane z wykorzystaniem sygnałów wejściowych z gładzików kontrolerów

Najlepszymi pod względem szybkości wprowadzania okazały się klawiatura na wzór perkusji (średnio 21,01 słowa na minutę) oraz klawiatura z rzutowaniem promieni (średnio 16,65 słowa na minutę). Mimo gorszego wyniku w prototypie zdecydowano się na implementację klawiatury obsługiwaną za pomocą promieni.

Uznano tę metodę za prostszą w implementacji i bardziej intuicyjną. W aplikacji promienie są już wykorzystywane do nawigacji w oknach. W przypadku zdecydowania się na wprowadzanie perkusyjne należałoby zaimplementować mechanizm przełączania się między promieniami do nawigacji po oknach i pałeczkami perkusyjnymi. Ponadto aby było to intuicyjne dla użytkownika, powinno się sygnalizować z jakimi elementami może podjąć interakcję za pomocą promieni, a które wymagają użycia pałeczek. Użycie tej samej metody do nawigacji po elementach interfejsu użytkownika i wprowadzania tekstu eliminuje ten problem.

Rysunek 22. Zrzut ekranu przedstawiający okno Klawiatura

Okno klawiatury umożliwia edycję tekstu zawartego w edytowalnych obszarach tekstowych. W jego skład wchodzą:

- Elementy do obsługi konwersji mowy na tekst

  Funkcjonalność jest dostępna podczas korzystania z aplikacji w przeglądarce, która wspiera Web Speech API. Ma na celu ułatwienie wprowadzania danych w aplikacji. Aby rozpocząć korzystanie z transkrypcji należy kliknąć przycisk „Start" w prawym górnym rogu (patrz Rysunek 22). Po jego kliknięciu obszar tekstowy na lewo od przycisku zmieni kolor na zielony sygnalizując, że rozpoznawanie mowy jest aktywne. To w nim są wyświetlane rozpoznane słowa. Dodatkowo przycisk „Start" znika, a w jego miejscu pojawiają się trzy przyciski. Kolejno od lewej pozwalają one na: usunięcie ostatniego rozpoznanego słowa, przeniesienie rozpoznanych słów do obszaru tekstowego i zakończenie korzystania z transkrypcji. Zmiany w interfejsie po kliknięciu przycisku „Start" przedstawiono na Rysunku 23.

Rysunek 23. Elementy do obsługi konwersji mowy na tekst podczas aktywnej transkrypcji

- Obszar tekstowy

  Po otworzeniu okna z klawiaturą ten element zawiera zawartość tekstową edytowanego elementu. Umożliwia on śledzenie wszystkich zmian wprowadzanych w tekście. Poza tekstem jest widoczny również kursor, wskazujący miejsce, w którym jest prowadzona edycja. Zmiana pozycji kursora może zostać zmieniona przez kliknięcie na miejsce, w które ma zostać przeniesiony lub używając klawiszy strzałek na klawiaturze.

- Przyciski pozwalające na wstawienie tekstu z predykcji

  W celu ułatwienia wprowadzania zapytań w trybie VR zaimplementowano prosty system predykcji tekstu. Na podstawie wprowadzonej przez użytkownika części słowa dopasowywane są słowa kluczowe dla języka zapytań, nazwy właściwości występujących w węzłach i relacjach, etykiety węzłów oraz typy relacji. Słowa kluczowe używane do sugestii są pobierane w oparciu o obecnie wybrane połączenie z bazą danych. Ich zawartość determinuje wynik funkcji `getKeywords` w klasie obsługującej połączenie z bazą danych. Pozostałe dane pochodzą z wyników funkcji `getNodesMetadata` i `getEdgesMetadata`. Po kliknięciu

przycisku wprowadzane przez użytkownika słowo zostaje dokończone zgodnie z podpowiadanym słowem.

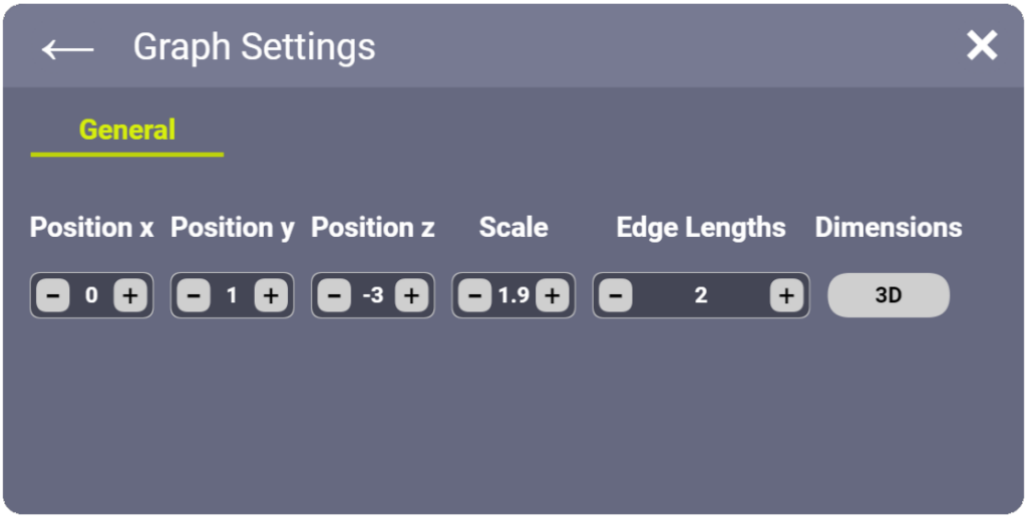
- Przyciski klawiatury

  Układ klawiatury został zaprojektowany z myślą o szybkim budowaniu zapytań w języku Cypher, który jest używany w bazach danych Neo4J. Część przycisków została oznaczona niebieskim kolorem i zawiera słowa kluczowe dla tego języka. Korzystanie z tych przycisków nie jest wymogiem. Alternatywnym sposobem wprowadzania tekstu jest korzystanie z klawiatury podłączonej do komputera.

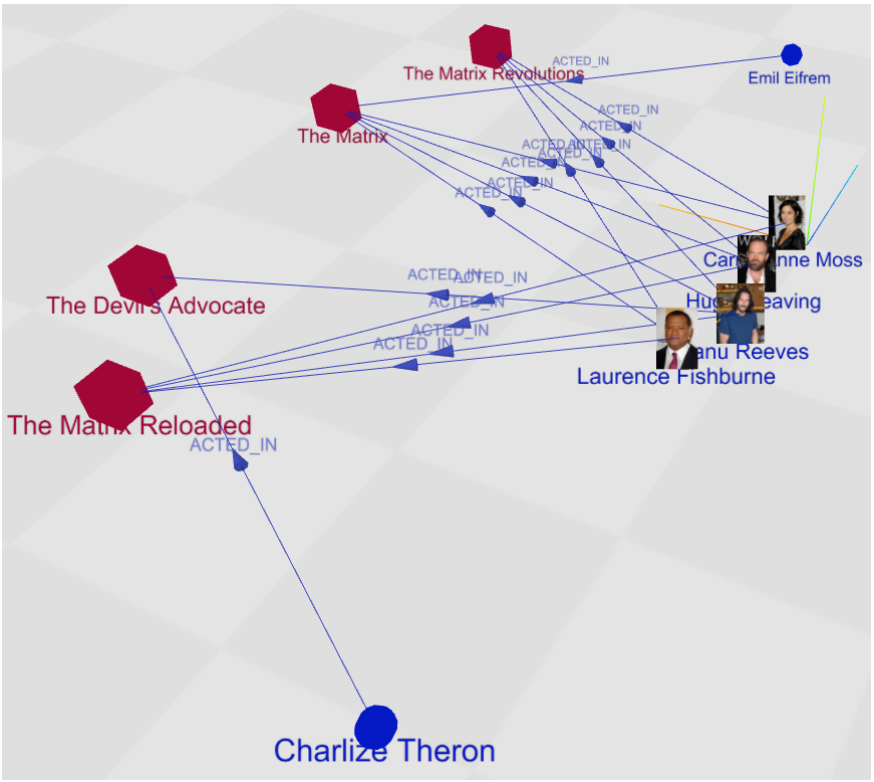
- Przyciski akcji
  - Przycisk „Confirm” zatwierdza edycję i zamyka okno z klawiaturą.
  - Przycisk „Cancel” odrzuca wprowadzone zmiany i zamyka okno z klawiaturą.

6.4.7. Ustawienia grafu

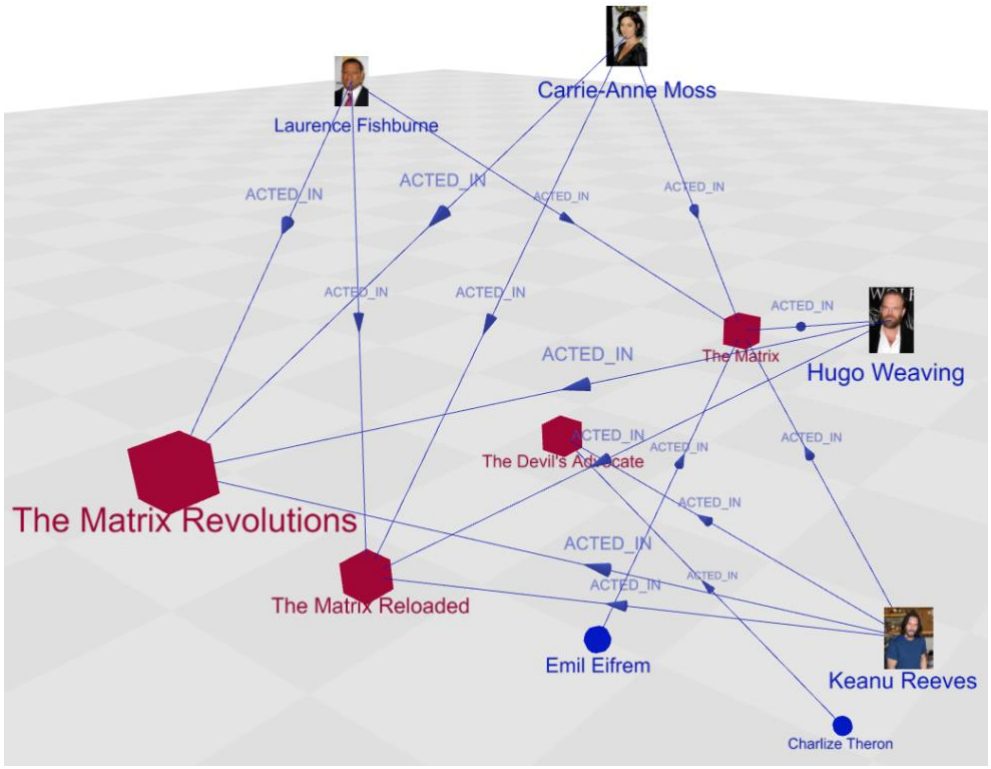
Okno pozwala na zmianę pozycji grafu w przestrzeni na osiach X, Y, Z. Ponadto możliwa jest modyfikacja skali grafu oraz długości krawędzi łączących wierzchołki. Rysunek 24 prezentuje wygląd okna. Przycisk w kolumnie „Dimensions” pozwala na wybór czy graf ma być generowany w przestrzeni 2D czy 3D. Przy klikaniu tekst przycisku zmienia się naprzemiennie, a graf rysowany jest na nowo w przestrzeni wskazywanej przez przycisk. W przypadku wybrania opcji „2D” graf jest generowany na płaszczyźnie równoległej do podłoża. Rysunek 25 przedstawia graf wygenerowany po wyborze opcji „2D”. Ten sam graf po wyborze opcji „3D” można zaobserwować na Rysunku 26.



Rysunek 24. Okno Ustawienia grafu



Rysunek 25. Graf wygenerowany w przestrzeni 2D



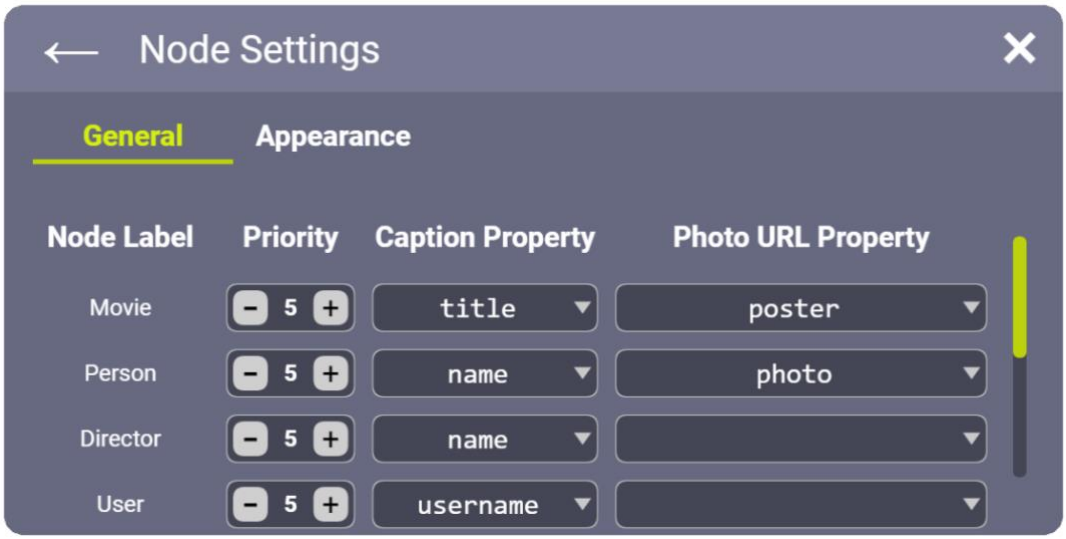
Rysunek 26. Graf z Rysunku 25 wygenerowany w przestrzeni 3D

6.4.8. Ustawienia węzłów

Okno ustawień wpływających na węzły występujące w grafie składa się z dwóch zakładek:

- Ogólne (General)

  Zakładkę przedstawiono na Rysunku 27. Zawiera ona tabelę z ustawieniami dotyczącymi węzłów o danej etykiecie (patrz Tabela 7). W pierwszej kolumnie tabeli znajdują się wszystkie etykiety obecne w bazie. Informacja o nich jest zwracana przez funkcję `getNodesMetadata` znajdującą się w klasie obsługującej połączenie z bazą danych.

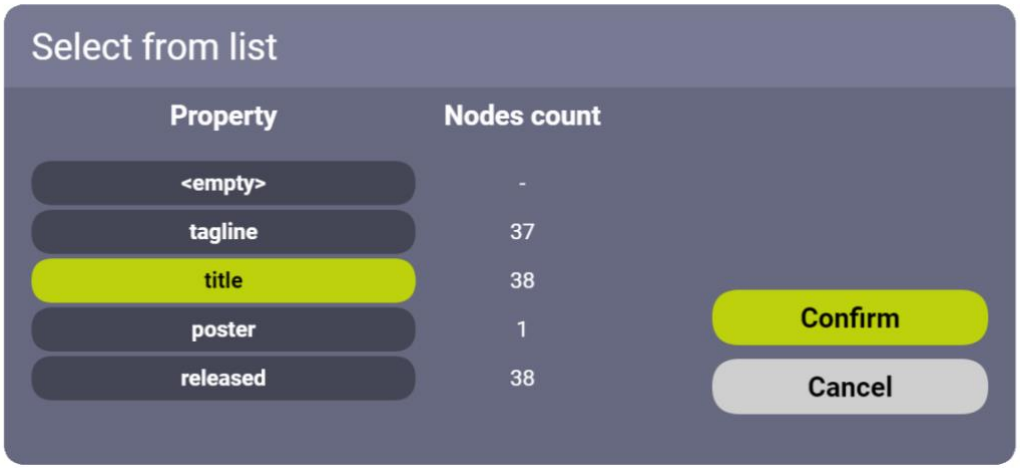


Rysunek 27. Okno Ustawienia węzłów z aktywną zakładką Ogólne

Tabela 7. Zastosowanie pól występujących w wierszach tabeli w oknie Ustawienia węzłów w zakładce Ogólne

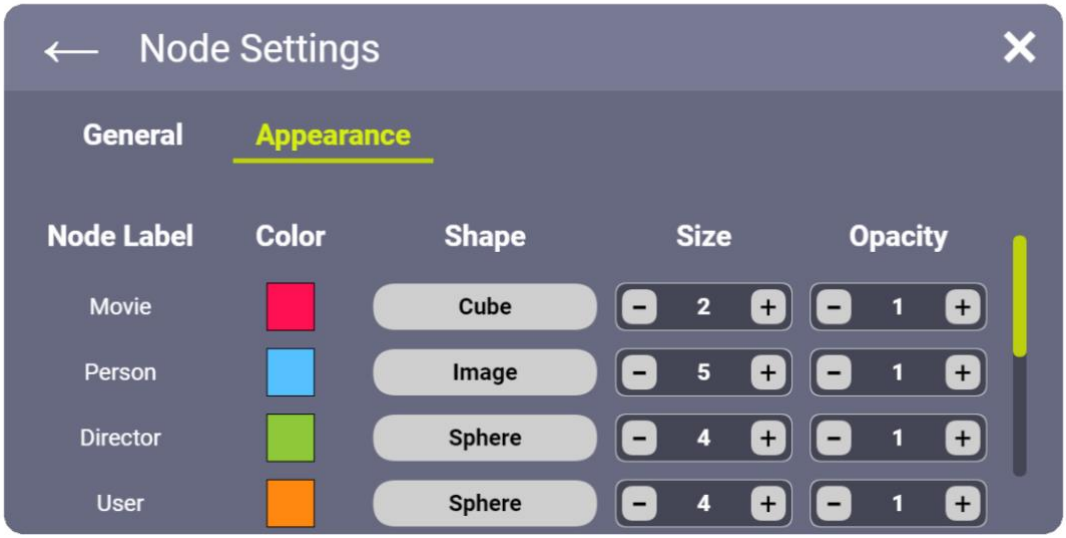
Pole	Zastosowanie
Node Label	Wyświetla etykietę, której dotyczą ustawienia
Priority	Wartość tego pola ma zastosowanie w przypadku węzłów posiadających więcej niż jedną etykietę. Takie węzły używają ustawień etykiety, dla której wartość priorytetu ma niższą wartość.
Caption Property	Decyduje o właściwości, której wartości są wyświetlane jako tekst podpisujący węzły o danej etykiecie. Po kliknięciu pola otwiera się okno z listą właściwości do wyboru (Rysunek 28).
Photo URL Property	Decyduje o właściwości, której wartość jest interpretowana jako adres URL prowadzący do obrazu. Węzły z tą etykietą będą wyświetlane jako obraz, jeśli użytkownik ustawi odpowiednią opcję odpowiadającą za kształt węzłów w kolejnej zakładce. Po kliknięciu pola otwiera się okno z listą właściwości do wyboru (Rysunek 28).

Okno z listą właściwości do wyboru, przedstawione na Rysunku 28, zawiera wszystkie właściwości występujące w węzłach o danej etykiecie w bazie danych. Te informacje również pochodzą z funkcji zwracającej metadane węzłów. Dzięki temu użytkownik nie musi pamiętać dokładnych nazw wszystkich właściwości dla węzłów w bazie danych. Ponadto prezentowana jest ilość węzłów z konkretną etykietą, w których znajduje się dana właściwość. Wybór można potwierdzić przyciskiem „Confirm” lub anulować przyciskiem „Cancel”.



Rysunek 28. Okno z listą właściwości do wyboru, które jest otwierane po kliknięciu pola odwołującego się do właściwości występującej w węzłach

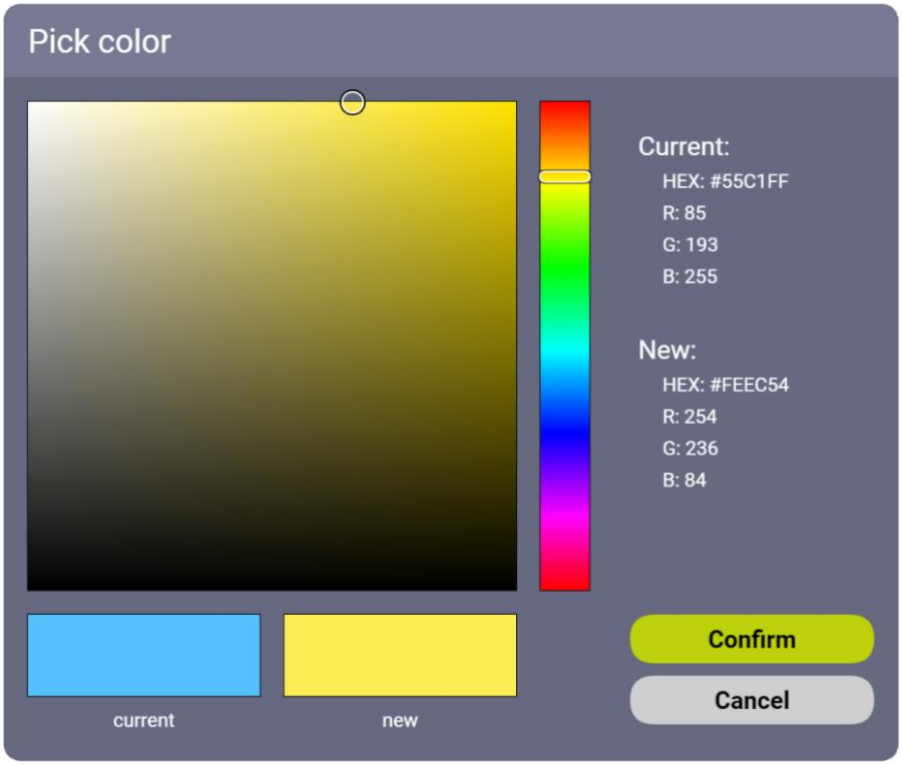
- Wygląd (Appearance)
  W tej zakładce znajduje się tabela z ustawieniami dotyczącymi tego jak mają być prezentowane węzły o danej etykiecie. Zakładka została przedstawiona na Rysunku 29, a zastosowanie pól w wierszach tabeli opisano w Tabeli 8.



Rysunek 29. Okno Ustawienia węzłów z aktywną zakładką Wygląd

Tabela 8. Zastosowanie pól występujących w wierszach tabeli w oknie Ustawienia węzłów w zakładce Wygląd

Pole	Zastosowanie
Node Label	Wyświetla etykietę, której dotyczą ustawienia
Color	Wyświetla kolor węzłów o danej etykiecie. Można go zmienić klikając na kwadrat wypełniony wybranym kolorem. Po kliknięciu zostanie otwarte okno wyboru koloru (Rysunek 30).
Shape	Jest to przycisk służący do zmiany kształtu węzłów o danej etykiecie. Zmian dokonuje się klikając na przycisk, do momentu aż wyświetli pożądaną wartość (do wyboru są: sześcian, kula, obraz).
Size	Pozwala zmienić rozmiar węzłów o danej etykiecie.
Opacity	Pozwala zmienić nieprzezroczystość węzłów o danej etykiecie.



Rysunek 30. Okno wyboru koloru

6.4.9. Ustawienia krawędzi

Powyższe okno pozwala na dostosowanie wyglądu krawędzi. W oknie znajduje się tabela, w której każdy wiersz zawiera ustawienia dla krawędzi danego typu. Wygląd okna przedstawiono na Rysunku 31, a zastosowanie pól w wierszach tabeli opisano w Tabeli 9.

Rysunek 31. Okno Ustawienia krawędzi

Tabela 9. Zastosowanie pól występujących w wierszach tabeli w oknie Ustawienia krawędzi

Pole	Zastosowanie
Relationship Type	Wyświetla typ relacji, którą reprezentują krawędzie
Caption	Edytowalne pole tekstowe, którego wartość jest wyświetlana jako tekst podpisujący krawędź danego typu.
Color	Wyświetla kolor krawędzi danego typu. Można go zmienić klikając na kwadrat wypełniony wybranym kolorem. Po kliknięciu zostanie otwarte Okno wyboru (Rysunek 30).
Line width	Możliwość ustawienia grubości linii
Opacity	Możliwość ustawienia nieprzezroczystości linii

6.4.10. Okno szczegółów węzła

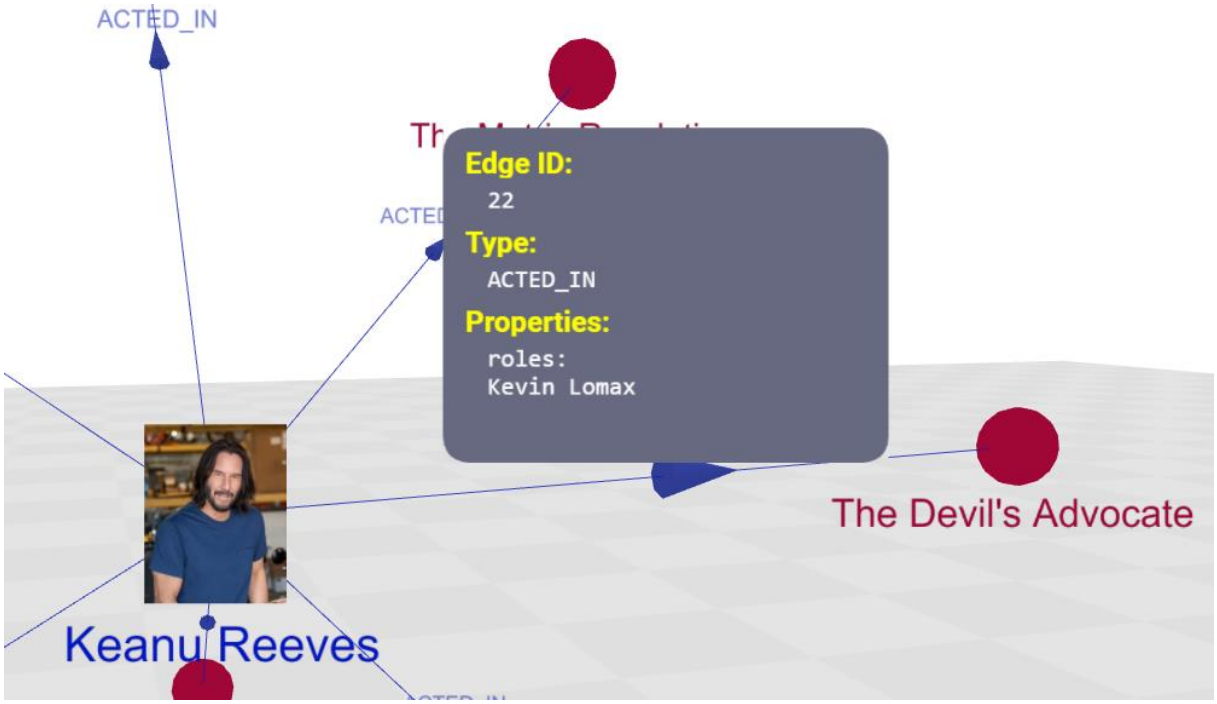
Zawiera informacje o identyfikatorze węzła, jego etykietach i wszystkich właściwościach w postaci listy z możliwością przewijania. Ponadto w górnej części okna znajduje się pasek postępu, który jest używany przy zapytaniach o węzeł, wszystkie jego relacje i węzły, z którymi te relacje posiada. Wygląd okna przedstawia Rysunek 32.

Jeżeli w Oknie ustawień węzłów wskazano właściwość zawierającą adres obrazu to ta właściwość jest wyświetlana jako pierwsza. Zamiast wartości tekstowej właściwości jest wyświetlany obraz, który wskazuje adres. Wyjątkiem jest sytuacja, kiedy obraz nie zostanie pomyślnie pobrany, np. przez niepoprawnie wprowadzony adres. W takim przypadku wyświetlana jest wartość tekstowa.

Rysunek 32. Okno szczegółów wierzchołka

6.4.11. Okno szczegółów relacji

Możliwy jest też podgląd szczegółów relacji łączących węzły. Okno szczegółów relacji zostało ukazane na Rysunku 33. Zawiera informacje o identyfikatorze, typie oraz właściwościach krawędzi. Są one prezentowane w postaci listy, która może być przewijana.



Rysunek 33. Okno szczegółów relacji

6.4.12. Powiadomienia

Powiadomienia zawierają informacje, które są przydatne dla użytkownika w trakcie korzystania z aplikacji. Może się z nich dowiedzieć między innymi czy aplikacji udało się pobrać metadane z bazy danych lub czy wysłane zapytanie zostało wykonane pomyślnie i w przypadku błędu poznać jego przyczynę.

Podczas korzystania z rozwiązania w wirtualnej rzeczywistości powiadomienia są pokazywane jako okna, które układają się w stos (patrz Rysunek 34). Aby upewnić się, że użytkownik nie przegapi jakiegoś powiadomienia, jedynym sposobem, aby powiadomienia znikły jest ich ręczne zamknięcie poprzez klikanie ikony „X” w ich prawym górnym rogu.

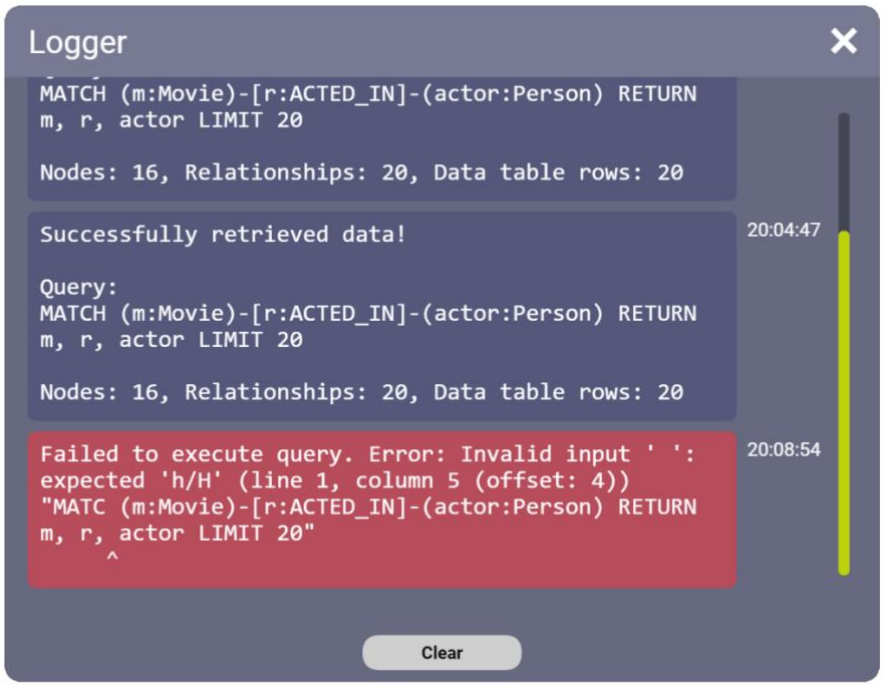
Ponadto jeśli użytkownik zmieni swoją pozycję w przestrzeni, stos powiadomień zacznie za nim podążać. Po chwili znajdzie się w odległości, z której użytkownik może komfortowo przeczytać przygotowane dla niego komunikaty. Stos powiadomień również obraca się wokół użytkownika, tak aby niezależnie w którym kierunku użytkownik się odwróci, po chwili znalazł się w polu jego widzenia.

Rysunek 34. Powiadomienia wyświetlane w aplikacji w trybie VR

Korzystając z aplikacji w trybie Desktop powiadomienia nie są wyświetlane w formie okien 3D. Zamiast nich zaimplementowano powiadomienia generowane z pomocą biblioteki Notyf, które są pokazywane w prawym górnym rogu strony. W przeciwieństwie do okien wyświetlanych w wirtualnej rzeczywistości są one automatycznie zamykane po 10 sekundach. Użytkownik może zamknąć powiadomienia wcześniej, jeśli kliknie ikonę „X” znajdującą się z prawej strony powiadomienia.

6.4.13. Logger

To okno pozwala na przegląd wszystkich powiadomień razem z informacją o czasie, w którym zostały wyświetlone. Są prezentowane w postaci listy z możliwością przewijania (patrz Rysunek 35). Przycisk w dolnej części okna pozwala na wyczyszczenie listy ze wszystkich powiadomień.

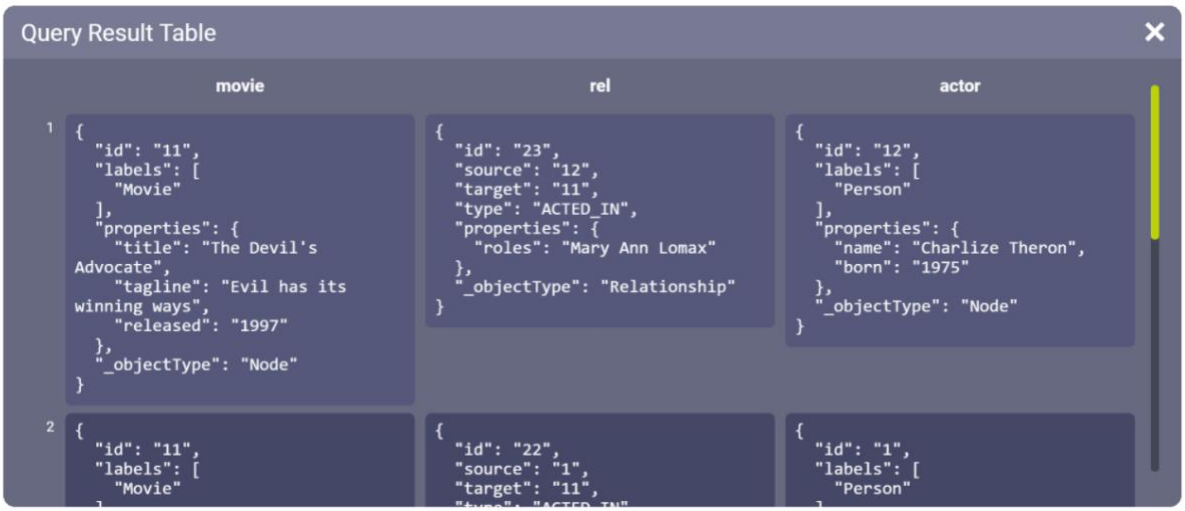


Rysunek 35. Okno Logger

6.4.14. Tabela z wynikiem zapytania

Powyższe okno umożliwia zapoznanie się z danymi zwróconymi przez bazę danych. Jest ono najbardziej przydatne, kiedy wynik zapytania nie może być zaprezentowany w postaci grafu. Przykładem może być sytuacja, kiedy na zwrócone dane składają się same krawędzie.

Okno przedstawiono na Rysunku 36, zawiera tabelę z możliwością przewijania, w której nagłówki kolumn są zmiennymi z wysłanego zapytania. Każdy wiersz zawiera wartości dla tych zmiennych w formacie JSON. Aby ułatwić nawigację po zwróconych rekordach wiersze są numerowane oraz kolory tła występują naprzemiennie w dwóch różnych odcieniach.



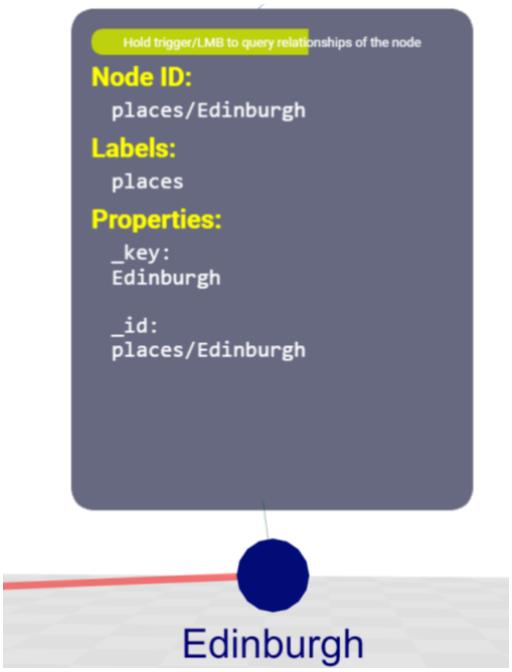
Rysunek 36. Okno Tabela z wynikiem zapytania

6.5. Interakcje z grafem

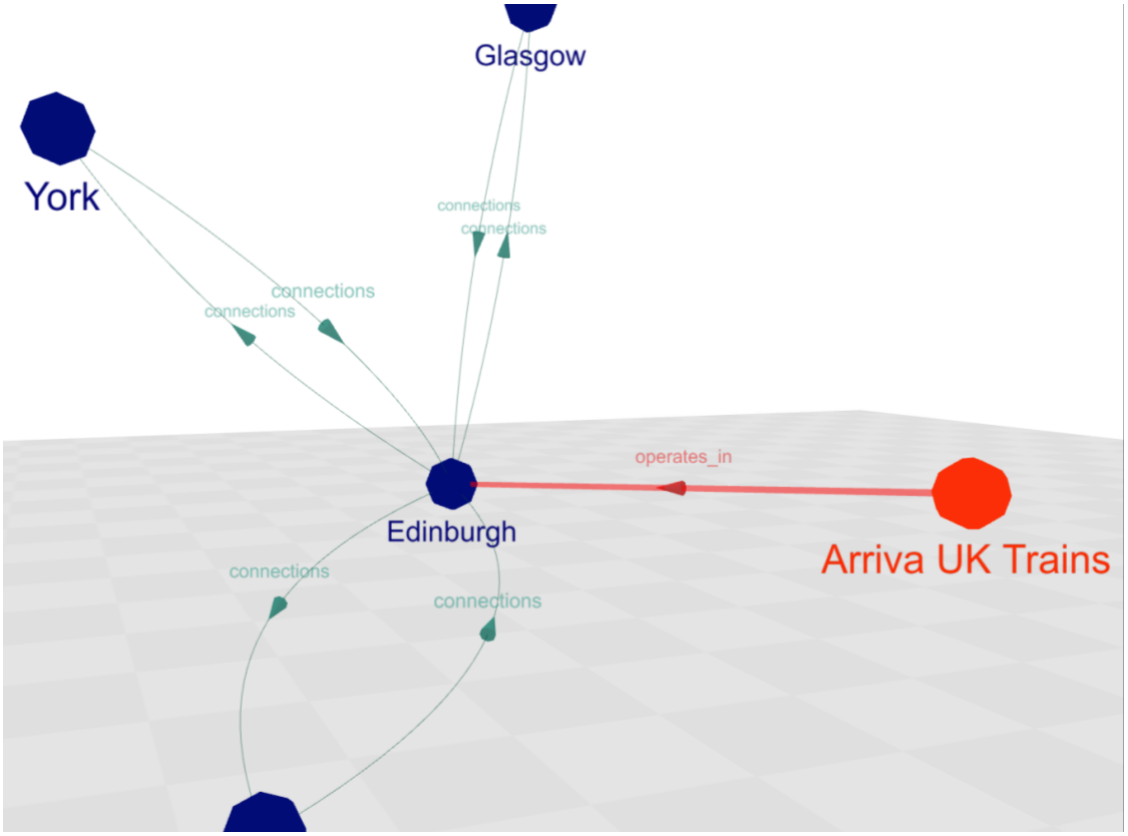
W rozwiązaniu użytkownik, poza dostosowaniem jak wygląda graf i w jaki sposób prezentuje dane, może wchodzić w interakcję z krawędziami i węzłami grafu.

Możliwe interakcje z węzłami to:

- wywołanie Okna szczegółów węzła po najechaniu na węzeł kursorem myszy lub promieniem wychodzącym z kontrolera
- wysłanie zapytania o węzeł, wszystkie jego relacje i węzły, z którymi te relacje posiada. W tym celu należy przytrzymać na węźle lewy przycisk myszy lub przycisk trigger na kontrolerze. Informację o tym jak długo należy trzymywać przycisk zapewnia pasek postępu znajdujący się w górnej części okna (Rysunek 37). Po jego wypełnieniu jest wysyłane zapytanie i generowany graf z wynikiem zapytania (Rysunek 38).
- przenoszenie węzłów – działa analogicznie do przenoszenia okien. Po najechaniu na węzeł należy przytrzymać przycisk grip. Węzeł będzie podążał za kontrolerem do chwili, aż przycisk zostanie zwolniony. W trybie Desktop nie ma możliwości przenoszenia węzłów.

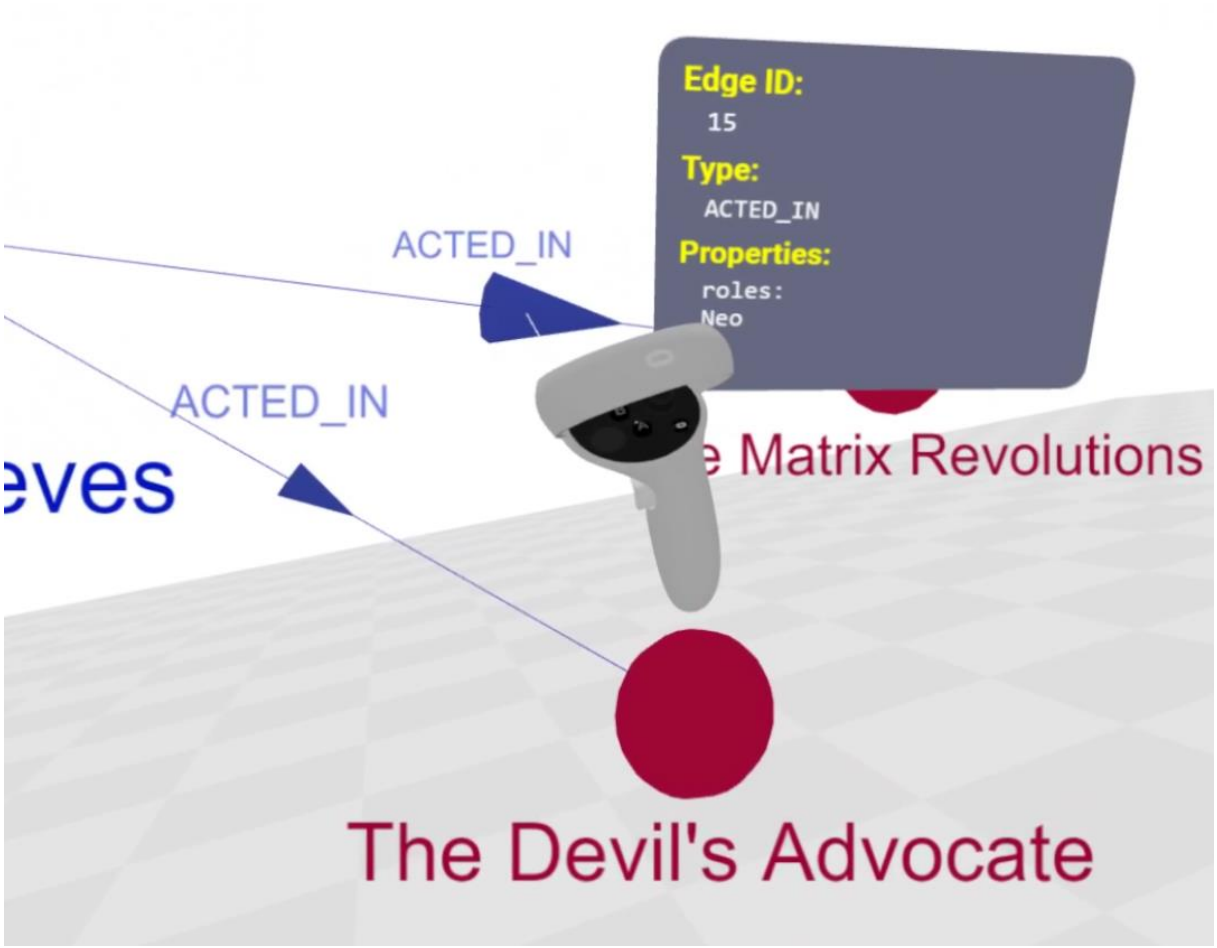


Rysunek 37. Okno szczegółów po najechaniu, kliknięciu i przytrzymaniu przycisku na węźle. Pasek postępu sygnalizuje czas pozostały do wykonania zapytania.



Rysunek 38. Wynik zapytania po przytrzymaniu przycisku na węźle Edinburgh

Możliwa interakcja z krawędziami to wywołanie Okna szczegółów krawędzi po najechaniu na stożek wskazujący kierunek krawędzi za pomocą myszy lub promienia wychodzącym z kontrolera. Z uwagi na małe rozmiary stożka, kiedy na niego wskażemy, jest on powiększany (patrz Rysunek 39). Ma to na celu ułatwienie podtrzymania wywołania okna.



Rysunek 39. Porównanie wielkości stożków wskazujących kierunek relacji. Stożek w górnej części zrzutu ekranu został powiększony po wskazaniu na niego.

7. Podsumowanie

W tym rozdziale wskazano wady i zalety opracowanego rozwiązania. Wskazane zostały również kierunku, w których aplikacja mogłaby być rozwijana. Na końcu zawarto podsumowanie pracy.

7.1. Wady i zalety rozwiązania

Prototyp w obecnym stanie spełnia postawione przed nim wymagania. Mimo posiadania szeregu zalet, można wskazać kilka wad cechujących opracowane rozwiązanie.

Wady:

- Użytkownik musi znać język zapytań używany w bazie danych. Jest to duże utrudnienie przy korzystaniu z aplikacji dla osób nieposiadających wiedzy technicznej.
- Aplikacja została napisana z myślą o jednym urządzeniu – Oculus Quest 2. Przy korzystaniu z innego urządzenia aplikacja może nie działać poprawnie.
- Układ klawiatury został dostosowany w celu szybszego wprowadzania zapytań. Niestety obecnie zawsze jest wyświetlana jedna wersja – ze słowami kluczowymi dla języka Cypher używanym przez bazy danych Neo4j.
- Aplikacja nie posiada możliwości zmiany języka. Wszystkie elementy interfejsu użytkownika oraz konwersja mowy na tekst są dostępne jedynie w języku angielskim.

Zalety:

- Aplikacja może być używana z poziomu urządzenia VR i komputera
- Zastosowanie technologii webowych sprawia, że z rozwiązania można korzystać niezależnie od platformy.
- Rozwiązanie może działać z rożnymi bazami danych. Architektura aplikacji została opracowana z myślą o łatwej implementacji obsługi kolejnych baz danych.
- Aplikacja pozwala na szybkie przełączanie obecnie używanej bazy danych z poziomu aplikacji klienckiej
- Istnieje możliwość dostosowania pozycji i skali grafu oraz przestrzeni, w której jest generowany. Użytkownik może zmieniać kolory, wielkość, kształt i podpisy znajdujące się pod elementami wchodzącymi w skład grafu.
- Zapewniono możliwość wejścia w interakcję z elementami grafu. Użytkownik po najechaniu na element jest w stanie wyświetlić szczegóły dotyczące elementu. Możliwe jest przenoszenie

węzłów w przestrzeni, a po przytrzymaniu na nich przycisku wykonywane jest zapytanie o ich relacje z innymi węzłami.

- Rozwiązanie pozwala na szybkie budowanie i wysyłanie zapytań z poziomu klienta, w językach używanych przez dane bazy danych. Jest to ułatwiane przez zaimplementowaną predykcję tekstu bazującą na słowach kluczowych języka zapytań i metadanych pochodzących z bazy danych. Dodatkowo użytkownik może korzystać z konwersji mowy na tekst.
- Możliwość przeglądania ostatnio wprowadzonych zapytań oraz zapisywania ich do późniejszego wykorzystania
- Możliwość poruszania się po scenie w kilku wariantach. Użytkownik może wybrać ten, który najbardziej mu odpowiada i nie budzi dyskomfortu w wirtualnej rzeczywistości.

7.2. Możliwości rozwoju aplikacji

Potencjalne kierunki rozwoju aplikacji, które warto rozważyć, to:

- Implementacja systemu logowania pozwoliłaby na korzystanie z aplikacji przez wielu użytkowników. Każdy z nich mógłby posiadać własne konto, łączyć się z innymi bazami danych, zapisywać interesujące go zapytania i ustawienia.
- Rozszerzenie poziomu możliwych interakcji z grafem, np. umożliwienie zmian właściwości dotyczących wyglądu wierzchołków i krawędzi po przytrzymaniu przycisku na elemencie czy usuwanie elementów z grafu.
- Funkcjonalność eksportowania grafu do pliku i importowania grafu z plików.
- Zaimplementowanie mechanizmu zmiany języka aplikacji oraz konwersji mowy na tekst. W chwili obecnej zarówno aplikacja jak i transkrypcja działają jedynie w języku angielskim.
- Umożliwienie modyfikacji danych w bazie z poziomu łatwego w obsłudze interfejsu użytkownika. Pozwoliłoby to na szybsze tworzenie i edytowanie danych. Dzięki temu nawet użytkownicy bez znajomości języka zapytań mogliby modyfikować zawartość bazy danych.
- Pozwolenie na łatwą modyfikację układów klawiatury wyświetlanej w aplikacji. Obecnie klawiatura jest dostępna tylko w jednej wersji i zawiera słowa kluczowe dla języka zapytań Neo4j. Warto zaimplementować mechanizm umożliwiający zmianę klawiszy i przełączanie się między układami klawiatury. Dzięki temu można byłoby zastąpić słowa kluczowe dla języka Cypher odpowiednikami dla języków zapytań innych baz danych.
- Możliwość filtrowania zawartości wygenerowanego grafu. Obecnie aby zawęzić wyświetlane wyniki należy zmodyfikować i ponownie wysłać zapytanie do bazy.

- Usprawnienie mechanizmu predykcji tekstu, na przykład przez dodanie do wyświetlanych sugestii wcześniej wprowadzonych słów czy wartości właściwości występujących w elementach z bazy danych
- Umożliwienie zmiany skali okien wyświetlanych użytkownikowi.
- Umożliwienie użytkownikom wspólnej pracy. Dzięki temu aplikacja pozwoliłaby na prezentację danych innym użytkownikom, łatwiejsze prowadzenie szkoleń czy wspólną analizę danych.
- Wsparcie większej ilości urządzeń VR. Aplikacja została napisana z myślą o Oculus Quest 2. Warto byłoby zapewnić wsparcie dla kilku najpopularniejszych urządzeń na rynku.

7.3. Podsumowanie pracy

W pracy przedstawione zostały istniejące na rynku rozwiązania do wizualizacji grafowych baz danych z wykorzystaniem wirtualnej rzeczywistości. Po przeanalizowaniu ich mocnych i słabych stron zaproponowano rozwiązanie, które stanowiłoby dla nich alternatywę. To co wyróżnia powstały prototyp to obsługa różnych baz danych, możliwość poruszania się po scenie w trzech wariantach oraz wspomaganie wprowadzania danych przez zastosowanie predykcji tekstu, konwersji mowy na tekst i okna klawiatury z układem dostosowanym tak, aby przyspieszyć budowanie zapytań.

Prototyp w obecnym stanie odpowiada na wymagania, które zostały przed nim postawione. Rozwinięcie go o możliwości wspomniane w 7.2 sprawiłoby, że aplikacja byłaby jedną z lepszych alternatyw dla istniejących już na rynku rozwiązań.

Bibliografia

[1] D. Sullivan, NoSQL. Przyjazny przewodnik, tłum. J. Hubisz, s. 308-310, ISBN: 978-83-283-2489-3, 2016.

[2] yEd Live, https://www.yworks.com/products/yed-live, dostęp: 13.06.2022.

[3] yFiles, https://www.yworks.com/products/yfiles, dostęp: 13.06.2022.

[4] Data Explorer for Neo4j, https://www.yworks.com/neo4j-explorer, dostęp: 13.06.2022.

[5] GraphXR, https://www.kineviz.com, dostęp: 13.06.2022.

[6] AviarGraph, https://aviar.tech, dostęp: 13.06.2022.

[7] Noda, https://noda.io, dostęp: 13.06.2022.

[8] HTML Standard, https://html.spec.whatwg.org/multipage/introduction.html, dostęp: 13.06.2022.

[9] https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS, dostęp: 13.06.2022.

[10] https://w3techs.com/technologies/details/cp-javascript, dostęp: 13.06.2022.

[11] Webpack, https://webpack.js.org, dostęp: 02.06.2022.

[12] Babel, https://babeljs.io, dostęp: 02.06.2022.

[13] Biblioteka three-forcegraph, https://github.com/vasturiano/three-forcegraph, dostęp: 02.06.2022.

[14] Biblioteka Notyf, https://carlosroso.com/notyf, dostęp: 02.06.2022.

[15] https://wicg.github.io/speech-api/#introduction, dostęp: 13.06.2022.

[16] https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API#browser_compatibility, dostęp: 13.06.2022.

[17] https://www.khronos.org/webgl/wiki/Getting_Started, dostęp: 13.06.2022.

[18] Biblioteka Three.js, https://threejs.org, dostęp: 02.06.2022.

[19] WebXR, https://immersiveweb.dev, dostęp: 02.06.2022.

[20] https://github.com/immersive-web/webxr/blob/master/explainer.md#goals, dostęp: 13.06.2022.

[21] B. Dayley, Node.js, MongoDB, and AngularJS Web Development, s. 2-3, ISBN: 978-0-321-99578-0, Addison-Wesley, 2014.

[22] Express.js, https://expressjs.com, dostęp: 02.06.2022.

[23] MongoDB, https://www.mongodb.com, dostęp: 02.06.2022.

[24] Mongoose, https://mongoosejs.com, dostęp: 02.06.2022.

[25] Neo4j, https://neo4j.com, dostęp: 02.06.2022.

[26] ArangoDB, https://www.arangodb.com, dostęp: 02.06.2022.

[27] M. Malińska, K. Żużewicz, J. Bugajska i A. Grabowski, Subiektywne odczucia wskazujące na występowanie choroby symulatorowej i zmęczenie po ekspozycji na rzeczywistość wirtualną, 2014.

[28] E. Langbehn, P. Lubos i F. Steinicke, Evaluation of Locomotion Techniques for Room-Scale VR: Joystick, Teleportation, and Redirected Walking, 2018.

[29] S. Kongsvik, Text Input Techniques in Virtual Reality Environments - An empirical comparison. Master thesis, University of Oslo, 2018.

Wykaz rysunków

Wykaz tabel

Wykaz listingów