

Dokumentacja projektu „Świat Drzwi” wykonanego w ramach przedmiotu: Przetwarzanie danych w chmurach obliczeniowych.

Michał Domin

1. Opis projektu koncepcja i założenia.

Tematem projektu jest „Świat Drzwi” (posiadający angielską nazwę: „World of Doors”).

Jest to społecznościowa aplikacja internetowa, pozwalająca na poruszanie się po świecie stworzonym z pokoi połączonych drzwiami. Aplikacja obsługuje rejestrację i logowanie użytkownika (użytkownik ma przypisane w jakim pokoju się znajduje). Możliwe akcje zalogowanego użytkownika:

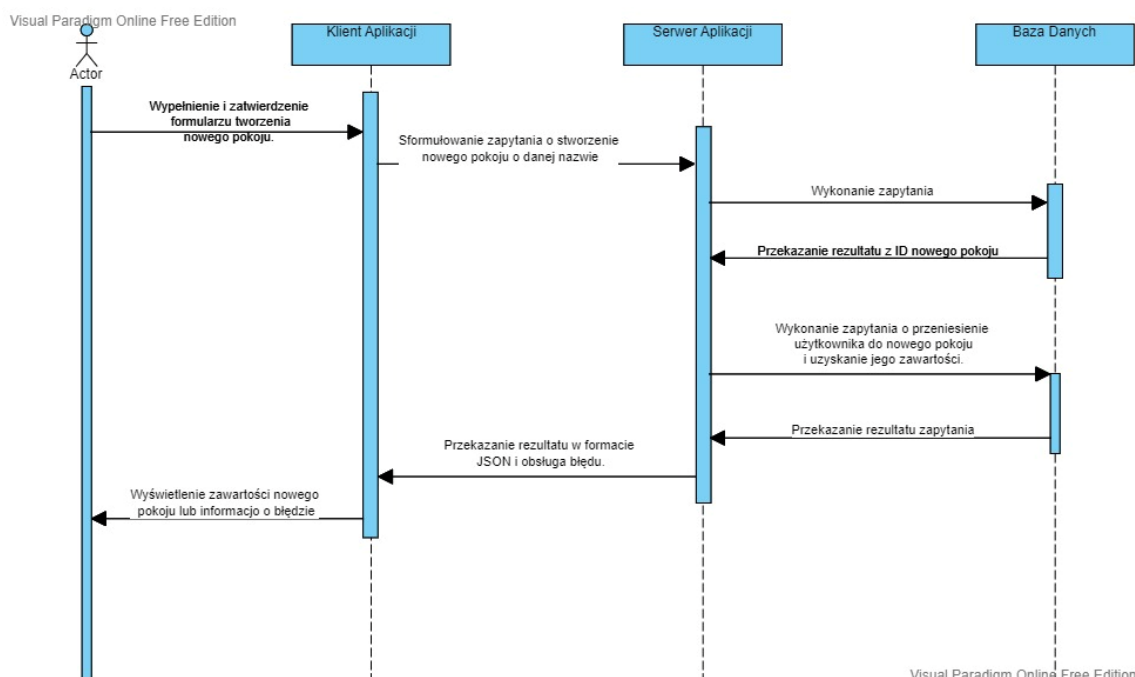
- Przejsie do wybranego pokoju poprzez kliknięcie na drzwi.

- Stworzenie nowego pokoju o wybranej nazwie (użytkownik zostanie przeniesiony do stworzonego pokoju).

- Stworzenie wiadomości w pokoju w którym się znajduje.

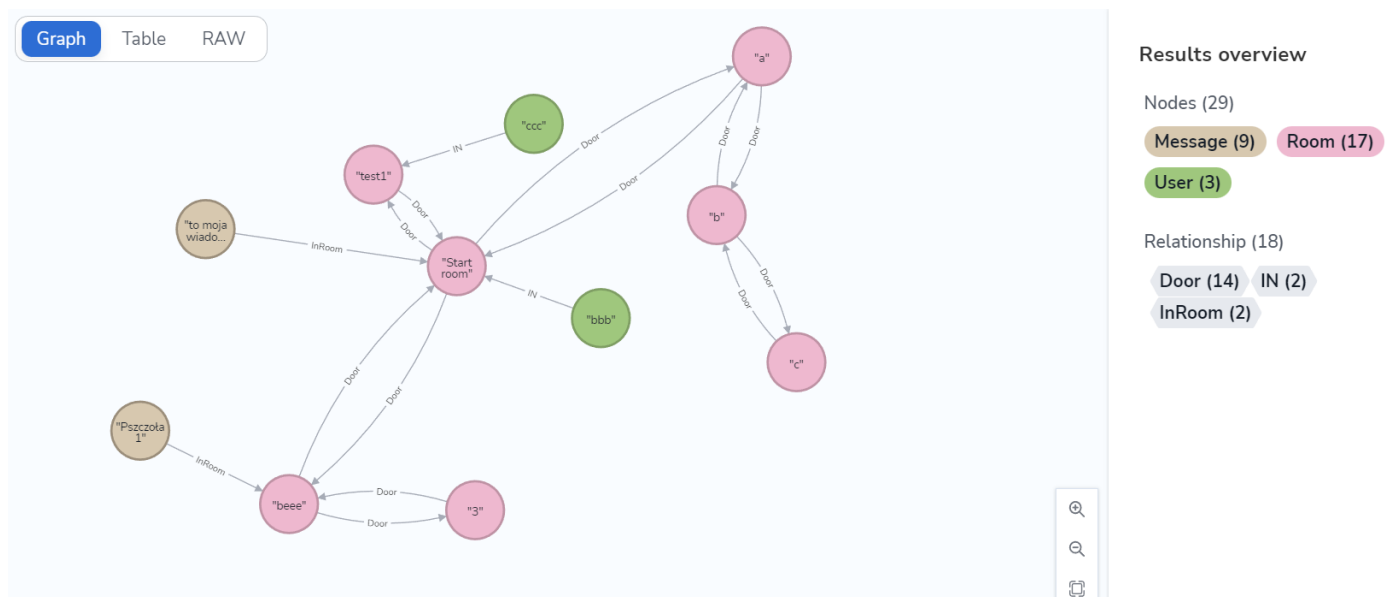
2. Projekt przepływu danych.

Poniższy diagram przedstawia przepływ danych w projekcie na przykładzie stworzenia nowego pokoju.



3. Reprezentacja grafowa.

Poniższy zrzut ekranu z przeglądarki danych bazy Neo4j prezentuje wycinek bazy danych prezentujący struktury danych w formie grafowej, jest to struktura grafu skierowanego.



4. Zastosowane Narzędzia i technologie.

Aplikacja została podzielona na dwie części frontendową odpowiedzialną głównie za wyświetlanie zawartości Pokoju i jej obsługę. Oraz backendową będącą serwerem odpowiedzialnym za obsługę zapytań i logikę aplikacji. Obie części zostały napisane w języku TypeScript z użyciem bundlera Webpack. Server stworzono w node.js z użyciem expres.js w którym obsłużono żądania POST/GET. Baza danych przechowywana jest w chmurowej usłudze Neo4j AuraDB.

5. Opis działania aplikacji.

Na początku gdy nie jest się zalogowanym aplikacja wyświetla stronę prosząc o zalogowanie lub rejestrację:

[Login](#)
[Register](#)

W przypadku Rejestracji wypełniamy następujący formularz:

Username:

Username

Password:

Password

register

[Back](#)

Po zatwierdzeniu formularzu jeśli nie istnieje użytkownik o wybranej nazwie, zostanie utworzony taki użytkownik i przypisany do pokoju startowego oraz zostanie się przeniesionym do strony przedstawiającej pokój startowy. W przypadku błędu lub wybrania istniejącej nazwy zostanie wyświetlony komunikat o błędzie.

W przypadku Logowania wypełniamy następujący formularz:

Username:

Username

Password:

Password

login

[Back](#)





Po zatwierdzeniu formularzu jeśli istnieje użytkownik o wybranej nazwie, oraz wpisane hasło jest poprawne zostanie się przeniesionym do strony przedstawiającej pokój w którym znajduje się użytkownik. W przypadku błędu zostanie wyświetlony komunika.

Strona przedstawiająca pokój:

Start room[Logout](#)

Name:

Message:



a**beee****test2****test1**

wwwww2

www

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

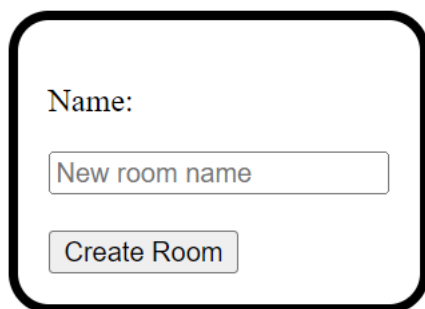
saaaaaaaaaaaaaaaaaaaaaaaaafisd
nfknfsfnldkfnddddddndddd
dddddnddddnddddnddddnddddvvv
vvvvvvvvvvvvvvvvvvvvvv
fdgfsfhgsrdg dsfg fdg sergsrg srth dgh
drth dth dy jdtjdrths trg

W lewym górnym rogu znajduje się nazwa pokoju w którym użytkownik się aktualnie znajduje. A w prawym możliwość wylogowania.

W przypadku kliknięcia na drzwi użytkownik zostanie przeniesiony do pokoju o nazwie znajdującej się pod drzwiami (użytkownik może zostać przeniesiony tylko do pokoju sąsiadującego z pokojem w którym się znajduje, w bazie danych krawędź wskazująca na pokój w którym jest użytkownik zostanie przeniesiona aby wskazywać na wybrany pokój).

Poniżej drzwi znajdują się wiadomości znajdujące się danym pokoju.

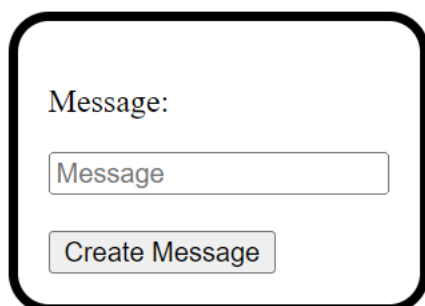
W przypadku wypełnienia i zatwierdzenia formularza stworzenia pokoju:



Formularz stworzenia pokoju. Zawiera on etykietę "Name:", pole tekstowe z placeholderem "New room name" oraz przycisk "Create Room".

Zostanie utworzony nowy pokój o wybranej nazwie, dostępny dla wszystkich użytkowników. Nowy pokój zostaje sąsiadem pokoju z którego został utworzony oraz zostają utworzone drzwi między nimi. Użytkownik zostaje przeniesiony do nowego pokoju.

W przypadku wypełnienia i zatwierdzenia formularza stworzenia Wiadomości:



Formularz stworzenia wiadomości. Zawiera on etykietę "Message:", pole tekstowe z placeholderem "Message" oraz przycisk "Create Message".

Zostanie utworzona nowa wiadomość o wybranej zawartości, dostępna dla wszystkich użytkowników. Nowa wiadomość zostaje przypisana do pokoju w którym została utworzona.

6. Dokumentacja kodu:

Endpointy aplikacji:

-(GET) - Strona domowa (w przypadku nie bycia zalogowanym zostaje się przeniesionym do /auth)

-/auth(GET) – Wybór logowania lub rejestracji

-/login(GET) – Formularz logowania

-/register(GET) – Formularz rejestracji

-/auth/login(POST) – Logowanie

-/auth/register(POST) – Rejestracja

-/auth/logout(POST) – Wylogowanie

- /game/create_room(POST) – Stworzenie nowego pokoju
- /game/create_message(POST) – Stworzenie nowej wiadomości
- /game/move_to_room(POST) – Przeniesienie do wybranego sąsiedniego pokoju
- /game/get_neighbor_rooms(POST) – Pobranie wszystkich sąsiednich pokoi

Cały kod znajduje się w folderze src: w folderze client znajduje się część frontendowa aplikacji, w folderze server znajduje się część backendowa aplikacji:

- plik server.ts - kod odpowiedzialny za inicjalizację aplikacji
- plik AuthRouter.ts – kod odpowiedzialny za endpointy autentykacji i autentykację użytkownika
- plik GameReouet.ts – kod odpowiedzialny za endpointy związane z logiką aplikacji
- plik DBController.ts – kod odpowiedzialny za połączenie z bazą danych.

w folderze templates znajdują się template'y używane w aplikacji, w folderze types znajdują się typescriptowe typy używane w obu częściach aplikacji.

7. Wdrożenie aplikacji.

Do aplikacji został przygotowany Dockerfile umożliwiający stworzenie kontenera z aplikacją. Aby stworzyć docker image należy użyć „docker build . -t *nazwa*” a następnie uruchomić docker image za pomocą: „docker run -p 3000:3000 -d *nazwa*”, po uruchomieniu można się już połączyć do aplikacji poprzez localhost na porcie 3000.

Repozytorium z kodem aplikacji: <https://github.com/mdmkl6/Clouds-Project>

8. Możliwości rozwoju.

Aplikacja została przystosowana do łatwego rozwoju, można bardzo łatwo dodać kolejne funkcjonalności obok, dodawania nowego pokoju, wiadomości. Przykładowe możliwe rozwinięcie aplikacji: wyszukiwanie najkrótszej drogi do wybranego pokoju, tworzenie drzwi między istniejącymi pokojami, poprawienie wyglądu aplikacji.