Zaawansowane Techniki Internetowe - WFiIS AGH

Dokumentacja projektu

Aleksandra Śliwska, 09.06.25r.

1 Opis

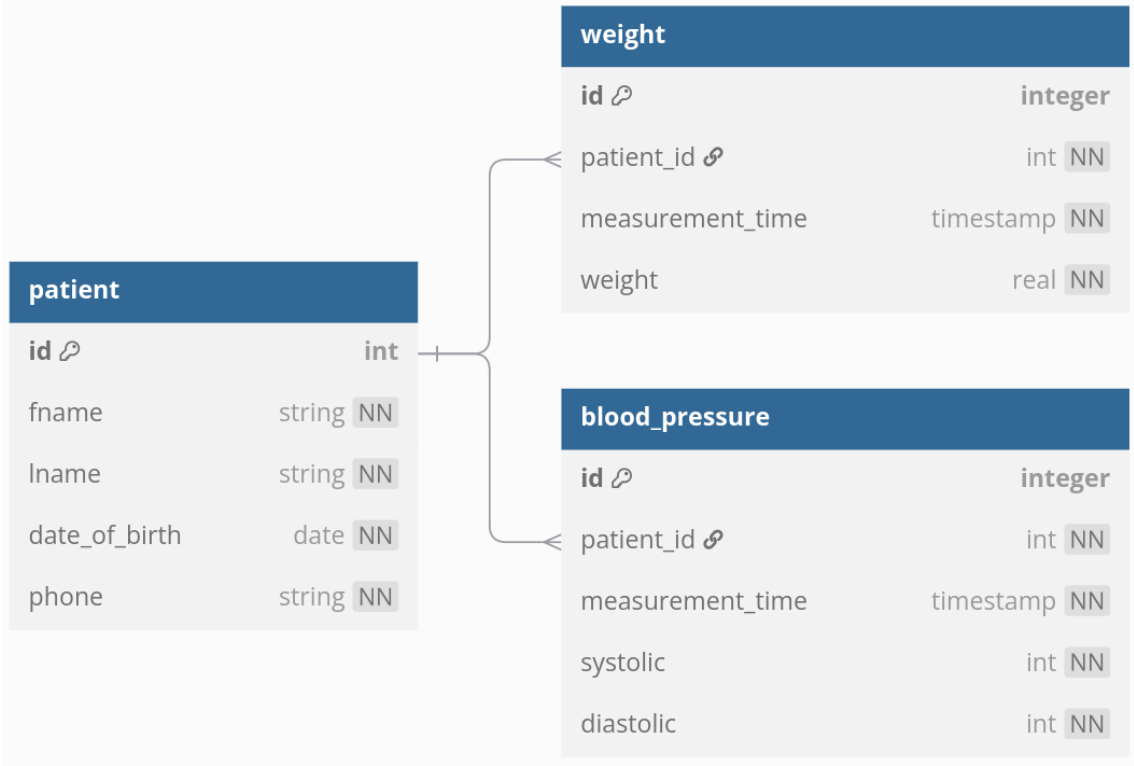
Aplikacja opisywana przez ten dokument to strona internetowa dla lekarzy pozwalająca na zapisywanie danych o pacjentach, ich ciśnienia i wagi oraz wyświetlanie i edycję tych informacji na wykresach.

Aplikacje klienta i serwera są skonteneryzowane przy pomocy technologii Docker, a do ich prostego uruchamiania wykorzystano Docker Compose.

Kod można znaleźć pod adresem: `https://github.com/aSliwska/zti-projekt`.

Działający projekt znajduje się w sieci wydziałowej pod adresem: `http://172.20.41.39:3001`.

2 Baza danych



Rysunek 1: Schemat bazy danych.

Za bazę danych aplikacji służy wydziałowy PostgreSQL. Trzy pokazane na powyższym obrazku tabele są trzymane są w schemacie „zti_proj”. Baza danych jest połączona do serwera przez JDBC.

3 Serwer

Serwer korzysta z technologii:

- Java,
- Spring Boot,
- Spring Data JPA,
- GraphQL Java - jako API dla front-endu,
- Lombok - do generacji getterów, setterów i loggerów,
- Spring AOP - do logowania zapytań do API.

Poszczególne pliki w katalogu „resources" spełniają odpowiednie funkcje:

- data.sql i schema.sql - tworzą i wypełniają schemat w bazie danych przykładowymi danymi,
- graphql/schema.graphqls - definiuje typy danych i nazwy/wejścia/wyjścia zapytań do API; dane stąd są mapowane bezpośrednio do metod i rekordów w PatientController.java,
- application.properties - definiuje dane łączeniowe do bazy danych.

Pliki z kodem źródłowym natomiast służą do:

- ServerApplication.java - startuje aplikację Spring Boot,
- Logger.java - wykorzystuje technologię Spring AOP, aby w trakcie działania programu zapisywać w logach spring-bootowych informacje o wywołaniu każdej publicznej metody w PatientController.java; zapisywane są nazwa funkcji, argumenty wejścia i wyjście,
- data/BloodPressure.java, data/Weight.java, data/Patient.java - klasy mapujące obiekty DTO do rekordów w tabelach w bazie danych przy użyciu Spring Data JPA; Lombok generuje dla nich gettery i settery; ID mapowane jest zgodnie z generatorami sekwencji w PostgreSQL; także tutaj definiowane jest zachowanie kaskadowego usuwania (w pliku Patient.java, nie w samej tabeli w bazie danych),
- control/BloodPressureRepository.java, control/WeightRepository.java, control/PatientRepository.java - repozytoria GraphQL + JPA automatycznie generujące odpowienie metody do manipulacji danymi w bazie danych po stronie serwera,
- control/PatientController.java - zawiera metody, które implementują logikę endpointów zdefiniowanych w pliku „resources/graphql/schema.graphqls".

Jedyny endpoint API, jaki udostępnia aplikacja to „/graphql". Przyjmuje on string z formą zapytania (definiującą jego wejście, nazwę funkcji i żądane elementy wyjścia) oraz string z obiektem typu JSON z danymi do zapytania. Przykładowe zapytania opisane są w następnej sekcji.

Zdefiniowana funkcjonalność w pliku „PatientController.java" pozwala na:

- patient - zwrócenie danych pacjenta o danym ID wraz z jego pomiarami medycznymi,
- patients - zwrócenie danych o wszystkich pacjentach wraz z ich pomiarami medycznymi,
- createPatient - stworzenie pacjenta o odpowiednich danych,
- updatePatient - ustawienie danych pacjenta o danym ID na podane nowe,
- removePatient - usunięcie pacjenta wraz ze wszystkimi jego pomiarami,
- createWeight - stworzenie nowego pomiaru wagi i przypisanie go do pacjenta z podanym ID,
- updateWeight - zmianę pomiaru wagi o danym ID,

- removeWeight - usunięcie pomiaru wagi o danym ID,
- createBloodPressure - stworzenie nowego pomiaru ciśnienia i przypisanie go do pacjenta z podanym ID,
- updateBloodPressure - zmianę pomiaru ciśnienia o danym ID,
- removeBloodPressure - usunięcie pomiaru ciśnienia o danym ID.

Plik „Dockerfile" w korzeniu plików serwerowych pozwala na zbudowanie obrazu serwera.

4 Klient

Klient korzysta z technologii:

- JavaScript,
- ReactJS,
- Next.js - middleware zajmujące się routingiem i renderujące UI,
- Mantine - biblioteka komponentów UI,
- Jotai - pozwala na definiowanie globalnych zmiennych (zastępuje Context w ReactJS),
- ApolloClient - do wysyłania zapytań GraphQL do serwera.

W korzeniu plików UI:

- Dockerfile - pozwala na zbudowanie obrazu klienta,
- postcss.config.cjs - konfiguruje CSS Mantine,
- next.config.mjs - konfiguruje output plików przy kompilacji oraz reverse proxy do serwera (przeglądarka wysyła zapytania do http://localhost:3001/api, które są przekierowywane do kontenera z serwerem).

Pliki z kodem źródłowym natomiast pełnią funcje:

- store/store.js - przechowuje atomy z biblioteki Jotai (zmienne globalne) oraz formaty zapytań do API GraphQL,
- components - komponenty UI wykorzystywane w aplikacji
  - header/MainHeader.js - nagłówek do nawigacji po stronie,
  - info_card/InfoCard.js - karta z informacjami o pacjencie wyświetlająca się z prawej strony ekranu,
  - listWithForm/ListWithForm.js - wrapper na tabelę z informacjami oraz formy do jej edycji, przekazuje im odpowienie zmienne i pozwala na komunikację między nimi,
  - listWithForm/list/EditableList.js - wyświetla i steruje tabelą danych, pozwala na usuwanie rzędów, włączenie trybu edycji i ustawianie odpowiednich danych do edycji,
  - listWithForm/form/AddForm.js - definiuje guzik do otwierania formy do dodawania, formę do dodawania, formę do edycji,
- app - struktura strony (definicja URI w przeglądarce)
  - layout.js, Providers.js - kontener na całą stronę, konfiguracja, umiejscowienie nagłówka nawigacyjnego,
  - (main)/layout.js - sterowanie wielkością marginesów w zależności od wielkości okna oraz ustawienie karty z informacjami o wyświetlanym pacjencie,

- (main)/patients/page.js - definicja zachowania aplikacji przy wciśnięciu przycisków tworzących/edytujących/usuwających (wysyłanie zapytań i danych do API, odświeżanie danych), zdefiniowanie walidacji wpisywanych danych i nazw kolumn tabel, formatowanie dat,
- (main)/measurements/page.js - to samo co „(main)/patients/page.js", dodatkowo renderuje wykresy.

Przykładowy przepływ danych z UI do bazy danych:

1. UI wybiera string z definicją obsługi endpointu z pliku „store/store.js" - na przykład „CREATE_WEIGHT".

```
mutation CreateWeight($patient_id: Int!, $weightInput: WeightInput!) {
    createWeight(patient_id: $patient_id, weightInput: $weightInput) {
        id
    }
}
```

2. UI kompiluje obiekt javascriptowy z danymi o nowym pacjencie wprowadzonymi w formularzu przez użytkownika.

```
{
    patient_id: 1,
    weightInput: {
        measurement_time: "23-04-2025",
        weight: 66.9
    }
}
```

3. Klient wysyła oba te obiekty do middleware UI (http://localhost:3001/api), które przekierowuje zapytanie (reverse proxy) do kontenera z serwerem i endpointu /graphql.
4. Serwer, zgodnie z plikiem „resources/graphql/schema.graphqls" rozpoczyna wykonywanie zmapowanej funkcji „createWeight" w „control/PatientController.java".
5. Spring Data JPA za pomocą DTO i repozytoriów tworzy w pamięci serwera nowy zapis wagi, przypisuje go do użytkownika i zapisuje nowy rekord w bazie danych.
6. Serwer zwraca do UI ID nowego wpisu wagi.

5 Kompilacja

Przed kompilacją należy w pliku „server/src/main/resources/application.properties" zmienić dane dostępowe do bazy danych na takie, jakie pozwolą nawiązać połączenie z wydziałową bazą danych.

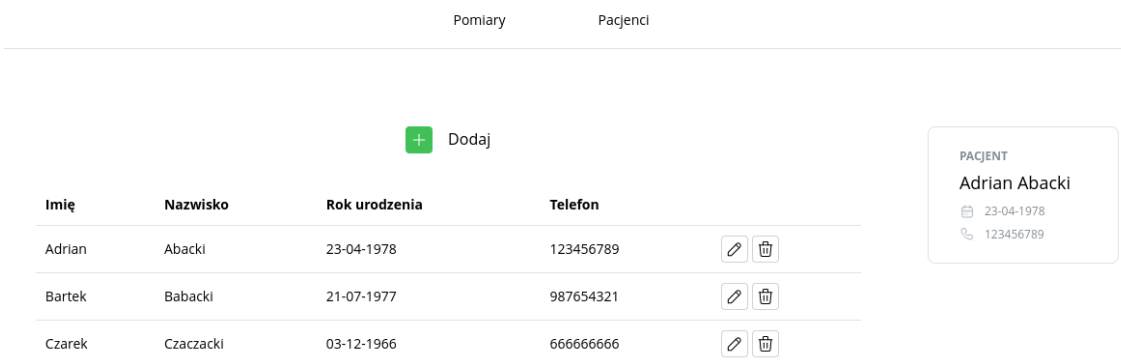
Następnie należy wykonać poniższe komendy z poziomu z plikiem „docker-compose.yaml":

```
docker network create zti-project
docker compose up
```

Na stronie `http://localhost:3001/` pojawi się działająca aplikacja.

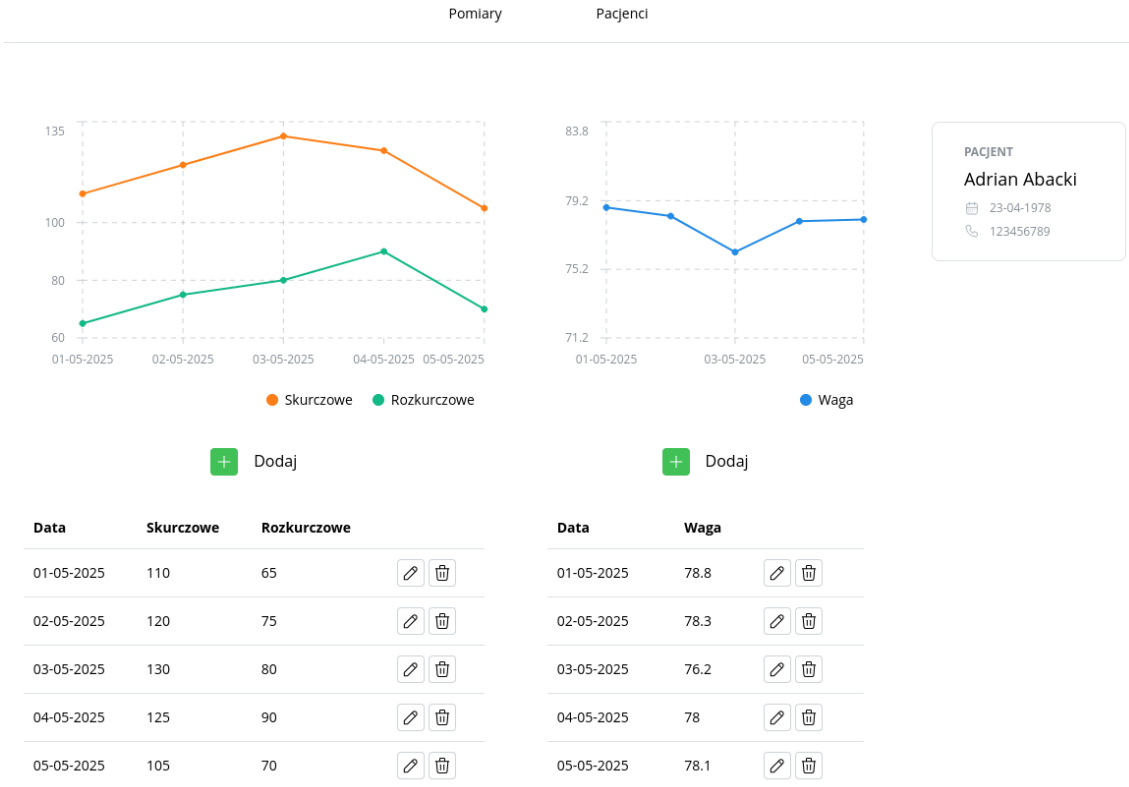
6 Podręcznik użytkownika

Przy pierwszym wejściu na stronę należy kliknąć na zakładkę „Pacjenci" i kliknąć na rząd z wybranym pacjentem. Jego dane pojawią się po prawej stronie ekranu.



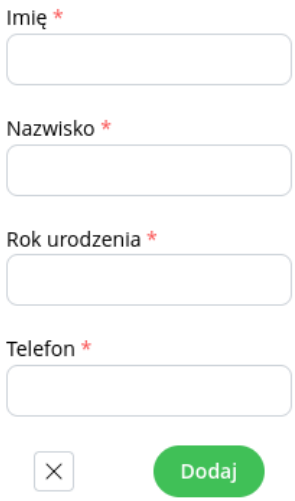
Rysunek 2: Strona z listą pacjentów.

Następnie można udać się na stronę „Pomiary”, aby wyświetlić wyniki badań pacjenta.



Rysunek 3: Strona z listą wyników badań.

Dane w każdej tabeli można edytować. Aby dodać nowy rekord, należy wcisnąć nad odpowiednią tabelą przycisk „+”, wypełnić formę, która się w jego miejscu pojawi i potwierdzić przyciskiem "Dodaj". Nowe dane od razu pojawią się w tabeli.



Rysunek 4: Forma do dodawania pacjenta do bazy danych.

Dodany rekord można edytować po naciśnięciu przycisku z ikonką ołówka w odpowiadającym mu rzędzie w tabeli. Wtedy w miejscu przycisku „+” pojawi się taka sama forma jak do dodawania, ale wypełniona danymi z tego rzędu. Po zmienieniu tych danych należy potwierdzić edycję przyciskiem „Zapisz”.

Każdy rekord można także usunąć przyciskiem z ikonką kosza na śmieci.