Bazy Danych
Projekt Karta Pacjenta

8 stycznia 2020

Spis treści

1 Autorzy

Krzysztof Czarnecki - implementacja niewidocznej dla użytkownika części programu, która posiada dostęp do wymaganych zasobów (ang. *backend*)
Błażej Czekała - implementacja testów obciążeniowych, z użyciem wielowątkowości
Patryk Wenz - implementacja wyglądu i zachowania strony (ang. *frontend*)
Hubert Braun - implementacja bazy danych i jej obsługi

2 Opis projektu

Projekt zakładał stworzenie aplikacji umożliwiającej przechowywanie danych pacjentów w serwisie bazodanowym. Aplikacja miała być swoistą kartą pacjenta przechowującą informacje dotyczące chorób przebytych przez pacjenta. Ma ona umożliwiać bezpieczne przechowywanie danych wrażliwych, takich jak pesel, numer telefonu itd. Umożliwia ona także prezentację tych danych oraz ich eksport (także w postaci anonimowej - bez danych osobowych - posiadających tylko informację o przebytej chorobie, nie o pacjencie).

Aplikacja powstała przy użyciu języków: Java (*backend*) oraz Angular (TypeScript) *frontend*. Serwis postawiony jest na darmowej domenie `http://trunk-kartapacje herokuapp.com/`. Gorąco zachęcamy do zapoznania się z działaniem aplikacji.

Dzięki serwisowi Heroku możliwe było darmowe opublikowanie witryny w internecie. Nawet w darmowej wersji serwis ten zapewnia usługi związane z CI *continuous integration*. Efektem tego, jest fakt, że po każdej aktualizacji zdalnego repozystorium *Git* serwis automatycznie przebudowuje się.

2.1 Testowanie aplikacji

Do dogłębnego przetestowania aplikacji wymagane jest, aby skorzystać z konta administratora - konta różnych użytkowników posiadają różne uprawnienia. Administrator ma dostęp do wszystkich możliwych miejsc na stronie. Podczas testowania aplikacji proszę używać konta:

login: admin
hasło: admin

3 Wykorzystane technologie

- Spring Boot
- Spring Security
- PostgreSQL
- Java
- Angular

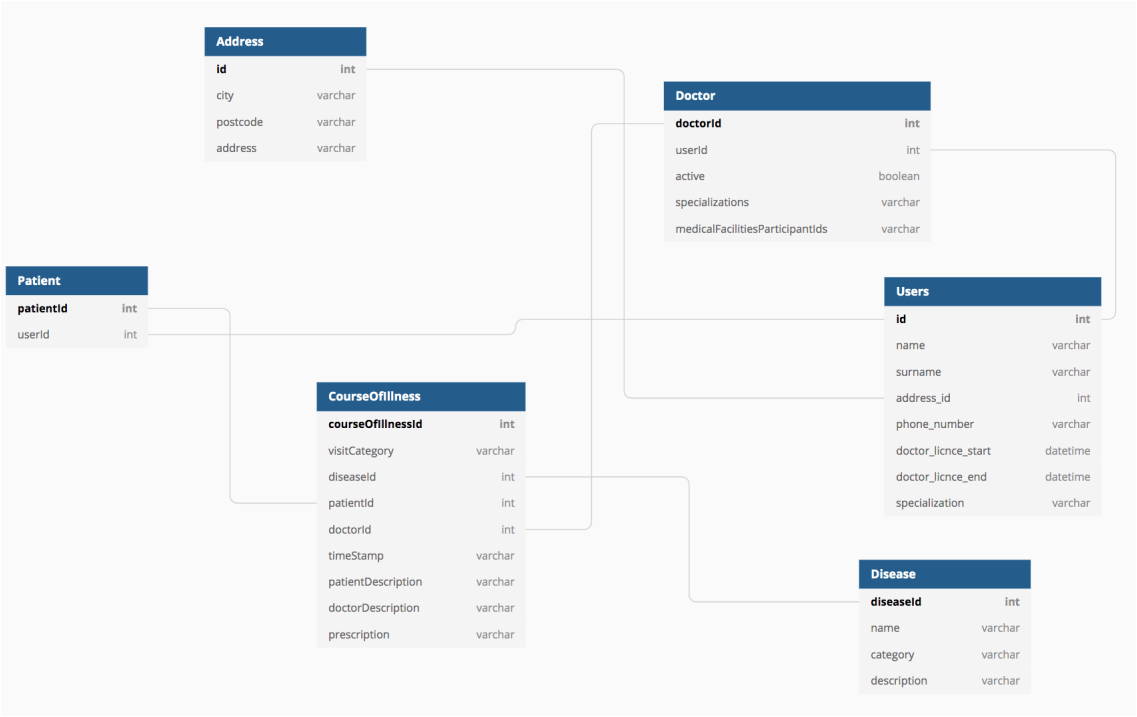
4 Implementacja

4.1 Kontrola wersji

Do pracy zespołowej wykorzystaliśmy znane i lubiane narzędzie *GitHub*. Umożliwiło nam to sprawne dzielenie się zmianami w kodzie, zarządzanie i wersjonowanie go.

4.2 Baza danych

Poniżej zamieszczony jest schemat ER bazy danych, wykorzystywanej w aplikacji karty pacjenta.



Rysunek 1: Diagram ER bazy danych

4.3 Aplikacja - backend

4.3.1 Endpointy

Dzięki ogromnej popularności aplikacji internetowych opartych na języku *Java* oraz *framework Spring* możliwe było szybkie wygenerowanie dokumentacji *Open API*. Pod linkiem dostępny jest spis wszystkich dostępnych w serwisie endpointów. Wejście w ten link będzie wymagało podania loginu i hasła (dostępnego tutaj: 2.1).

4.3.2 Dlaczego REST?

Zalety REST API:

- Bezstanowość klienta - serwer nie ma potrzeby zapamiętywania wcześniejszego statnu, ponieważ zapytania HTTP zawierają wszystkie potrzebne informacje,
- Łatwość manipulowania obiektami z poziomu URL,
- Czytelność wykonywanych działań ze względu na Endpointy,
- Szybsze przetwarzanie informacji zwrotnej - uzycie JSON.

4.3.3 Zabezpieczenie danych - API

Większość *endpointów* dostępnych w serwisie zapezpieczone jest przy użyciu metody *Basic Auth*. Bez podania loginu i hasła niemożliwy jest dostęp do serwisu. Jedyne dostępne bez konieczności autoryzacji endpointy to te dotyczące logowania i rejestracji.

4.3.4 Zabezpieczenie danych - baza danych

Do zabezpieczenia danych skorzystaliśmy z hashowania danych, przy użyciu klucza symetrycznego wbudowanego w API Springa.

4.3.5 Ograniczenia

W trakcie implementacji kolejnych funkcjonalności musieliśmy zmierzyć się z ograniczeniami serwera Heroku. Obsługa requestów jest jednowątkowa, stąd testowanie jego wydajności jest mocno ograniczone. Skorzystanie z darmowej domeny sprawia, że funkcjonowanie strony jest po prostu wolne.

4.3.6 Możliwości dotyczące rozwoju - przyspieszenie

Przyspieszenie może zostać uzyskane np. poprzez zastosowanie architektury mikroserwisowej, tak by każda atomowa operacja mogła zostać wykonywana niezależnie. Zapewniłoby to dużą skalowalność systemU i pod dużym obciążeniem przełożyłoby się to na przyspieszenie.

4.4 Aplikacja - frontend

4.4.1 Działanie warstwy wizualnej

Warstwa wizualna oparta jest na koncepcie reakcyjnego obsługiwania zdarzeń użytkownika końcowego. Zamiast przeładowania strony po wykonaniu akcji związanej z zdarzeniem wejścia aplikacji widoku preferowane jest odświeżenie jej części.

4.4.2 Prostota implementacji i wieloplatformowość

Frontend i Backend zostały oddzielone od siebie podczas implementacji - osobne repozytoria git. W części Frontendu skorzystano z Bootsrapa, a obsługę akcji wykonuje inny framework TypeScryptowy - Angular. Rozdzielnie projektu w taki sposób, umożliwia oddzielną pracę nad tymi częściami.

4.4.3 Rola warstwy wizualnej aplikcji

Frontend jest kanałem komunikacji między serwerem a klientem. Taka zależność zapewnia szybkie wykonywanie akcji zadanych przez użytkownika bez znajomości wewnętrznej implementacji serwisu. Podczas tworzenia "Karty Pacjentaśtarano się, aby wystrój był maksymalnie przejrzysty, wszystkie funkcjonalności opisane i rozmieszczone w jednoznaczny sposób, a komunikacja między serwerem a klientem była jak najszybsza.

4.5 Testy obciążeniowe

4.5.1 Testowane zagadnienia

Z uwagi na to, że dostęp do aplikacji powinno mieć w tym samym czasie dziesiątki tysięcy użytkowników (pacjenci i lekarze jednocześnie). Stąd bardzo ważną rzeczą jest obsługa endpointów, które dostarczają odpowieni zbiór danych.
Zaimplementowane zostały testy obciążeniowe mające na celu sprawdzenie wydajności naszego serwisu. Skorzystaliśmy z RestTemplate, czyli biblioteki dostępnej we frameworku Spring.
Badane zagadnienia:

1. Obsługa requestów http, metod GET, POST, PUT, DELETE,
2. Weryfikacja statusów HTTP,
3. Czas odpowiedzi serwera,

4. Odporność na żądania wielu (dziesiątki tysięcy) użytkowników.

4.5.2 Wielowątkowość

Wielowątkowość została zaimplementowana po to, aby możliwie najlepiej oddać realny sposób użytkowania. W jednym momencie tysiące użytkowników może zażądać tego samego zbioru danych. Każde z urządzeń powinno zostać poprawnie obsłużone.

4.5.3 Szybkość działania aplikacji

4.5.4 Rola testów w rozwoju aplikacji

Testy pomagają zweryfikować, czy postawione endpointy spełniają swoją rolę.

Literatura

[Walls(2015)] Craig Walls. *Spring in Action, Fourth Edition.* 2015. ISBN 9788328308497.

[Forta(2012)] Ben Forta. *Sams Teach Yourself.* 2012. ISBN 0672336073.

[Krystyna Balińska(2004)] Krzysztof T. Zwierzyński Krystyna Balińska. *Projektowanie algorytmów grafowych.* 2004. ISBN 8371435487.

[Bykowski(2019)] Przemysław Bykowski. Spring boot od podstaw. `https://www.youtube.com/playlist?list=PLUtcRmGoaP27ypMB5aokWbf9KWuWv3UDC/`, 2019.

[Brains(2019)] Java Brains. Spring security. `https://www.youtube.com/playlist?list=PLqq-6Pq4lTTYTEooakHchTGglSvkZAjnE/`, 2019.

[is A RESTful API?(2017)] What is A RESTful API? Traversy media. `https://www.youtube.com/watch?v=Q-BpqyOT3a8/`, 2017.

[in 8 Hours | Angular Tutorial For Beginners(2019)] Angular 8 Full Course Learn Angular in 8 Hours | Angular Tutorial For Beginners. edureka! `https://www.youtube.com/watch?v=oXr4lpXEg1o/`, 2019.