Specyfikacja implementacyjna dla projektu pt. „Noticeboard”

Wykonał: Franciszek Wysocki
Sprawdzający: mgr inż. Paweł Zawadzki
Data: 24-03-2021

Spis treści

1 Cel dokumentu

Celem dokumentu jest przedstawienie planów implementacyjnych dotyczących projektu „Noticeboard". Zostaną w nim opisane szczegóły dotyczące środowiska deweloperskiego, struktury projektu, jak również zasady wersjonowania i testowania.

2 Cel aplikacji

Celem projektu jest stworzenie prostego serwisu ogłoszeniowego - aplikacji webowej w architekturze klient-serwer. Będzie ona rozwiązywać problem ograniczonego zasięgu ogłoszeń na tradycyjnych tablicach czy słupach reklamowych.

Serwis ten będzie umożliwiać m.in. rejestrację oraz logowanie do serwisu, wyświetlanie/filtrowanie stron z ogłoszeniami (bez logowania), jak również dodawanie/kasowanie/edycja ogłoszeń (po uwierzytelnieniu).

Aplikacja będzie także odpowiednio zabezpieczona - walidacja danych, hashowanie hasła, reakcja na błędy ze strony użytkownika.

3 Środowisko deweloperskie

Aplikacja będzie tworzona w systemie Ubuntu 20.04. Front-end aplikacji zostanie napisany w JavaScriptcie (ES6) z wykorzystaniem biblioteki React (v. 17.0.1) i biblioteki Bootstrap 5 (v. 5.0). Back-end aplikacji zostanie napisany w Javie (openjdk v. 13.0.4) w oparciu o framework Spring Boot (v. 2.4.3). Zostanie również utworzony serwer bazy danych PostgreSQL

Kontakt pomiędzy front-endem, a back-endem odbywać się będzie za pomocą metod HTTP zgodnie ze stylem architektury oprogramowania REST (każde zapytanie płynące z front-endu będzie musiało zawierać komplet informacji). Formatem wymienianych danych będzie JSON. Do stwierdzania uwierzytelniania użytkowników w żądaniach zostanie zaimplementowany Basic Auth.

Narzędzia:

- IntelliJ IDEA Ultimate 2020.3 - zintegrowane środowisko programistyczne (back-end);
- Visual Studio Code 1.54.3 - zintegrowane środowisko programistyczne (front-end);
- Postman (v. 7.36.5) - manualne testowanie żądań;
- Git (v. 2.25.1) - system kontroli wersji;

- Github - hosting repozytorium;
- Npm (v. 6.14.4) - menadżer pakietów (front-end);
- Apache Maven (v. 3.6.3) - narzędzie automatyzujące budowę oprogramowania (back-end);
- Docker (v. 19.03.13) - narzędzie do konteneryzacji;
- SQuirreL (v. 4.1.0) - klient GUI do PostgreSQL.

4 Zasady wersjonowania

Wersjonowanie odbędzie się za pomocą systemu kontroli wersji (git). Nazwy branchy, tagów i commitów będą pisane w języku angielskim. Praca z systemem kontroli wersji git będzie rozłożona na wiele gałęzi. Łączenie ich będzie wykonywane za pomocą komendy git merge.

W repozytorium znajdą się oddzielne katalogi na front-end, back-end, konfiguracje bazy danych i dokumentację.

5 Uruchomienie

Założeniem aplikacji jest jest jej ciągła praca na serwerze, zatem nie zakłada się jej częstego uruchamiania. W tym celu jednak można będzie wykorzystać dockera, uruchamiając odpowiednie kontenery, bądź skorzystać z poleceń:

`npm start` - dla aplikacji front-endowej;
`java -jar nazwaAplikacji` - dla aplikacji back-endowej.

6 Bezpieczeństwo

Do zabezpieczenia aplikacji zostanie wykorzystany framework Spring Security. Endpointy służące dodawaniu/edytowaniu/kasowaniu ogłoszeń oraz edycji danych użytkownika będą wymagały uwierzytelnienia. Wgląd w panel administracyjny będzie wymagał dodatkowych uprawnień administratora.

Hasła przechowywane w bazie danych będą hashowane za pomocą funkcji hashującej Bcrypt, a uwierzytelnianie requestów odbywać się będzie za pomocą BasicAuth, wykorzystując funkcję szyfrującą Base64.

7 Obsługa danych

Do modyfikowania danych (w bazie danych) po stronie serwera zostanie wykorzystany moduł Spring Data JPA.

Dane wprowadzane przez użytkownika będą walidowane zarówno przez front-end (aby zapewnić szybki feedback np. odnośnie złożoności hasła), jak i przez back-end aplikacji (np. aby sprawdzić czy użytkownik o podanej nazwie użytkownika już istnieje).

Pobieranie danych będzie podlegało paginacji i lazy-loadingowi, co przyspieszy ładowanie ogłoszeń.

Użytkownicy będą przechowywani w trwałym magazynie danych.

8 Testowanie

Testy aplikacji będą pisane przed tworzeniem samego kodu zgodnie z techniką TDD i zgodnie z zasadami F.I.R.S.T.

1. Back-end

   Biblioteki, które zostana wykorzystane: JUnit 4, AssertJ oraz Mockito.

   Nazwy testów będą pisane zgodnie z konwencją:

   `nameOfTheMethod_stateUnderTest_expectedBehavior`

   Testy będą automatycznie uruchamiane za pomocą Mavena przed stworzeniem pliku JAR, bezpośrednio uruchamiając je w IDE lub wywołując `mvn test`.

2. Front-end

   Zostanie wykorzystana biblioteka: testing-library oraz framework „Jest" (do mockowania).
   Nazwy testów będą dopełniały funkcję testową „it" np.

   (it) `'hides login page when user logged in'`

   Testy będą uruchamiane automatycznie przez npm po każdym zapisaniu zmian w projekcie, po wcześniejszym wywołaniu `npm test`.

Zarówno testy back-endowe, jak i front-endowe będą pisane zgodnie ze schamatem given/when/then.

9 Struktura, klasy i moduły w projekcie

1. Back-end

   Struktura projektu będzie zgodna z tą narzuconą przez Mavena. Dodatkowo powstaną pakiety:

   - controllers - pakiet przechowujący klasy, będące restowymi kontrolerami - w tym pakiecie zostaną wyróżnione podpakiety przechowujące m. in. klasy UserController, NoticeController, LoginController;
   - repositories - pakiet przechowujący interfejsy, służące komunikacji z bazą danych - w tym pakiecie zostaną wyróżnione podpakiety przechowujące m. in. interfejsy UserRepository, NoticeRepository;
   - services - pakiet przechowujący klasy, służące obsłudze danych (przełożenie odpowiedzialności z kontrolerów) - w tym pakiecie zostaną wyróżnione podpakiety przechowujące m. in. klasy UserService, NoticeService;
   - entities - pakiet przechowujący klasy, będące encjami (tabelami w bazie danych) - w tym pakiecie zostaną wyróżnione podpakiety przechowujące m. in. klasy User, Notice;
   - dto - pakiet przechowujący klasy, których obiekty będą służyły wymienie danych - w tym pakiecie zostaną wyróżnione podpakiety przechowujące m. in. klasy UserDTO, NoticeDTO;
   - configurations - pakiet przechowujący klasy, służące konfiguracji aplikacji np. SecurityConfiguration;
   - errors - pakiet przechowujący wyjątki i klasy odpowiedzialne za obługę błędów.

2. Front-end

   Kod będzie umieszczony w odpowiednich pakietach w folderze src.

   - api - pakiet przechowujący moduł odpowiedzialny za wysyłanie requestów;
   - components - pakiet przechowujący komponenty Reacta takie jak np. NoticeList.js czy TopBar.js;
   - pages - pakiet przechowujący strony takie jak strona logowania, główna, rejestracji, administratora, dodawania/edycji ogłoszeń - w rzeczywistości to również będą komponenty Reacta, jednak będą pełniły funkcję podstron, zmienianych przy wykorzystaniu React Router.

3. Baza danych

   W bazie danych zostaną utworzone m. in. tabele:

   - User - tabela reprezentująca użytkownika aplikacji, przechowująca takie dane jak: id użytkownika, email, nazwa użytkownika, nazwa do wyświetlenia, hasło, rolę i ogłoszenia danego użytkownika;
   - Notice - tabela reprezentująca ogłoszenie, przechowująca takie dane jak: id ogłoszenia, id użytkownika (autora), lokalizacja, tytuł, opis.