Sprawozdanie z pracy projektowej

Ewelina Lasowy, Szymon Skrzypacz, Mateusz Żmijewski

Maj 2020

1 Opis funkcjonalny systemu

HypE-Learning to aplikacja, która służy do nauczania zdalnego. Student, po wcześniejszej rejestracji i akceptacji administratora strony, może przeglądać kursy, do których jest zapisany, rozwiązywać quizy oraz przesyłać rozwiązania zadań w pliku .pdf. Natomiast wykładowca ma możliwość zarządzania swoimi kursami, poprzez dodawanie i edytowanie treści. Co więcej, może przeglądać pliki przesłane przez studentów. Są cztery typy użytkowników.

Uprawnienia użytkownika niezalogowanego:

- rejestracja/logowanie
- przeglądanie listy wszystkich dostępnych kursów

Uprawnienia użytkownika zalogowanego (student) - wszystkie uprawnienia użytkownika niezalogowanego oraz:

- przeglądanie kursów do których jest zapisany
- edycja swojego profilu
- rozwiązywanie quizów
- dodawanie rozwiązań zadań (w formie PDF)

Uprawnienia użytkownika zalogowanego (wykładowca) - wszystkie uprawnienia użytkownika niezalogowanego oraz:

- edycja swojego profilu
- przeglądanie/dodawanie/edytowanie/usuwanie quizów
- przeglądanie/dodawanie/edytowanie/usuwanie kursów
- przeglądanie/dodawanie/edytowanie/usuwanie tematów
- przeglądanie zadań przesłanych przez studentów

Uprawnienia użytkownika zalogowanego (administrator) - wszystkie uprawnienia użytkownika niezalogowanego i wykładowcy oraz:

- przeglądanie logów (na Heroku)
- banowanie użytkowników
- zarządzanie rolami użytkowników

2 Streszczenie opisu technologicznego

Backend został napisany w języku TypeScript z wykorzystaniem frameworka NestJS. Projekt wykorzystuje relacyjną bazę danych PostgreSQL. Wersja mobilna aplikacji powstała przy użyciu frameworka Flutter w języku Dart i została wykonana przez Mateusza Żmijewskiego. Wersję desktopową z wykorzystaniem frameworka Angular (język TypeScript) i Electrona zrealizował Szymon Skrzypacz. Wersja webowa została utworzona w frameworku VueJS w języku JavaScript przez Ewelinę Lasowy.

3 Streszczenie wykorzystanych wzorców projektowych w każdej aplikacji

Flutter:

- komunikacja pomiędzy widgetami, wstrzykiwanie zależności oraz zarządzanie stanem aplikacji odbywa się za pomocą Providera. Widgety znajdujące się w hierarchii poniżej Providerów są w stanie nasłuchiwać zmian o pojawieniu się których informuje konkretny Provider.
- warstwa modeli oddzielona od warstwy interfejsu użytkownika

Angular:

- podział na moduły, dzięki czemu kod jest bardziej czytelny
- wzorzec obserwatora

VueJS:

- Vuex - wzorzec zarządzania stanem + biblioteka dla aplikacji Vue.js. Służy jako scentralizowany magazyn dla wszystkich komponentów aplikacji, z regułami zapewniającymi, że stan może być zmutowany tylko w sposób przewidywalny

4 Instrukcje lokalnego i zdalnego uruchomienia systemu

Instrukcja lokalnego uruchamiania testów i systemu:

Backend:

- pobranie repozytorium `https://github.com/DEViper/hype_learning_api/tree/feature/student`
- zainstalowanie wszystkich potrzebnych paczek poleceniem
  `$ npm i`
- uruchomienie projektu
  `$ npm run start`
- otworzenie przeglądarki
- wpisanie adresu `localhost:3000`

Flutter:

- pobranie repozytorium `https://github.com/DEViper/hypE-Learning-Mobile`
- zainstalowanie wszystkich potrzebnych paczek:
  `$ flutter pub get`
- potrzebne jest podłączenie telefonu z systemem android lub wybranie emulowanego modelu z Android Studio AVD Manager
- uruchomienie projektu
  `$ flutter run`
- aby uruchomić testy, należy użyć polecenia
  `$ flutter test test/widget_test.dart`

Angular:

- pobranie repozytorium `https://github.com/DEViper/hypE-Learning-Desktop`
- zainstalowanie wszystkich potrzebnych paczek:
  `$ npm i`

- uruchomienie projektu
  `$ ng serve -o`

- otworzenie przeglądarki
- wpisanie adresu urllocalhost:4200
- aby wygenerować aplikację desktopową z użyciem Electrona należy ściągnąć wszystkie narzędzia potrzebne do pracy z Electronem `https://www.electronjs.org/docs/development/build-instructions-windows` oraz program Bridge.exe (podstawowy plik w Electronie) i umieścić plik w głównym katalogu projektu
- następnie należy wpisać polecenie do konsoli
  `$ npm run auto`
  .Ta instrukcja uaktualnia wszystko do najnowszych wersji oraz generuje aplikację .exe.
- aby uruchomić testy, należy użyć polecenia
  `$ ng test`

VueJS:

- pobranie repozytorium `https://github.com/DEViper/hypE-Learning-Web`
- zainstalowanie wszystkich potrzebnych paczek:
  `$ npm i`

- uruchomienie projektu
  `$ npm run serve`

- otworzenie przeglądarki
- wpisanie adresu `localhost:8080`
- aby uruchomić testy, należy użyć polecenia
  `$ npm run test:unit`

Instrukcja zdalnego uruchamiania systemu:

Backend:

- API zostało wrzucone na serwer Heroku i znajduje się pod tym linkiem: `https://hype-learning.herokuapp.com/`.
- natomiast pod linkiem `https://hype-learning.herokuapp.com/api/` znajduje się dokumentacja w Swaggerze.

Flutter:

- plik app-release.apk został wrzucony na GitHub i znajduje się pod adresem: `https://github.com/DEViper/hypE-Learning-Mobile`

Angular:

- folder HypE-Learning -win32-x64 został wrzucony na GitHub i znajduje się pod adresem: `https://github.com/DEViper/hypE-Learning-Desktop`

VueJS:

- na Netlify została wdrożona aplikacja webowa i można ją znaleźć pod adresem `https://elated-hugle-d6b5e0.netlify.app/`.

5 Wnioski projektowe

- źle napisana specyfikacja
- regularna praca nad projektem pozwala wykonać projekt nawet w oparciu o najgorszą specyfikację
- taki projekt, który wymagał napisania zarówno backendu jak i frontendu dla trzech różnych platform to było ciekawe wyzwanie i bardzo pouczające
- praca w zespole układała się bardzo dobrze