Projektowanie i programowanie systemów internetowych II

Ice Cream Places

1 Opis funkcjonalny systemu

Projekt aplikacji internetowej przeznaczonej do umieszczania informacji o lokalach sprzedających lody przez ich właścicieli wraz z oferowanymi rodzajami lodów oraz godzinami otwarcia. Użytkownicy odwiedzający stronę mogą przeglądać i wyszukiwać zarejestrowane lokale oraz ich ofertę, natomiast zalogowani użytkownicy mogą także dodawać lodziarnie do swoich ulubionych, oceniać i recenzować lodziarnie oraz otrzymują dostęp do panelu powiadomień. Dodatkowo użytkownicy wyznaczeni przez właścicieli lodziarni, mogą w tych lodziarniach zmieniać dostępne smaki oraz godziny otwarcia.

1.1 Funkcjonalność aplikacji obejmuje:

- Uwierzytelnianie użytkownika (również SocialMedia);
- Rejestracja i logowanie;
- Profil użytkownika;
- Resetowanie hasła;
- Wyszukiwarka;
- Multitenancy - wielodostępność i system autoryzacji akcji użytkowników;
- Wyświetlanie szczegółów o lodziarniach wraz z mapą;
- Zarządzanie smakami lodów;
- Możliwość edycji oferty przez właściciela oraz pracowników lokalu;
- Dodawanie opinii na temat lokali;
- Migracja rzeczywistych danych - ograniczona do obszaru miasta Legnica.

2 Streszczenie opisu technologicznego

W projekcie zdecydowaliśmy korzystać ze stack-u MERN (MongoDB, Express.js, React.js, Node.js), ze względu na poprzednie doświadczenie niektórych członków zespołu oraz względnie przystępny proces wdrożenia w technologię dla osób, które nie miały z nią styczności. Ponadto wskazane technologie poniżej posiadają przystępną w odbiorze oraz obszerną dokumentację.

- **GitHub** - miejsce do kontroli wersji kodu. Pozwala na łatwą synchronizację między członkami grupy oraz na prostą kontrolę zmian w kodzie.
- **Asana** - platforma do kontrolowania workflow. Rozpisujemy na niej taski. Dzięki temu możemy kontrolować postępy kodowania aplikacji, przypisywać zadania odpowiednim osobom oraz zdobywać założone kamienie milowe.
- **Visual Studio Code** - edytor kodu źródłowego, odpowiada za poprawę składni i przyspiesza pisanie kodu.
- **JavaScript** - język programowania, najczęściej spotykany i używany na stronach internetowych, jest mocno związany z technologiami wymienionymi poniżej.
- **MongoDB** - nie relacyjna baza danych, pracująca w chmurze, pozwala nam przechowywać dane.
- **Express.JS** - framework pracujący po stronie serwera. Pomaga nam mapować endpointy, które uruchamiają funkcjonalność po stronie serwera. Pomaga obsługiwać zapytania oraz odpowiedzi HTTP.
- **React.JS** - elastyczna biblioteka frontendowa, bazująca na komponentach wielokrotnego użytku. W naszym projekcie komponenty będą wykorzystywane w wielu miejscach.
- **Node.JS** - środowisko do uruchamiania javascriptu, pozwala na obsługę backendu.

3 Instrukcja lokalnego i zdalnego uruchomienia

3.1 Lokalne

3.1.1 Backend

- Zainstaluj Node.js
- Zainstaluj klienta Git
- Uruchom konsole i wpisz:

```
git clone https://github.com/Ice-Cream-Places-ICP/API.git
```

- Następnie przejdź do folderu z projektem otwórz konsolę i wpisz:

```
npm install
```

- Poczekaj na koniec procesu i wpisz w konsoli:

```
npm run dev
```

- Poprawne działanie aplikacji sygnalizuje informacja:

```
Server up at port
```

Aplikacja potrzebuje do działania pliku **.env**, w którym będą następujące informacje:

- **PORT** - numer portu używany przez aplikację
- **API_URL** - adres URL aplikacji
- **WEB_PORT** - adres URL klienta
- **TOKEN_SECRET** - token wykorzystywany do kodowania i dekodowania tokenu JWT
- **DB_CONNECTION** - ciąg znaków do podłączenia bazy danych MongoDB
- **TEST_DB_CONNECTION** - ciąg znaków do podłączenia bazy danych MongoDB wykorzystywanej w testach
- **USER** - adres email do wysyłania maili przez aplikację
- **PASS** - hasło do poczty email (z konfiguracji dla aplikacji)
- **GOOGLE_CLIENT_ID** - identyfikator klienta otrzymany od google
- **GOOGLE_CLIENT_SECRET** - tajny klucz klienta otrzymany od google
- **FACEBOOK_CLIENT_ID** - identyfikator klienta otrzymany od facebook
- **FACEBOOK_CLIENT_SECRET** - tajny klucz klienta otrzymany od facebook

3.1.2 Frontend

- Zainstaluj Node.js
- Zainstaluj klienta Git
- Uruchom konsole i wpisz:

```
git clone https://github.com/Ice-Cream-Places-ICP/WEB.git
```

- Następnie przejdź do folderu z projektem, otwórz konsolę i wpisz:

```
npm install
```

- Poczekaj na koniec procesu i wpisz w konsoli:

```
npm start
```

Po tych krokach powinna uruchomić się przeglądarka internetowa z adresem **localhost:3000** i po chwili wyświetlić stronę startową projektu.

3.2 Zdalne

Aplikacja jest podzielona na backend oraz frontend. Wymagało to od nas zastanowienia się nad serwisem hostującym naszą aplikację. Decyzja padła na platformę Vercel, która spełnia wszystkie nasze wymagania. Pozwala uruchomić aplikację wykorzystującą React.js oraz Node.js, Express.js. Baza danych, która jest niezbędna do prawidłowego działania aplikacji wykorzystuje platformę MongoDB. Platforma Vercel zapewnia darmowe hostowanie aplikacji oraz darmową domeny, które wykorzystujemy jako adresy URL naszej aplikacji. Pozwala na używanie zmiennych środowiskowych, zapisuje logi aplikacji oraz na łatwą koordynację wdrażania aplikacji. Dzięki połączeniu platformy z GitHub każda zmiana jest automatycznie wykrywana i Vercel tworzy nowy build, który jako można wykorzystać jako wersja produkcyjna. Aby platforma Vercel odpowiednio współpracowała z aplikacją backendu wymaga pliku konfiguracyjnego vercel.json w głównym katalogu repozytorium. Przykład takiego pliku dla aplikacji Node.js:

```
{
"version": 2,
"builds": [
    {
    "src": "./index.js",
    "use": "@vercel/node"
    }
],
"routes": [
    {
    "src": "/(.*)",
    "dest": "/"
    }
]
}
```

Aplikacja frontendu wymaga mniej zaangażowania, ponieważ wystarczy aby platforma Vercel miała możliwość śledzenia repozytorium. i Vercel automatycznie zacznie taką aplikacje hostować. Problemem, o którym na początku projektu nie pomyśleliśmy, był podzielony projekt na dwa repozytoria. Spowodowało to, iż najbardziej rozsądnym rozwiązaniem było przypisanie aplikacji frontendu i backendu dwóch różnych domen. Jednak ostatecznie udało się rozwiązać wszystkie problemy związane z zastosowaniem takiego rozwiązania.

4 Instrukcja uruchomienia testów oraz opis testowaych funkcjonalności

4.1 Backend

W celu uruchomienia testów należy skorzystać z instrukcji lokalnego uruchomienia. Następnie znajdując się w folderze projektu należy wprowadzić w terminal komendę:

```
npm run test.
```

Wykorzystywane w projekcie frameworki testowe to:

- **jest** https://github.com/facebook/jest
- **supertest** https://github.com/ladjs/supertest

W projekcie występują testy integracyjne, które sprawdzają działanie aplikacji oraz poprawność operacji przez nią wykonywanych na bazie danych. W testach wykonywane są zapytania do aplikacji i sprawdzana jest odpowiedź w celu określenia jej poprawności. Pokrycie testowe obejmuje takie funkcjonalności jak:

- rejestracja
- logowanie
- pobieranie danych sklepu

Spis scenariuszy testowych wygenerowany po uruchomieniu testów:

```
PASS  tests/sequentialRunner.test.js (9.26 s)
    sequentially run tests
POST /auth/login
given any body
        should specify json in the content type header (2 ms)
        should contain json message in reponse body
given valid email and password
        should return body containing token (1 ms)
        should return valid token (1 ms)
        should return "Login Successfully" message
        should return status code 200
given invalid email or password
        should return "Invalid credentials" message
        should not return token (1 ms)
        should return status code 400 (1 ms)
when the email or password is missing
```

```
        should return "All fields are required" message
        should not return token
        should return status code 400 (1 ms)
POST /auth/register
given any body
        should specify json in the content type header (349 ms)
        should contain json message in reponse body (80 ms)
given valid email and password
        should return "New user created" message (285 ms)
        should respond with status code 200 (229 ms)
given invalid email
        should return "Invalid email" message (18 ms)
        should return status code 400 (22 ms)
given invalid password
        should return "Password must contain minimum 8 letters containing at least - 1 low
        should return status code 400 (24 ms)
when the email or password is missing
        should return status false (44 ms)
        should return "All fields are required" message (52 ms)
        should return status code 400 (37 ms)
when user already exists
        should contain "User already exists" json message inside body (211 ms)
        should return status false (192 ms)
        should return status code 400 (200 ms)
GET /shops/:id
given valid id
        should specify json in the content type header (28 ms)
        should contain json message in reponse body (27 ms)
given valid id of shop that exists within database
        should return "Shop retrieved" message (25 ms)
        should return body containing all fields with valid values (27 ms)
        should return status code 200 (23 ms)
given valid id of shop that doesn't exists within database
        should return "Shop not found" message (37 ms)
        should return status code 400 (25 ms)
given invalid id
        should return "Invalid shop id" message (20 ms)
```

```
    should return status code 400 (19 ms)
```

4.2 Frontend

W celu uruchomienia testów należy skorzystać z instrukcji lokalnego uruchomienia. Następnie znajdując się w folderze projektu należy wprowadzić w terminal komendę:

```
npm test
```

Wykorzytany w projekcie pakiet testowy

- **testing-library-react** https://testing-library.com/docs/react-testing-library/intro/

W projekcie występują testy, które sprawdzają poprawność wyświetlenia danych w odpowiednich komponentach. Testy obejmują również sprawdzenie dostęp do odpowiednich stron/komponentów w zależności od rodzaju użytkownika. Spis elementów testowanych wygenerowany po uruchomieniu testów:

```
PASS  src/tests/components/AdminUsersProfile.test.js (7.473 s)
PASS  src/tests/components/Header.test.js (8.676 s)
PASS  src/tests/components/DrawerNavbar.test.js
PASS  src/tests/components/AdminHeader.test.js
PASS  src/tests/components/AdminShops.test.js
  Console

    console.log
      []

      at src/services/shop.js:9:15

    console.log
      [
        {
          name: 'name: test 1',
          address: {
            country: 'country: test 1',
            city: 'city: test 1',
            postCode: 'postCode: test 1',
            streetName: 'streetName: test 1',
            streetNumber: 'streetNumber test 1'
          },
          openingHours: {
```

```
        weekDay: 'weekDay test 1',
        startHour: 'startHour test 1',
        startMinute: 'startMinute test 1',
        endHour: 'endHour test 1',
        endMinute: 'endMinute test 1'
      },
      flavors: [ [Object] ],
      _id: '1'
    }
  ]

  at src/services/shop.js:9:15

 PASS  src/tests/components/ShopCard.test.js
 PASS  src/tests/components/AdminNavigation.test.js
 PASS  src/tests/components/AdminUsers.test.js
 PASS  src/tests/components/ProfileNavigation.test.js
 PASS  src/tests/components/ProfileNotification.test.js
 PASS  src/tests/components/ProfileView.test.js
 PASS  src/tests/pages/Lost.test.js
 PASS  src/tests/components/ProfileEdit.test.js
 PASS  src/tests/components/ProfileFavorite.test.js

Test Suites: 14 passed, 14 total
Tests:       15 passed, 15 total
Snapshots:   0 total
Time:        29.599 s
Ran all test suites.
```

5 Linki do dokumentacji projektu

- **Backend** - https://github.com/Ice-Cream-Places-ICP/API
- **Frontend** - https://github.com/Ice-Cream-Places-ICP/API

6 Wnioski projektowe

Proces tworzenia aplikacji okazał się dużym wyzwaniem zarówno pod względem wymagań funkcjonalnych, jaki komunikacji i współpracy w zespole. W trakcie realizacji projektu staraliśmy się wykorzystywać narzędzia ułatwiające pracę zespołową - początkowo w celu implementacji metodyki SCRUM wykorzystywaliśmy narzędzie o nazwie Asana, później natomiast przenieśliśmy się na platformę GitHub. Posunięcie te uprościło orientowanie się w postępach projektu, ponieważ mogliśmy korzystać z narzędzia GitHub Projects, ale przede wszystkim rozpisywać zadania w formie Issues. Początkowo w grupie były przeprowadzane dość częste spotkania, na których wymienialiśmy się informacjami na temat poczynionych postępów, planów, czy też uwagami na temat implementacji określonych rozwiązań. W pewnym momencie natomiast częstotliwość spotkań zmalało, a część członków projektu nieco straciła zainteresowanie w związku z czym pojawiła się stagnacja w postępach. Przełomowym etapem w projekcie było wdrożenie aplikacji na serwer, co pozwoliło na bezpośrednią obserwację efektów pracy, a także wzrost motywacji w zespole (niestety, nie każdy członek zespołu ten wzrost poczuł). Ostatecznie projekt udało się doprowadzić do stanu używalności oraz spełnić większość założeń funkcjonalnych. Pomimo wielu problemów aplikacja na obecnym etapie funkcjonuje w sposób zadowalający. Pewne funkcjonalności wciąż wymagają dopracowania, a niektóre nawet nie zostały jeszcze zaimplementowane, natomiast są one zaplanowane i rozwijane.