Wynajem Zakwaterowania

Wykonali:
Adrian Bełciak
Kamil Michno

Spis treści

A. Identyfikacja zagadnienia biznesowego

Grupę docelową aplikacji stanowią osoby podróżujące po Polsce. Aby korzystać ze wszystkich funkcjonalności aplikacji, użytkownik powinien posiadać konto w serwisie społecznościowym Facebook. Aplikacja umożliwia znalezienie apartamentu spełniającego wymagania użytkownika w krótkim czasie. Aplikacja rozwiązuje problem szukania noclegu w innym mieście, zarówno przez pojedynczych gości, jak i dla większych grup.

Harmonogram prac:

- utworzenie schematu bazy danych
- utworzenie back-endu
- utworzenie front-endu
- testy aplikacji
- utworzenie dokumentacji

B. Określanie wymagań

Wymagania funkcjonalne:

- Możliwość przeglądania apartamentów
- Możliwość szukania hoteli według kryteriów:
    - miasto
    - przedział czasowy
    - liczba osób
- Możliwość zalogowania użytkownika
- Możliwość rezerwacji apartamentu
- Możliwość rezygnacji z rejestracji hotelu
- Panel administracyjny dla administratora umożliwiający zarządzanie:
    - apartamentami
    - rezerwacjami
    - użytkownikami

Wymagania sprzętowe:

Aplikacja wymagać będzie dostępu do Internetu.
Obsługiwane systemy operacyjne:

- Windows (min. Windows 7)
- Linux

Obsługiwane przeglądarki (w najnowszych wersjach):

- Firefox

- Google Chrome
- Opera

Używane technologie

- Relacyjna baza danych MySQL
- Node.js (Express)
- React.js

C. Zadaniowy harmonogram prac i podział na członków zespołu

Zadania:

- stworzenie repozytorium GIT + dodanie kontrybutorów
- uruchomienie środowiska w Node.js i React.js
- utworzenie schematu bazy danych (MySQL)

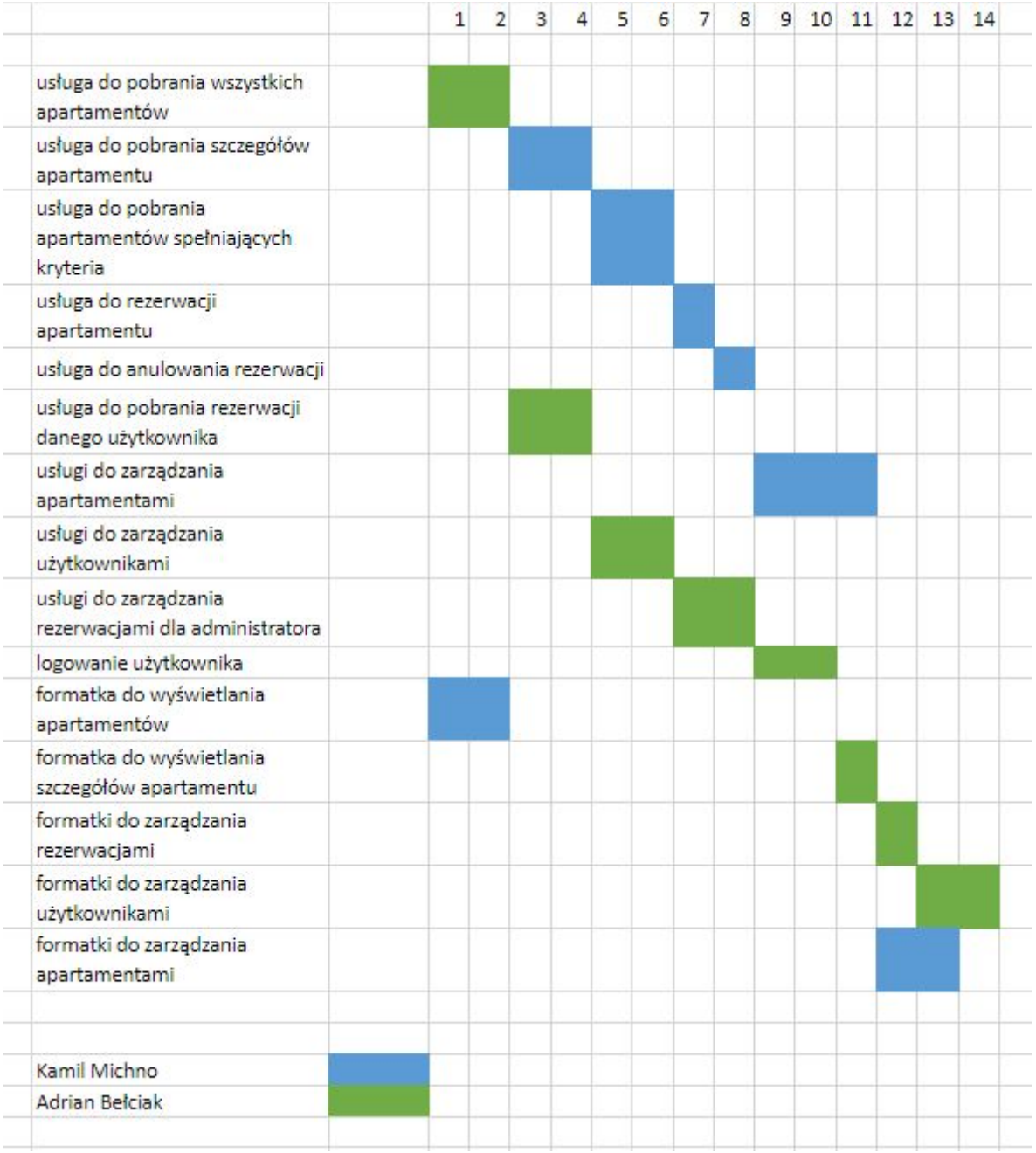
Backend:

- usługa do pobrania wszystkich apartamentów
- usługa do pobrania szczegółów apartamentu
- usługa do pobrania hoteli według kryteriów wyszukiwania:
    - miasto
    - przedział czasowy
    - liczba osób
- usługa logowania użytkownika
- usługa rezerwacji apartamentu
- usługa panelu administracyjnego dla administratora

Frontend:

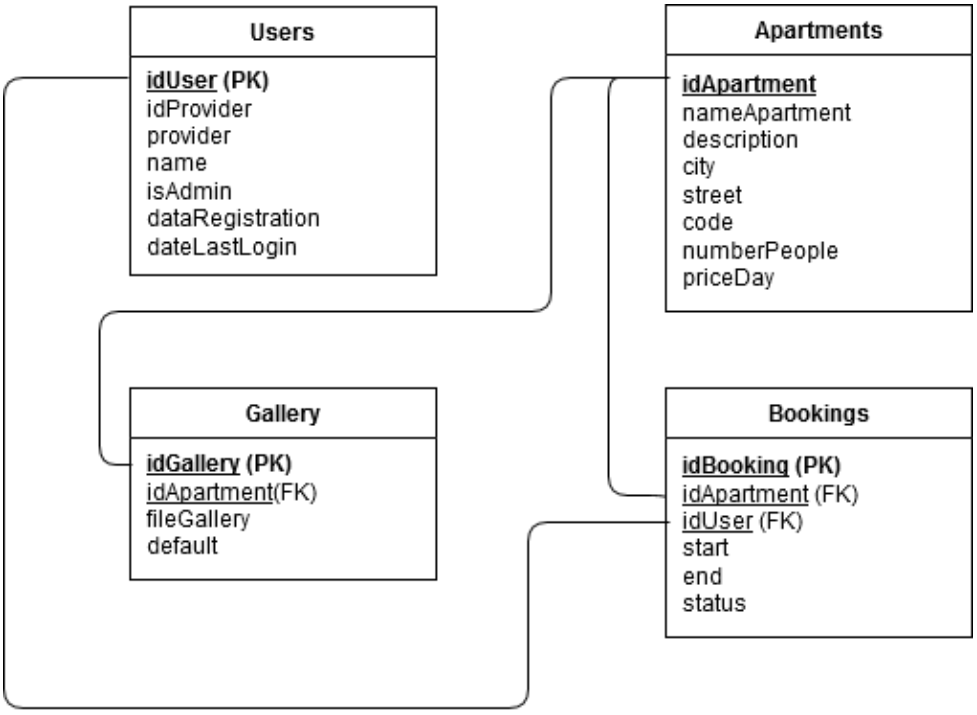
- logowanie użytkownika
- formularz wyszukiwania wolnych apartamentów
- panel administracyjny do zarządzania
    - apartamentami
    - rezerwacjami
    - użytkownikami
- strona do wyświetlania szczegółów danego hotelu
- strona do wyświetlania własnych rezerwacji

Wykres Gantta



D. Analiza zagadnienia i jego modelowanie

Diagram encji



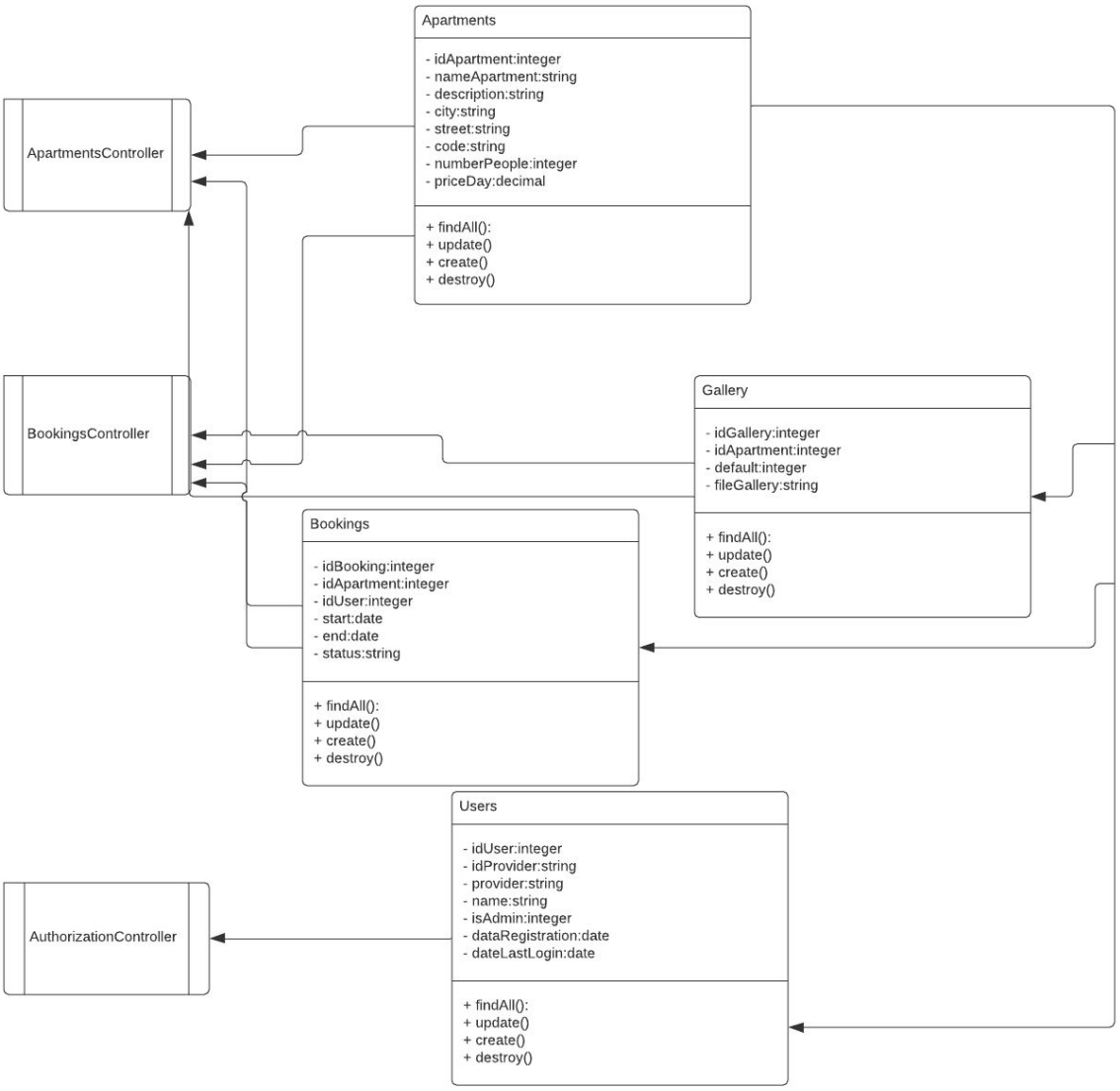
Baza danych

- tabela “Apartments”
  - **idApartment (PK)**
  - nameApartment
  - description
  - city
  - street
  - code
  - numberPeople
  - priceDay
- tabela “Bookings”
  - **idBooking (PK)**
  - idApartment (FK)
  - idUser (FK)
  - start
  - end
  - status
- tabela “Gallery”
  - **idGallery (PK)**
  - idApartment (FK)
  - fileGallery
  - default
- tabela “Users”

- **idUser (PK)**
- idProvider
- provider
- name
- isAdmin
- dateRegistration
- dataLastLogin

PK - Primary Key, Klucz Główny
FK - Foreign Key, Klucz obcy

Diagram klas



E. Implementacja

Część backend

Część backend została wykonana w środowisku Node.js (Express framework).

Najważniejsze moduły (paczki NPM) wraz z wersjami

- "cors": "^2.8.5",
    - Cross-origin resource sharing (w skrócie CORS) – mechanizm umożliwiający współdzielenie zasobów pomiędzy częścią backend oraz frontend (aplikacje nasłuchują na różnych portach)
- "express-session": "^1.16.2",

    - Wykorzystywany do tworzenia sesji opartych na ciasteczkach. Wykorzystywane do podtrzymywania sesji zalogowania.
- "multer": "^1.4.2",
    - Wykorzystywany do wgrywania plików na serwer (zdjęć)
- "mysql2": "^1.7.0",
    - Klient Mysql, wymagany przez pakiet “sequelize”
- "nodemon": "^1.19.2",
    - Wtyczka umożliwiająca zaczytywanie zmian w kodzie bez potrzeby restartu aplikacji node.js
- "passport": "^0.4.0",
    - Służy do autoryzacji użytkowników
- "passport-facebook": "^3.0.0",
    - Strategia paszportowa do uwierzytelniania użytkownika przy pomocy Facebooka za pomocą interfejsu API OAuth 2.0.
- "sequelize": "^6.0.0"
    - Wtyczka ORM (Object-Relational Mapping) mapowania obiektowo-relacyjnego do odwzorowania obiektowej architektury systemu informatycznego na bazę danych MySQL.

W części backendowej zastosowane zostały 3 główne pliki routingowe

- app.use('/apartments', apartmentsRouter);
    - Umożliwia obsługę apartamentów
- app.use('/bookings', bookingsRouter);
    - Umożliwia obsługę rezerwacji
- app.use('/authorization', authorizationRouter);
    - Umożliwia autoryzację użytkowników oraz operacje na ich uprawnieniach

Wgrywanie zdjęć

Przy implementacji wgrywania zdjęć dla apartamentów, przed nazwą pliku dopisywany jest znacznik czasu (timestamp - czas uniksowy) aby uniknąć kolizji nazw tego samego zdjęcia dla różnych obiektów. Odpowiada za to metoda

```
const storage = multer.diskStorage({
    destination: function(req, file, cb) {
    cb(null, 'uploads/')
    },
    filename: function(req, file, cb) {
    var uploadedFileName = Date.now() + '_' + file.originalname;
    cb(null, uploadedFileName)
    }
```

```
})
```

Autoryzacja użytkowników

Aby zalogować się w aplikacji potrzebne jest konto w portalu społecznościowym Facebook.com. Użytkownik po kliknięciu w części frontend przycisku “Zaloguj” odwołuje się do backendu, a następnie sprawdzane jest w bazie MySQL czy użytkownik istnieje (wtedy zostanie on zalogowany) lub w przeciwnym wypadku zostanie on zarejestrowany a następnie zalogowany. Odpowiada za to część kodu

```
passport.use(new FacebookStrategy(connectionFacebook.facebookParameters,

        function (accessToken, refreshToken, profile, cb) {
        UsersModel (sequelize).count({
        where: {
                idProvider: { [Sequelize.Op.eq]: profile.id  }
        }}).
        then(function(Users) {
        if (Users==1) { // jeśli użytkownik został znaleziony w bazie danych
                console.log("User found in DB");
                UsersModel (sequelize).update({
                dateLastLogin: new Date()},
                {
                where: {
                        idProvider: profile.id  }}).
                then(function(Users) {
                console.log ("Update last login date");
                }, function(error) {
                console.log("Error during update last login date");
                });
        }
        else if (Users==0) // jeśli użytkownik nie został znaleziony w bazie danych
        {
                console.log("User not found in DB");
                var insertUsers = {
                "idProvider": profile.id,
                "provider": "facebook",
                "name": profile.displayName,
                "dateRegistration": new Date()
                }
                UsersModel (sequelize).create(insertUsers).
                then(function(Users) {
                console.log("Add user to DB");
```

```
            }, function(error) {
            conosle.log("Error during add user to DB");
            });
        }
        }, function(error) {
        console.log("Error");
        });

        return cb(null, {
        user: profile.id,
        name: profile.displayName
        });
        }
    ));
```

W aplikacji została zaimplementowana autoryzacja użytkowników na podstawie konta Facebookowego, jednak w przyszłości istnieje możliwość dodania innych strategii (np. Google, Twitter obsługiwanych przez moduł Passport). W tym celu należy tylko w tabeli **Users** w kolumnie **provider** wskazać nowego dostawcę logowania i zaimplementować tę funkcjonalność. Użytkownik po zalogowaniu w części backend jest przekierowywany do strony głównej części frontend.

Część frontend

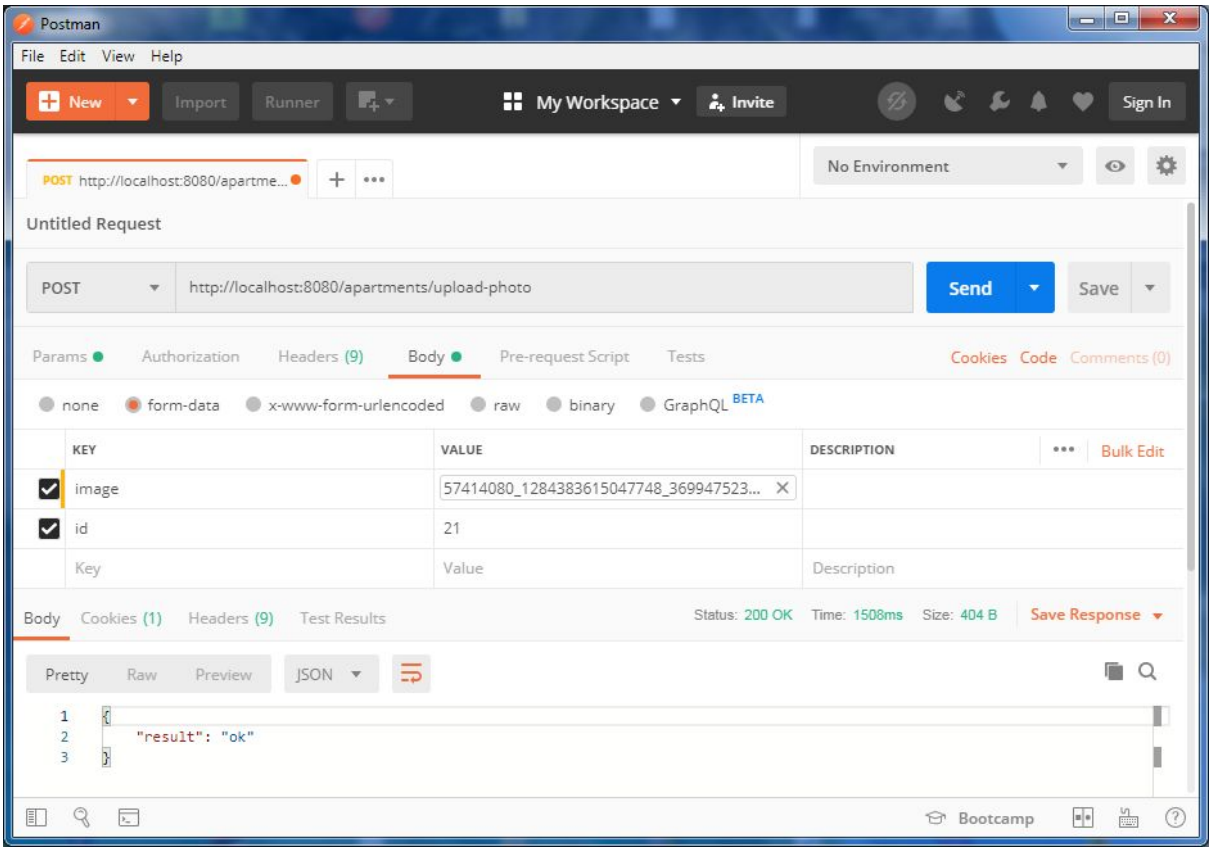
Część frontend została wykonana przy pomocy **React.js** - biblioteki języka programowania Javascript.

Wykorzystano poniższe najważniejsze moduły, zależności wraz z wersjami

- "moment": "^2.24.0",
    - Wykorzystywany przy kalendarzu, umożliwiający ustalenie lokalnego formatu daty i strefy czasowej
- "prop-types": "^15.7.2",
    - Pozwala na określanie tego jakie poperty jest w stanie obsłużyć komponent, wykorzystywane przy autoryzacji użytkowników
- "react-datetime-picker": "^2.7.1",
    - Wykorzystywane do prezentacji widoku kalendarza po kliknięciu ikony z symbolem kalendarze
- "react-images-upload": "^1.2.7",
    - Moduł wykorzystywany do formularza wgrywania zdjęć (apartamenty)
- "react-modal-image": "^2.4.0",
    - Moduł wykorzystywany do powiększania zdjęć po kliknięciu (przegląd apartamentów)
- "react-widgets": "^4.4.11",
    - Moduł wykorzystywany do widgetów takich jak ikona kalendarza

F. Testowanie

Zostały wykonane “ręczne” testy części **backend** oraz **frontend**.
Do testów skomplikowanej części backendowej wykorzystano aplikację **Postman** pozwalającą na tworzenie zapytań metodami PUT, GET, POST, DELETE.
Poniżej przedstawiono obraz z aplikacji Postman podczas testowania endpointa odpowiadającego za upload zdjęcia wybranego apartamentu.



Otrzymane zapytanie w aplikacji backend

```
  backend:server Listening on port 8080 +0ms
Executing (default): INSERT INTO `Gallery` (`idGallery`,`idApartment`,`default`,`fileGallery`) VALUES (DEFAULT,?,?,?);
POST /apartments/upload-photo 200 475.102 ms - 15
```

Proste zapytania części backendowej były testowane przy pomocy **cURL** - sieciowej biblioteki programistycznej dostępnej w systemie Linux.

Wykorzystanie biblioteki **cURL** podczas ręcznego testowanie dodawania apartamentów

```
λ curl -v -X POST -H "Content-Type: application/json" -d @post_apartment_insert.txt http://localhost:8080/apartments/add
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8080 (#0)
> POST /apartments/add HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.61.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 144
>
* upload completely sent off: 144 out of 144 bytes
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Vary: Origin
< Access-Control-Allow-Credentials: true
< Content-Type: application/json; charset=utf-8
< Content-Length: 15
< ETag: W/"f-51rzdr5zfwWQpf+iUepheZg7siQ"
< Set-Cookie: connect.sid=s%3A4vso7a5G-ao-sYBoo-XwzqsiS0C1BmqD.r5lMkC0BAOO4KZKo6asV5xMdvjuKRWbOqcFmztRF2jo; Path=/; HttpOn
ly
< Date: Sun, 15 Sep 2019 17:19:54 GMT
< Connection: keep-alive
<
{"result":"ok"}* Connection #0 to host localhost left intact
```

Zawartość pliku “post_apartment_insert.txt” (przesyłany obiekt JSON)

{
"name":"Testowy Gdańsk",
"description":"Morski Apartament",
"city":"Gdańsk",
"street":"Starówka 1",
"code":"55-678",
"people":"12",
"price":"199"
}

Logi aplikacji

```
Executing (default): INSERT INTO `Apartments` (`idApartment`,`description`,`city`,`street`,`code`) VALUES (DEFAULT,?,?,?,?);
POST /apartments/add 200 265.826 ms - 15
```

Dzięki powyższym testom możliwe jest sprawdzenie czy aplikacja działa prawidłowo (zwraca błąd, nie wykonuje żądania użytkownika) w przypadku złych danych/braków.

Przykład ręcznego testu, w którym dane zostały uzupełnione niepoprawnie aby zbadać zachowanie aplikacji

```
λ curl -v -X POST -H "Content-Type: application/json" -d @post_booking_insert.txt http://localhost:8080/bookings/add
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8080 (#0)
> POST /bookings/add HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.61.1
> Accept: */*
> Content-Type: application/json
> Content-Length: 75
>
* upload completely sent off: 75 out of 75 bytes
< HTTP/1.1 500 Internal Server Error
< X-Powered-By: Express
< Vary: Origin
< Access-Control-Allow-Credentials: true
< Content-Type: application/json; charset=utf-8
< Content-Length: 18
< ETag: W/"12-+X0SE4H2bNcuFZyA1DZLaKMpIMM"
< Set-Cookie: connect.sid=s%3AsVXlWlRG9KKp1t8dSm6P_8JBD4biYkR6.aUrz8%2FmKXz1c2nyzYMoxCoHCT1iGkxhILS0Q860Dvmg; Path=/; HttpOnly
< Date: Sun, 15 Sep 2019 17:25:04 GMT
< Connection: keep-alive
<
{"result":"error"}* Connection #0 to host localhost left intact
```

Żądanie dodania rezerwacji z zawartością pliku “post_booking_insert.txt” nie powiodło się ze względu na niepoprawny identyfikator użytkownika (nie istnieje taki użytkownik). Serwer zwrócił nam komunikat 500 i rezultat “error” zgodnie z zaimplementowaną częścią kodu

{
"idApartment":"7",
"idUser":"9999",
"start":"2019-09-23",
"end":"2019-09-27"
}

Powyższe testy można w przyszłości zaimplementować jako testy jednostkowe.

Część **frontend** została również przetestowana ręcznie, poprzez “klikanie” w przeglądarce internetowej i włączonym narzędziu debugującym - konsoli przeglądarkowej. Pozwala ona na wykrycie problemów w Javascript.

G. Instalacja/deployment/instrukcja użytkownika

Aby zainstalować aplikację, należy wykonać następujące kroki

1. Zainstalować środowisko uruchomieniowe **Node.js** w wersji 10, wydanie LTS (Long Term Support)
2. Zainstalować serwer bazodanowy **MySQL** lub **MariaDB**.
3. W katalogu docelowy pobrać zawartość projektu z repozytorium GIT
    a. Używając klienta **git** poleceniem
        git clone git@github.com:kmichno/apartment-rental-ui.git
    b. Bez używania klienta GIT, w przeglądarce wpisać poniższy adres i rozpakować archiwum
        https://github.com/kmichno/apartment-rental-ui/archive/master.zip
4. Zaimportować schemat bazodanowy z pliku *DB_Schema.sql* do bazy danych MySQL lub MariaDB

5. Przejść do katalogu **backend**, zmienić nazwę pliku **database.js**_original na **database.js** z parametrami naszej bazy danych (parametry zaznaczone kolorem żółtym powinny zostać zmienione)

```
module.exports = {
        databaseParameters: {
                username: 'user', - tutaj wpisujemy nazwę użytkownika bazodanowego
                password: 'pass', - tutaj wpisujemy hasło dla powyższego użytkowniak
                database: 'db', - tutaj wpisujemy nazwę bazy danych
                dialect: 'mysql',
                host: 'localhost', - tutaj wpisujemy host serwera bazodanowego
                port: '3306', - tutaj wpisujemy port serwera bazodanowego
                define: {
                        timestamps: false
                }
        }
}
```

6. Przejść do katalogu **frontend**, zmienić nazwę pliku **facebook.js_original** na **facebook.js**
    a. Następnie przejść na stronę https://developers.facebook.com/ i utworzyć aplikację **“Create App”** i jako produkt należy wybrać **“Facebook Login”**
    b. Po stworzeniu aplikacji w serwisie “Facebook for developers” należy skopiować **“App ID”** oraz **“App Secret”** i uzupełnić tymi danymi plik **facebook.js** (pola zaznaczone kolorem żółtym)

```
module.exports = {
        facebookParameters: {    clientID: 'XXX',  - tutaj wartość App ID
        clientSecret: 'YYY', - tutaj wartość App Secret
        callbackURL: 'http://localhost:8080/authorization/facebook/callback'
    }
}
```

7. Uruchomić terminal (CMD w Windows lub XTERM w Linux) i przejść do katalogu **backend**
    a. Należy pobrać zależności do projektu **backend** poleceniem
        npm install
    b. Po pobraniu zależności uruchomić **backend** poleceniem
        npm start
    c. Domyślnie aplikacja **backend** uruchomiona jest na porcie **8080**, aby sprawdzić czy powyższe kroki zostały wykonane prawidłowo, należy w przeglądarce wpisać adres
        http://localhost:8080/
    d. Powinniśmy otrzymać plik JSON z poniższą treścią

```
result "It works!"
```

8. Uruchomić terminal (CMD w Windows lub XTERM w Linux) i przejść do katalogu **frontend**
    a. Należy pobrać zależności do projektu **frontend** poleceniem
        npm install
    b. Po pobraniu zależności uruchomić **frontend** poleceniem
        npm start
    c. Domyślnie aplikacja **frontend** uruchomiona jest na porcie **3000**, aby sprawdzić czy powyższe kroki zostały wykonane prawidłowo, należy w przeglądarce wpisać adres
        http://localhost:3000/
    d. Powinniśmy otrzymać wizualną stronę internetową.
9. Pod adresem części **frontend** (http://localhost:3000/) należy kliknąć przycisk **"Zaloguj"** aby utworzyć swoje konto.
10. Jeśli chcemy uprawnienia administracyjne dla swojego konta (pierwszy użytkownik) należy zalogować się do klienta **mysql** i wpisać następujące polecenie
        UPDATE Users SET isAdmin=1 WHERE idUser=1;

Po powyższej operacji powinna pojawić nam się zakładka "Administracja". Dla pozostałych użytkowników ta operacja nie jest konieczna - uprawnienia mogą zostać nadane w zakładce "Administracja" -> "Użytkownicy".

UWAGA: Najpierw należy uruchomić środowisko aplikacji **backend** a następnie **frontend**.

Skrócona instrukcja korzystania z aplikacji

Użytkownik bez uprawnień administracyjnych ma możliwość:

- Zalogowania się używając przycisku "Zaloguj", który jest zintegrowany z portalem społecznościowym Facebook
- Przeglądania apartamentów oraz wyfiltrowania kryteriów takich jak miasto, czas rezerwacji oraz liczba osób
- Przeglądania swoich rezerwacji i możliwość anulowania

Użytkownik z uprawnieniami administracyjnymi ma możliwość:

- Korzystania z funkcjonalności zwykłego użytkownika
- Dostępu do panelu administracyjnego pozwalającego zarządzać
    - apartamentami (dodawanie, przegląd, usuwanie, edycja)
    - rezerwacjami (podglądu rezerwacji, zmiany statusu rezerwacji: anulowania, niepotwierdzona, potwierdzona)
    - użytkownikami (podgląd użytkowników, możliwość nadania lub odebrania praw administratora innym użytkownikom)