**Софийски университет „Св. Климент Охридски“**

Факултет по математика и информатика

Специалност: “Информационни системи ”

Курс: 4, Група: 1

Дисциплина:

*Разработка на клиент-сървър (fullstack) приложения с Node.js + Express.js + React.js, летен семестър 2020/2021”*

**КУРСОВ ПРОЕКТ**

Travel Finder

**Изготвил:**

Кристиан Кременов Кирилов (ФН: 71818)

**Преподавател:**

Траян Илиев

София

Летен семестър 2019/2020

1. **Backend API**

Backend-а е реализиран с MVC патърн, като има стремеж към възможно най-високо ниво на абстракция. За това всяка бизнес логика е опитано да бъде изведена в отделен сървис, същото важи и за всякакви мидълуери, които са били използвани.

Има 6 основни ендпоинта за комуникация с frontend приложението:

* user router
* admin router
* ratings router
* comments router
* favourites router
* destinations router

1. **Админ API**

adminRouter.post('/**createUser'**, authAdmin.isAdmin, adminController.createUser) – създава юзър

adminRouter.post('/**deleteUser'**, authAdmin.isAdmin, adminController.deleteUser); - изтрива юзър

adminRouter.post('/**login'**, authAdmin.isAdmin,

adminController.login'); - логва като админ

adminRouter.delete('/**logout'**, authAdmin.isAdmin,

adminController.logout'); - логаутва админ

adminRouter.put('/**editUser**, authAdmin.isAdmin,

adminController.editUser); - ъпдейт на данните на юзър

adminRouter.get('/**getAllUsers**, authAdmin.isAdmin,

adminController.getAllUsers); - взема всички юзъри

1. **User API**

userRouter.post('/**register'**, userController.register) - регистрация

userRouter.post('/**login'**, userController.login) - логин

userRouter.post('/**logout'**, AuthenticationMiddleware.isAuthenticated, userController.logout) - логаут

userRouter.get('/**currentUser'**, AuthenticationMiddleware.isAuthenticated,

userController.currentUser) - връща юзъра създал текущата сесия

userRouter.put('/**editUser'**, AuthenticationMiddleware.isAuthenticated,

userController.editUser) - ъпдейтва юзър

1. **Comments API**

commentRouter('/**addComment'**)

.post(AuthenticationMiddleware.isAuthenticated, commentController.addComment) - добавяне на коментар

commentRouter('/**deleteComment**/:id')

.delete(AuthenticationMiddleware.isAuthenticated,

commentController.deleteComment) - изтриване на коментар

commentRouter('/**getAllComments**/:id')

.get(AuthenticationMiddleware.isAuthenticated,

commentController.allCommentsForHotel) - fetch-va всички коментари

commentRouter('/**toggleThumbReact'**)

.post(AuthenticationMiddleware.isAuthenticated,

commentController.toggleReactThumb) - слага палец за реакция на коментар

commentRouter('/**togglePartyReact'**)

.post(AuthenticationMiddleware.isAuthenticated,

commentController.toggleReactParty) - слага друг вид реакция на коментар

commentRouter('/**editComment**/:id')

.put(AuthenticationMiddleware.isAuthenticated,

commentController.editComment) - ъпдейтва коментар

1. **Ratings API**

ratingsRouter('/**getAverageRating/:id**')

.get(AuthenticationMiddleware.isAuthenticated,

ratingsController.getAllRatingsForHotel)

- взема рейтинг за посочената дестинация по **{id}**

ratingsRouter('/**addRating/:id/:hotelId**')

.post(AuthenticationMiddleware.isAuthenticated,

ratingsController.addRating)

- добавя рейтинг на дадена дестинация от текущия юзър

ratingsRouter('/**deleteRating/:id/:hotelId**')

.delete(AuthenticationMiddleware.isAuthenticated,

ratingsController.deleteRating)

- изтрива рейтинг за дадена дестинация

ratingsRouter('/**getUserRating/:id/:hotelId**')

.get(AuthenticationMiddleware.isAuthenticated,

ratingsController.getUserRatingForHotel)

- взема рейтинга на юзър за определена дестинация (ако има такъв)

1. **Favourites API**

favouriteRouter('/**getAllFavorites/:id**')

.get(AuthenticationMiddleware.isAuthenticated,

favoritesController.getAllFavorites)

– взима всички фаворити за определен **{userId}**

favouriteRouter('/**addFavorites/:id**')

.post(AuthenticationMiddleware.isAuthenticated,

favoritesController.addFavorite)

– добавя фаворит **{hotelId}** за определен **{userId}**

favouriteRouter('/**deleteFavorites/:id/:hotelId**')

.post(AuthenticationMiddleware.isAuthenticated,

favoritesController.deleteFavorite)

– изтриване на фаворит

favouriteRouter('**isAddedToFavourites/:id/:hotelId**')

.get(AuthenticationMiddleware.isAuthenticated,

favoritesController.isAddedToFavourites)

– проверява дали за определен **{userId}** има запис в базата с **{hotelId}**

1. **Destinations API**

destinationsRouter('/**getHotelById'**)

.get(AuthenticationMiddleware.isAuthenticated,

destinationController.getHotelById)

– приема **{lat, lng, hotelId}** и връща пълен обект с данни за хотела

destinationsRouter ('/**getHotels'**)

.get(AuthenticationMiddleware.isAuthenticated,

destinationController.getHotels)

* Приема **{lat, lng, budgetValue, dateFilter}** и връща всички хотели, които удовлетворяват филтрите и координатите

destinationsRouter ('/**getPointOfInterest'**)

.get(AuthenticationMiddleware.isAuthenticated,

destinationController.getPointOfInterest)

* Приема **{lat, lng}** и връща всички интересни точки около съответната локация

destinationsRouter ('/**getRecommendations'**)

.get(AuthenticationMiddleware.isAuthenticated,

destinationController.getRecommendations)

* Приема **{cityCode}** и връща всички препоръки спрямо подадения код на град. Пример: България - BG

destinationsRouter ('/**getToursAndActivities'**)

.get(AuthenticationMiddleware.isAuthenticated,

destinationController.getToursAndActivities)

* Приема **{lat, lng }** и връща всички турове и евенти, които удовлетворяват координатите

1. **What technologies have been used**

* **For the frontend** – Material ui, SCSS,
* **React, React router dom** – for components and for routing
* **Redux** – for state management
* **Google API’s** - for maps, directions and other
* **For the backend** - ExpressJS, PostgresSQL, Typescript
* **Passport** – for authentication
* **Redis** – as a backend store
* **Bcrypt** – for hashing passwords

1. **Used libraries**

react-places-autocomplete, react-dropzone-uploader, react-ratings, @material-ui, redux, react-spinners, yup, axios, formik, apollo-client, cors, nodemon, express, passport, typeorm, redis

1. **Functional and non-functional requirements**

|  |  |  |
| --- | --- | --- |
| Requirement | Type | Description |
| User can register, login and logout | Functional |  |
| User can view his Profile page | Functional |  |
| User can open Homepage | Functional |  |
| User can open Destinations | Functional |  |
| User can search for locations in the search bar | Functional |  |
| User has 3 search types: Hotels, Points of Interest, Tours | Functional |  |
| User can open a destination and see its details | Functional |  |
| Every destination must display its location on the map | Functional |  |
| User must be able to comment on a destination if logged in | Functional |  |
| User must be able to rate a destination if logged in | Functional |  |
| User must be able to add a destination in favourites list when logged in | Functional |  |
| User must be able to upload photos for a destination and view them | Functional |  |

1. **Frontend realization**

**/src** - съдържа целия сорс код за проекта

/**components** - съдържа всички Реакт компоненти, стилове, логика и помощни функции, които се използват в проекта

/**redux** - съдържа имплементация на redux state management, екшъни, редюсъри, селектори

/**pages** - съдържа всички компоненти от по-високо ниво, които сглобяват отделните страници

1. **Backend realization**

Файлът **index.ts** съдържа инстанцията на самия проект, подготвяа билд схеми, свързване с база данни, отделни билбиотеки и модули, express server

/**routers** - съдържа всички отделни ендпоинти

/**controllers** - съдържа контролери с логика за ендпоинта

/**services** - съдържа методите изпълняващи логиката за всеки ендпоинт спрямо базата данни

/**migrations** - всички миграции направени към базата

/**config** - Passport configuration

/**models** - contains all Typescript exposed types with the relational db design from PostgreSQL