**Документация на сървърната част:**

Сървъра е написан на nodejs с express. За да се стартира, в терминал, отворен в директорията на сървъра (“./server”) напишете npm install и след това npm run start. Този скрипт от своя страна ще стартира сървъра, използвайки nodemon. Nodemon следи за промени по кода (в папката “./server/src”) и рестартира сървъра на всяка промяна. Като стартираща команда се използва “bun run src/index.ts”. Bun е нов и доста бърз all-in-one JavaScript runtime.

**Dependency list** на сървъра:

* Axios – използван за заявки към външни източници (uploadthing)
* Bcrypt – използва се за криптиране или хеширане (в случая за хеширане на пароли на потребители)
* Cors – използва се за забраняване на достъпа от устройства, от които не искаме да могат да достъпват сървъра
* Dotenv – библиотека за използване на environment variables в кода
* Express – библиотеката, използвана за реализация на сървъра
* Jsonwebtoken – използва се заедно с passport-jwt за запазване на сесии на потребители в localstorage/sessionstorage
* Mongodb, mongoose – библиотеки за работа с mongoDB
* Passport + passport-jwt + passport-local – библиотека, използвана за регистриране и логване на потребители, както и за запазване и обновяване на тяхната сесия в сайта.
* Uploadthing – библиотека, направена от <https://uploadthing.com/>, добавяща компоненти за добавяне на снимки и съхранението им и функционалността зад тях.
* Zod – използва се за валидация на данни

**Относно самата реализация на сървъра:**

В index.ts се инициализира сървъра, осъществява се връзка с базата данни (метода за което е дефиниран в друг файл), задава опции на cors, както и опции на сървъра, след което задава на сървъра да използват import-нати route-ове на съответните пътища за тях. След което се стартира сървъра.

rout-овете са дефинирани в `src/routes` и всеки от тях задава специфични методи за манипулиране на съответните ресурси.

authRoutes.ts съдържа rout-ове, свързани с аутентикацияна потребители. Използва `AuthService` за управление на логиката за логване, регистрация и проверка на сесии.

* POST /auth/password: Регистрира или логва потребител в акаунта му.
* GET /auth/profile: Проверява дали потребителя има достъп до това което иска да направи.
* GET /auth/admin: Връща дали потребителя има админски или staff права.

orderRoutes.ts съдържа rout-ове, свързани с поръчките на потребителите. Използва `OrderService` за управление на логиката за създаване, промяна и изтриване на поръчки.

* GET /orders: Връща всички поръчки на потребителя.
* POST /orders: Създава нова поръчка.
* PUT /orders/:id: Обновява съществуваща поръчка.
* DELETE /orders/:id: Изтрива съществуваща поръчка.

productRoutes.ts съдържа rout-ове, свързани с продуктите. Използва `ProductService` за управление на логиката за създаване, промяна и изтриване на продукти.

* GET /products: Връща всички продукти.
* POST /products: Създава нов продукт.
* GET /products/featured: Връща всички продукти флагнати като featured.
* POST /products/featured: Заменя featured продуктите с нови.
* GET /products/:id: Връща продукта с подадено id.
* PUT /products/:id: Обновява съществуващ продукт.
* DELETE /products/:id: Изтрива съществуващ продукт.

userRoutes.ts Този файл съдържа rout-ове, свързани с потребителите. Използва `UserService` за управление на логиката за създаване, промяна и изтриване на потребители.

* GET /users: Връща всички потребители.
* POST /users: Създава нов потребител.
* GET /users/total: Връща броя на всички потребители.
* GET /users/:email: Връща информация за конкретен потребител.
* PATCH /users/:email: Обновява информация за конкретен потребител.
* PUT /users/:email: Обновява информация за конкретен потребител.
* DELETE /users/:email: Изтрива конкретен потребител.

Моделите на таблиците, както и zod обекти за валидация към тях се намират в `src/models` и всеки от тях дефинира структурата на данните за съответния ресурс.

* User.ts Дефинира структурата на данните за потребителите, включително полета като име, имейл, парола и др.
* Order.ts Дефинира структурата на данните за поръчките, включително полета като потребител, продукти, статус на поръчката и др.
* Product.ts Дефинира структурата на данните за продуктите, включително полета като име, цена, описание и др.

В `src/services` се намира бизнес логиката, която съответните rout-ове извикват за достъп до базата данни и операции с нея.

* OrderService.ts Съдържа логика за управление на поръчките, включително методи за създаване, промяна и изтриване на поръчки.
* ProductService.ts Съдържа логика за управление на продуктите, включително методи за създаване, промяна и изтриване на продукти.
* UserService.ts Съдържа логика за управление на потребителите, включително методи за създаване, промяна и изтриване на потребители.

Допълнителни файлове

* `uploadthing.ts`: Съдържа конфигурация и логика за работа с библиотеката `uploadthing` за качване на снимки.
* `config/passport.ts`: Съдържа конфигурация за passport (за локалната стратегия и jwt стратегията).
* `lib`: Съдържа помощни функции за свързване към базата данни.

Link към кода на сървър приложението:

https://github.com/Kaisiq/node-express-react-2024/tree/main/project01/server