САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 2

Выполнил:

Викторова Анастасия

M3220d

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

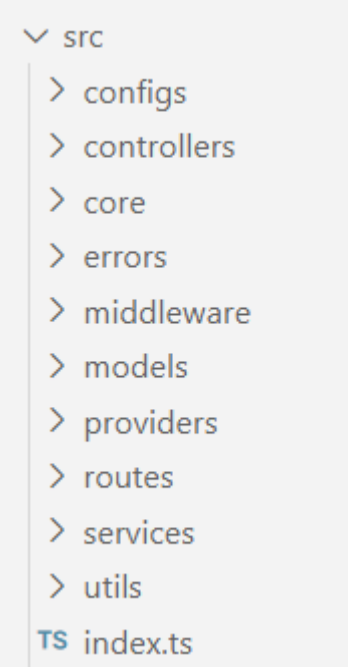
Задача

реализовать RESTful API средствами express + typescript.

Сервис для буккроссинга. Требуемый функционал: регистрация, авторизация, создание профиля, создание списка своих книг, создание заявок на обмен книгами, работа с заявками на обмен.

Ход работы

Была организована структура проекта следующим образом:



Для реализации проекта были созданы следующие модели

1. User – пользователь
   Поля: имя, фамилия, email, пароль,
2. RefreshToken
   Поля: токен, id пользователя
3. Book
   Поля: название, id владелеца, описание
4. ExchangeRequest – заявка на обмен книгами
   Поля: id заявителя, id книги, активна

```
@Table
class User extends Model {
    @Column
    firstname: string

    @Column
    lastname: string

    @Unique
    @Column
    email: string

    @AllowNull(false)
    @Column
    password: string

    @BeforeCreate
    @BeforeUpdate
    static async generatePasswordHash(instance: User) {
        const { password } = instance

        if (instance.changed('password')) {
            instance.password = await hashPassword(password)
        }
    }
}
```

Рисунок 1. Модель User

```
@Table
class RefreshToken extends Model {
    @Unique
    @AllowNull(false)
    @Column
    token: string

    @ForeignKey(() => User)
    @Column
    userId: number
}
```

Рисунок 2. Модель RefreshToken

```
@Table
class Book extends Model {
    @Column
    title: string

    @AllowNull(false)
    @Column({
        type: DataTypes.INTEGER,
        validate: {
            isInt: true
        }
    })
    ownerId: number

    @Column
    description: string
}
```

Рисунок 3. Модель Book

```
3   @Table
4   class ExchangeRequest extends Model {
5       @AllowNull(false)
6       @Column
7       applicantId: number
8
9       @AllowNull(false)
10      @Column
11      bookId: number
12
13      @Default(true)
14      @Column
15      isActive: boolean
16  }
```

Рисунок 4. Модель ExchangeRequest

На примере Book и User рассмотрим некоторые из роутов.
Для User интересно рассмотреть следующие

src > routes > user > TS User.ts > ...

```
27  // получить список книг пользователя
28  router.route('/books/:id')
29      .get(controller.bookList)
30
31  // получить список заявок пользователя
32  router.route('/exchangeRequests/:id')
33      .get(controller.exchangeRequestList)
```

Контроллер с методом bookList

src > controllers > user > TS User.ts > UserController

```
17  class UserController {
124     bookList = async (request: any, response: any) => {
125         try {
126             const bookService = new BookService()
127             const books: Book[] | BookError = await bookService.getByOwner(Number(request.params.id))
128
129             response.send(books)
130         } catch (error: any) {
131             response.status(404).send({ "error": error.message })
132         }
133     }
```

Сервис с методом getByOwner

src > services > book > TS Book.ts > BookService > create

```
4   class BookService {
29      async getByOwner(ownerId: number) : Promise<Book[]|BookError> {
30          const books = await Book.findAll({ where: { "ownerId": ownerId } })
31
32          if (books) return books
33
34          throw new BookError('Not found!')
35      }
```

Для Book (аналогично для ExchangeRequest) реализованы CRUD методы

src > routes > book > TS Book.ts > ...

```
8   // создание книги
9   router.route('/')
10      .post(controller.create)
11
12  // получить все
13  router.route('/')
14      .get(controller.getAll)
15
16  // получить по id
17  router.route('/:id')
18      .get(controller.getById)
19
20  // обновить
21  router.route('/:id')
22      .put(controller.update)
23
24  // удалить
25  router.route('/:id')
26      .delete(controller.delete)
```

Вывод

В ходе работы было реализовано RESTful API средствами express + typescript для платформы буккроссинга.