Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет Инфокоммуникационных технологий

Образовательная программа 09.03.03 Мобильные и сетевые технологии

О Т Ч Е Т

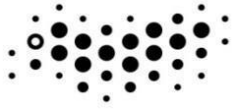
по практической работе № 2
по дисциплине «**Бэк-энд разработка**»

Выполнил:

Мосин З. И.

Проверил:

Добряков Д. И.



Санкт-Петербург, 2024

Цели работы

- Продумать свою собственную модель пользователя
- Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize
- Написать запрос для получения пользователя по id/email

Ход работы

Модель пользователя в выбранном сервисе должна содержать данные об аккаунте (логин и пароль), а также почта, которая должна быть уникальной у каждого пользователя.

```
@Table
class User extends Model {
    @PrimaryKey
    @AutoIncrement
    @Column(DataType.INTEGER)
    declare id: number

    @Column(DataType.STRING)
    declare name: string

    @Unique
    @Column(DataType.STRING)
    declare email: string

    @Unique
    @Column(DataType.STRING)
    declare password: string
}
```

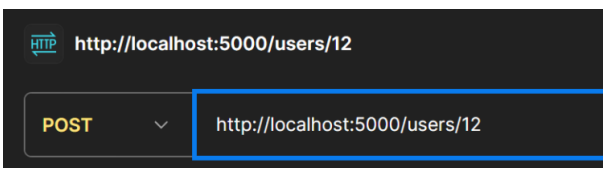
Далее создаем Controller для пользователя, подсоединяя методы из соответствующего сервиса.

```ts
import { Request, Response } from 'express'
import jwt, { JwtPayload } from 'jsonwebtoken'

import UserService from '../services/UserService'
import handleError from '../utils/handleError'

export default {
    async getUserById(req: Request, res: Response) {
    },

    async getAllUsers(req: Request, res: Response) {
    },

    async createUser(req: Request, res: Response) {
    },

    async updateUser(req: Request, res: Response) {
    },

    async deleteUser(req: Request, res: Response) {
    },

    async verify(req: Request, res: Response) {
    },
}

```

Внутри сервиса описываем методы взаимодействия с пользователем, используя паттерн Репозиторий для прослойки между сервисом и моделью.

```ts
import sequelize from '../database/index'
import User from '../database/models/User'
import serviceHandleError from '../utils/serviceHandleError'

const userRepository = sequelize.getRepository(User)

class UserService {
    static async getUserById(id: number) {
    }

    static async getAllUsers() {
    }

    static async createUser(userData: any) {
    }

    static async updateUser(id: number, userData: any) {
    }

    static async deleteUser(id: number) {
    }
}

export default UserService

```

В итоге получаем удобное описание всех свойств модели пользователя и взаимодействия с ним. Для создания запроса, обращаемся к эндпоинту users/:id с методом get, тем самым получая данные о пользователе по его id



ВЫВОД

В процессе выполнения продумали собственную модель пользователя, поработали с express и Sequelize, получили опыт составления RESTful API для модели пользователя