

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ

по Домашней работе № 2

Специальность:

09.03.03 Мобильные и сетевые технологии

Проверил:

Добряков Д. И. _____

Дата: «__» _____ 202__ г.

Оценка _____

Выполнил:

студенты группы К33401

Ковалев В. М.

Санкт-Петербург

2023

ЦЕЛЬ 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

ВЫПОЛНЕНИЕ

Модель пользователя:

User		
id	Int	autoincrement()
email	String	
password	String	
name	String?	
● JWTAccess	JWTAccess?	

```
model User {  
  id      Int      @id @default(autoincrement())  
  email   String   @unique  
  password String  
  name    String?  
  JWTAccess JWTAccess?  
}
```

Набор CRUD-операций:

```
register = async (request: Request, response: Response) => {
  try {
    const {email, name, password} = request.body
    console.log(email, name, password)
    const user = await this.userService.createUser( user: {email, name, password})
    return response.json( body: {id:user.id, email: user.email, name: user.name})
  } catch (e:any) {
    return response.status( code: 404).json( body: {"error": e.message})
  }
}

get = async (request: Request, response: Response) => {
  try {
    const user = await this.userService.getById(Number(request.params.id))
    return response.json( body: {id:user.id, email: user.email, name: user.name})
  } catch (e:any) {
    return response.status( code: 404).json( body: { "error": e.name })
  }
}

deleteUser = async (request: Request, response: Response) => {
  try {
    const user = await this.userService.deleteUser(Number(request.params.id))
    return response.json( body: {id:user.id, email: user.email, name: user.name})
  } catch (e:any) {
    return response.status( code: 404).json( body: { "error": e.name })
  }
}

changeName = async (request: Request, response: Response) => {
  try {
    if(request.body.name){
      const user = await this.userService.changeName(Number(request.params.id), String(request.body.name))
      return response.json( body: {id:user.id, email: user.email, name: user.name})
    }
    return response.status( code: 403).json( body: {"error": "name field is required"})
  } catch (e:any) {
    return response.status( code: 404).json( body: { "error": e.name })
  }
}
```

Получение по ID:

GET ▼ `{{url}}/auth/users/3`

Params Headers Body

Query Params

Key
Key

Body Headers (7)

Pretty Raw Preview JSON ▼

```
1 {
2   "id": 3,
3   "email": "admin@gmail.com",
4   "name": "Ezio"
5 }
```

Получение по email:

GET ▼ `{{url}}/auth/users?email=admin@gmail.com`

Params ● Authorization Headers (7) Body Pre-req

Query Params

Key
<input checked="" type="checkbox"/> email
Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 [
2   {
3     "id": 3,
4     "email": "admin@gmail.com",
5     "password": "admin",
6     "name": "Ezio"
7   }
8 ]
```

ВЫВОД

В результате выполнения домашней работы я получил модель пользователя и CRUD-операции для работы с ней.