

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа

Выполнил:

Олейникова Полина

Группа К33402

Проверил:

Добряков Д. И.

Санкт-Петербург

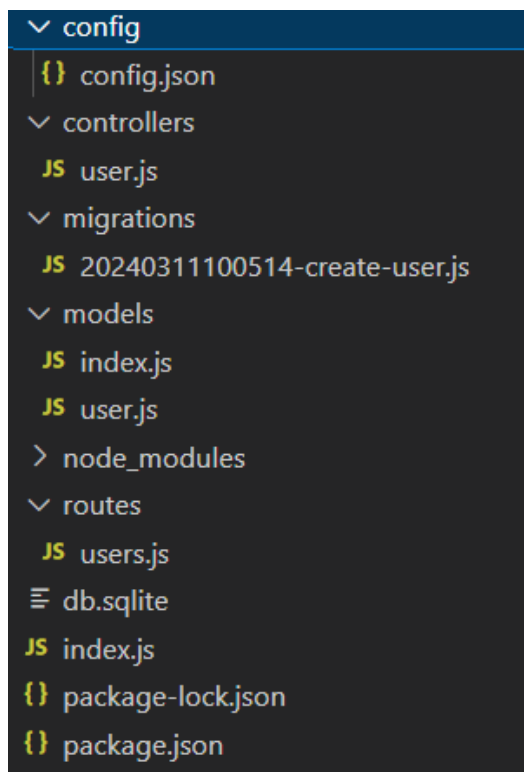
2024 г.

Задача

- 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

Структура папок:



Запуск сервера и определение маршрута для пользователей

```
const express = require('express')
const users = require('./routes/users');

const app = express()
app.use(express.json());

const port = 3000
app.use('/api/users', users);

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Определение маршрутов к методам с пользователями

```
JS users.js X
1  const express = require('express');
2  const router = express.Router();
3  const User = require('../controllers/user');
4
5  router.post('/', User.createUser);
6  router.get('/', User.getUsers);
7  router.get('/:id', User.getUser);
8  router.patch('/:id', User.updateUser);
9  router.delete('/:id', User.deleteUser);
10
11 module.exports = router;
```

Создания пользователя

```
exports.createUser = async (req, res) => {
  try {
    const user = await db.User.create(req.body);
    return res.status(201).send(user);
  } catch (error) {
    return res.status(500).json({ message: error.message });
  }
}
```

POST http://localhost:3000/api/users

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON

```
1 {
2   "firstName": "Polina",
3   "lastName": "Oleynikova",
4   "email": "example@gmail.com"
5 }
```

Body Cookies Headers (7) Test Results

Status: 201 Created

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "firstName": "Polina",
4   "lastName": "Oleynikova",
5   "email": "example@gmail.com",
6   "updatedAt": "2024-03-11T11:26:38.337Z",
7   "createdAt": "2024-03-11T11:26:38.337Z"
8 }
```

Получение пользователей\пользователя

```
exports.getUsers = async (req, res) => {
  const users = await db.User.findAll()
  return res.send(users)
}

exports.getUser = async (req, res) => {
  const user = await db.User.findByPk(req.params.id)
  if (user) {
    return res.send(user)
  }
  return res.status(404).send({ message: "user is not found" })
}
```

GET http://localhost:3000/api/users

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "firstName": "Polina",
4   "lastName": "Oleynikova",
5   "email": "example@gmail.com",
6   "createdAt": "2024-03-11T11:26:38.337Z",
7   "updatedAt": "2024-03-11T11:26:38.337Z"
8 }
```

GET http://localhost:3000/api/users/4

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

ody Cookies Headers (7) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "firstName": "Polina",
4   "lastName": "Oleynikova",
5   "email": "example@gmail.com",
6   "createdAt": "2024-03-11T11:26:38.337Z",
7   "updatedAt": "2024-03-11T11:26:38.337Z"
8 }
```

Изменение пользователя

```
exports.updateUser = async (req, res) => {
  const user = await db.User.findPk(req.params.id);
  if (!user) {
    return res.status(404).send({ message: "user is not found" });
  }

  try {
    const updatedUser = await user.update(req.body);
    return res.send(updatedUser);
  } catch (e) {
    return res.status(500).json({ message: error.message });
  }
}
```

The screenshot shows a REST client interface with a PATCH request to `http://localhost:3000/api/users/4`. The request body is a JSON object: `{ "firstName": "Polina", "lastName": "Oleynikova", "email": "example@gmail.com" }`. The response is a 200 OK status with a JSON body: `{ "id": 4, "firstName": "Polina", "lastName": "Oleynikova", "email": "example@gmail.com", "createdAt": "2024-03-11T11:26:38.337Z", "updatedAt": "2024-03-11T11:26:38.337Z" }`.

Удаление пользователя

```
exports.deleteUser = async (req, res) => {
  const user = await db.User.destroy({
    where: {
      id: req.params.id
    }
  });
  if (user) {
    return res.send({ message: "user deleted" });
  }
  return res.status(404).send({ message: "user is not found" });
}
```

DELETE

▼

http://localhost:3000/api/users/2

ParamsAuthorizationHeaders (8)Body ●Pre-request ScriptTestsSettings

Query Params

	Key	Value
	Key	Value

bodyCookiesHeaders (7)Test Results

Status: 200 OK

PrettyRawPreviewVisualizeJSON ▼

1

2

3

```
{  "message": "user deleted"}
```

Вывод

В ходе данной работы был реализован набор из CRUD-методов для работы с пользователями средствами Express + Sequelize.