САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Backend development

Отчет по практической работе

Выполнил:
Чуб Илья Евгеньевич

Группа:
K33401

Проверил:
Добряков Давид Ильич

Санкт-Петербург
2023 г.

Задачи

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

Ход работы

1. Инициализация проекта и установка зависимостей

npm init

npm i express -S

npm i sequelize -S

npm i sqlite3 -S

npm i sequelize-cli -D

npx sequelize init

2. Модель пользователя
   - **username** : String
   - **email** : String
   - **password** : String

npx sequelize-cli model:generate --name User --attributes username:string,email:string,password:string

npx sequelize-cli db:migrate

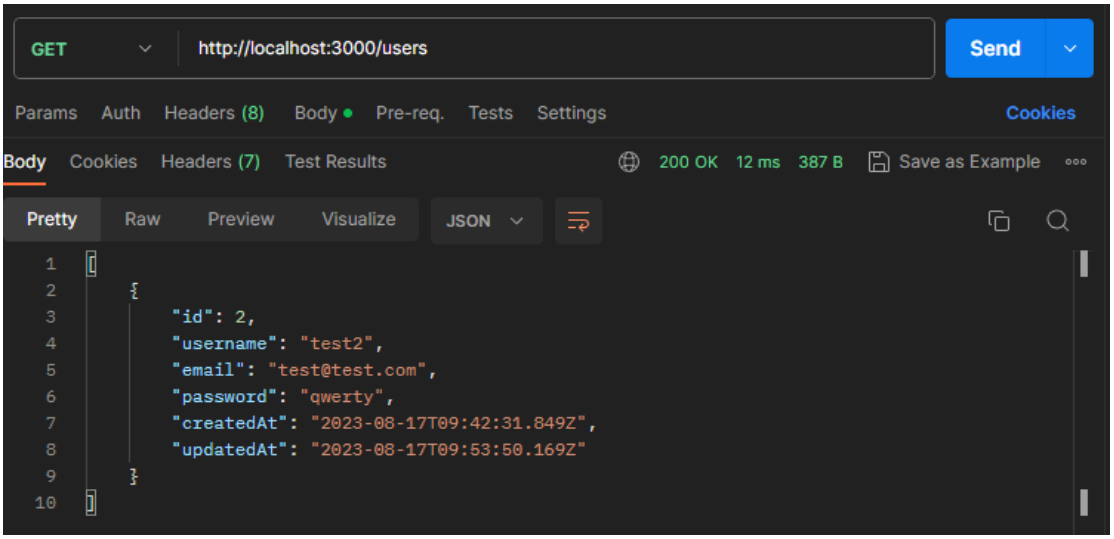
3. CRUD-методы
   3.1. Получение всех пользователей (с возможностью поиска по email) и создание нового пользователя

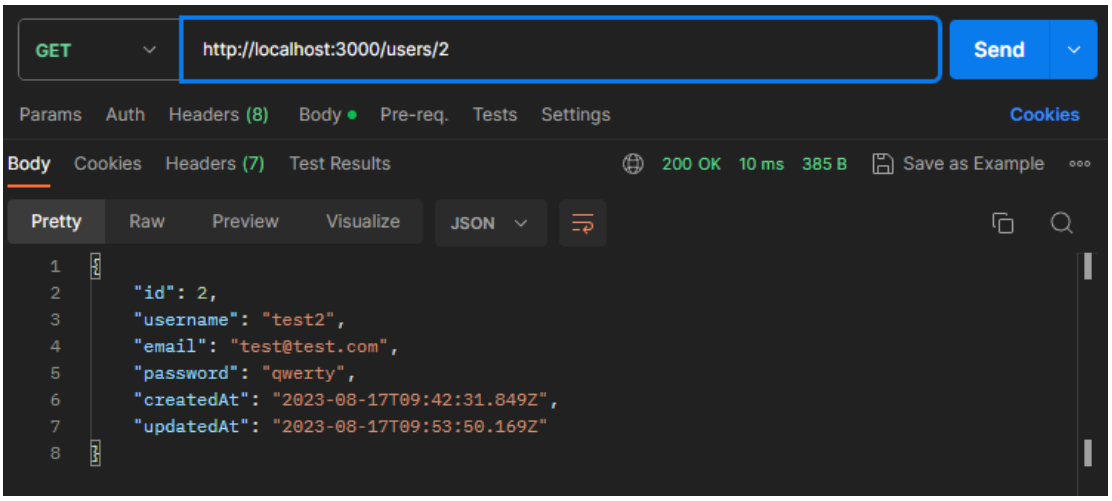
```
app.route( prefix: "/users/")
    .get( handlers: async (req, res) => {
        let users
        if (req.query && req.query.email) {
            users = await db.User.findAll( options: {where: {email: req.query.email}})
        } else {
            users = await db.User.findAll()
        }
        return res.send(users)
    })
    .post( handlers: async (req, res) => {
        try {
            const user = await db.User.create(req.body);
            return res.send({"status": 201, "statusText": "Successfully created " + user.username})
        } catch (e) {
            return res.send({"msg": e})
        }
    })
```

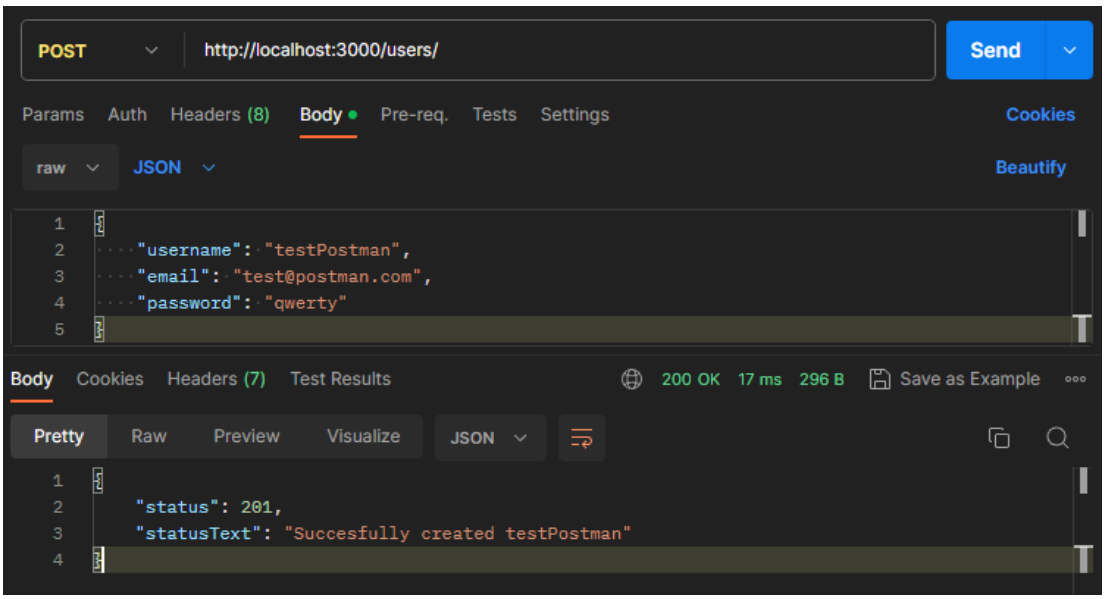
3.2. Получение, удаление и обновление пользователя по id

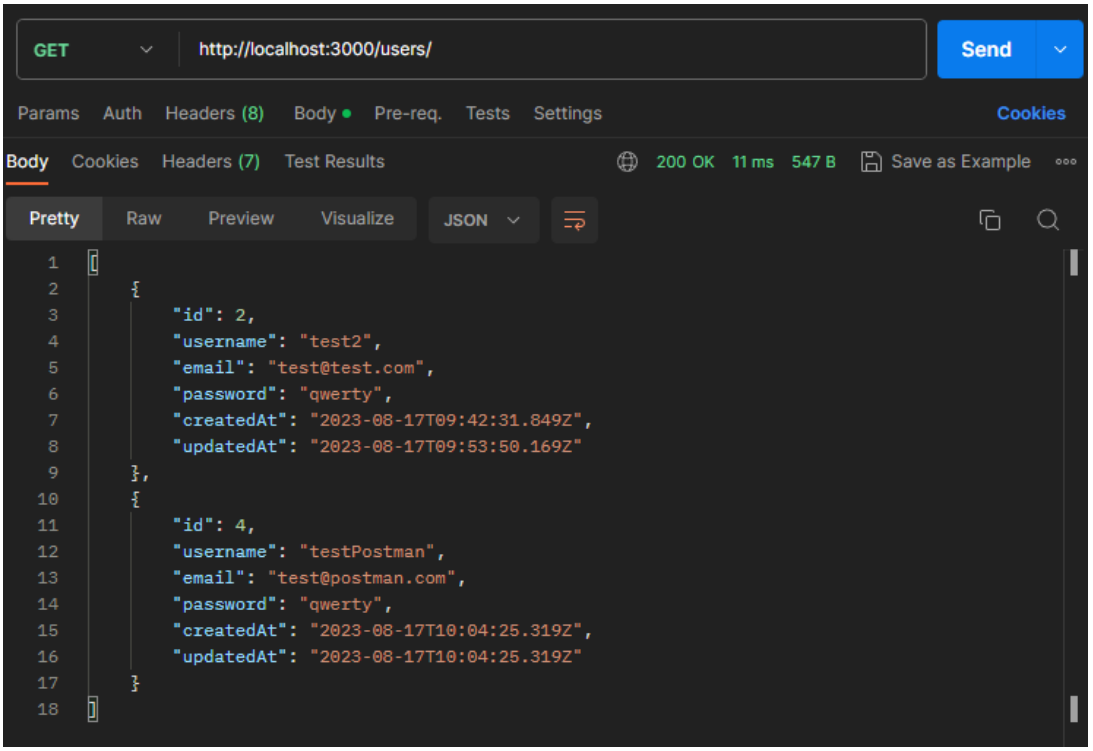
```
app.route( prefix: "/users/:id")
    .get( handlers: async (req, res) => {
        const user = await db.User.findByPk(req.params.id)
        if (user) {
            return res.send(user.toJSON())
        }
        return res.send({"status": 404, "statusText": `User with id=${req.params.id} does not exist`})
    })
    .delete( handlers: async (req, res) => {
        const user = await db.User.destroy({where: {id: req.params.id}})
        if (user) {
            return res.send({"status": 200, "statusText": `User with id=${req.params.id} deleted`})
        }
        return res.send({"status": 204, "statusText": `User with id=${req.params.id} does not exist`})
    })
    .put( handlers: async (req, res) => {
        const user = await db.User.findByPk(req.params.id);
        if (user) {
            try {
                user.update(req.body, {where: {id: req.params.id}});
                return res.send({"msg": "Successfully updated!"})
            } catch (e) {
                return res.send({"msg": e})
            }
        }
        return res.send({"msg": "User is not found"})
    })
```

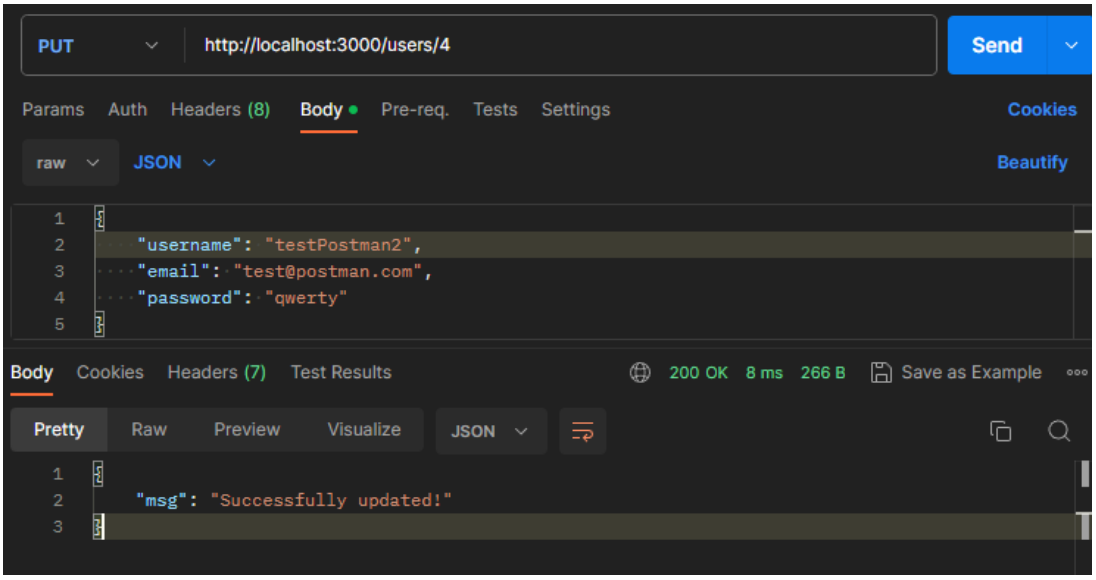
4. Проверка в Postman

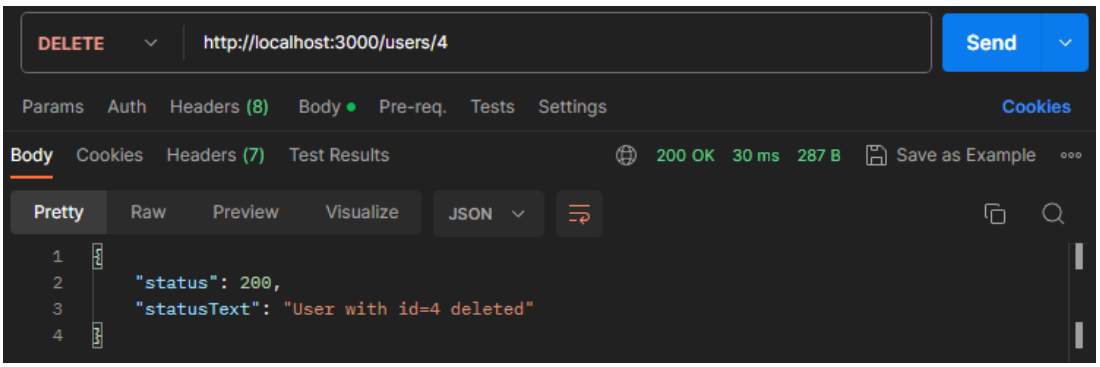

GET
http://localhost:3000/users
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body Cookies Headers (7) Test Results
200 OK 12 ms 387 B
Save as Example
Pretty Raw Preview Visualize JSON
[
{
"id": 2,
"username": "test2",
"email": "test@test.com",
"password": "qwerty",
"createdAt": "2023-08-17T09:42:31.849Z",
"updatedAt": "2023-08-17T09:53:50.169Z"
}
]


GET
http://localhost:3000/users/2
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body Cookies Headers (7) Test Results
200 OK 10 ms 385 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"id": 2,
"username": "test2",
"email": "test@test.com",
"password": "qwerty",
"createdAt": "2023-08-17T09:42:31.849Z",
"updatedAt": "2023-08-17T09:53:50.169Z"
}


POST
http://localhost:3000/users/
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
{
"username": "testPostman",
"email": "test@postman.com",
"password": "qwerty"
}
Body Cookies Headers (7) Test Results
200 OK 17 ms 296 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"status": 201,
"statusText": "Succesfully created testPostman"
}


GET
http://localhost:3000/users/
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body Cookies Headers (7) Test Results
200 OK 11 ms 547 B
Save as Example
Pretty Raw Preview Visualize JSON
[
{
"id": 2,
"username": "test2",
"email": "test@test.com",
"password": "qwerty",
"createdAt": "2023-08-17T09:42:31.849Z",
"updatedAt": "2023-08-17T09:53:50.169Z"
},
{
"id": 4,
"username": "testPostman",
"email": "test@postman.com",
"password": "qwerty",
"createdAt": "2023-08-17T10:04:25.319Z",
"updatedAt": "2023-08-17T10:04:25.319Z"
}
]


PUT
http://localhost:3000/users/4
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
{
"username": "testPostman2",
"email": "test@postman.com",
"password": "qwerty"
}
Body Cookies Headers (7) Test Results
200 OK 8 ms 266 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"msg": "Successfully updated!"
}


DELETE
http://localhost:3000/users/4
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
Cookies
Body Cookies Headers (7) Test Results
200 OK 30 ms 287 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"status": 200,
"statusText": "User with id=4 deleted"
}

Вывод

В ходе выполненной работы были выполнены все поставленные задачи. Тем самым был получен опыт с Express и Sequelize. Создана модель пользователя и реализованы CRUD операции.