САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:
Куцало Александр
Группа
M3221d

Проверил:
Добряков Д. И.

Санкт-Петербург

Задача

Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

- модели
- контроллеры
- роуты
- сервисы для работы с моделями (реализуем паттерн “репозиторий”)

Ход работы

В рамках выполнения работы была сделана следующая файловая структура, соответствующая требованиям:

```
.
|-- Makefile
|-- nodemon.json
|-- package-lock.json
|-- package.json
|-- src
|   |-- config
|   |   `-- config.ts
|   |-- controllers
|   |   `-- user.ts
|   |-- errors
|   |   `-- user_errors.ts
|   |-- index.ts
|   |-- instances
|   |   `-- db.ts
|   |-- middleware
|   |   `-- auth_middleware.ts
|   |-- models
|   |   `-- user.ts
|   |-- routers
|   |   |-- router.ts
|   |   `-- users.ts
|   |-- services
|   |   `-- user.ts
|   `-- utility
|       |-- create_token.ts
|       |-- decode_token.ts
|       `-- password_check.ts
`-- tsconfig.json
```

Для хранения данных о пользователях и обработки запросов связанных с ними был использован Sequelize:

```
exports.createUser = async (req, res) => {
    try {
        const newUser = await db.User.create(req.body);
        return res.status(201).json(newUser);
      } catch (error) {
        console.error('Error creating user:', error);
        return res.status(500).json({ error: 'Internal Server Error' });
      }

}
```

```
exports.getUserById = async (req, res) => {
    console.log("here");
    const user = await db.User.findByPk(req.params.id);
    if (user) {
        return res.send(user);
    };
    return res.status(404).send({ message: `User with id of ${req.params.id} is not found` });
}
```

```
exports.updateUserById = async (req, res) => {
    const user = await db.User.findByPk(req.params.id);
    if (!user) {
        return res.status(404).send({ message: `User with id of ${req.params.id} 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 });
    };
}
```

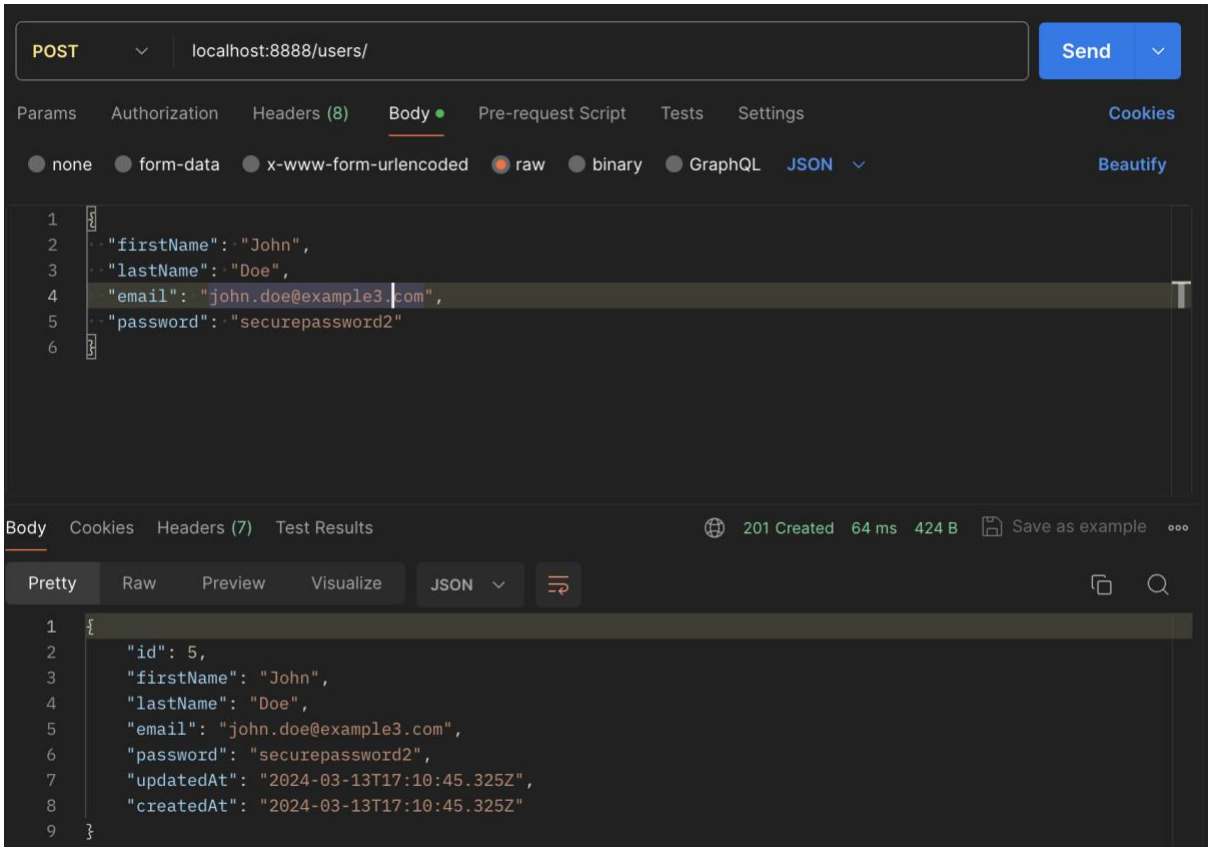
```
exports.deleteUserById = async (req, res) => {
    const user = await db.User.destroy({
        where: {
            id: req.params.id
        }
    });
    if (user) {
        return res.send({ message: `User with id ${req.params.id} is deleted` })
    };
    return res.status(404).send({ message: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

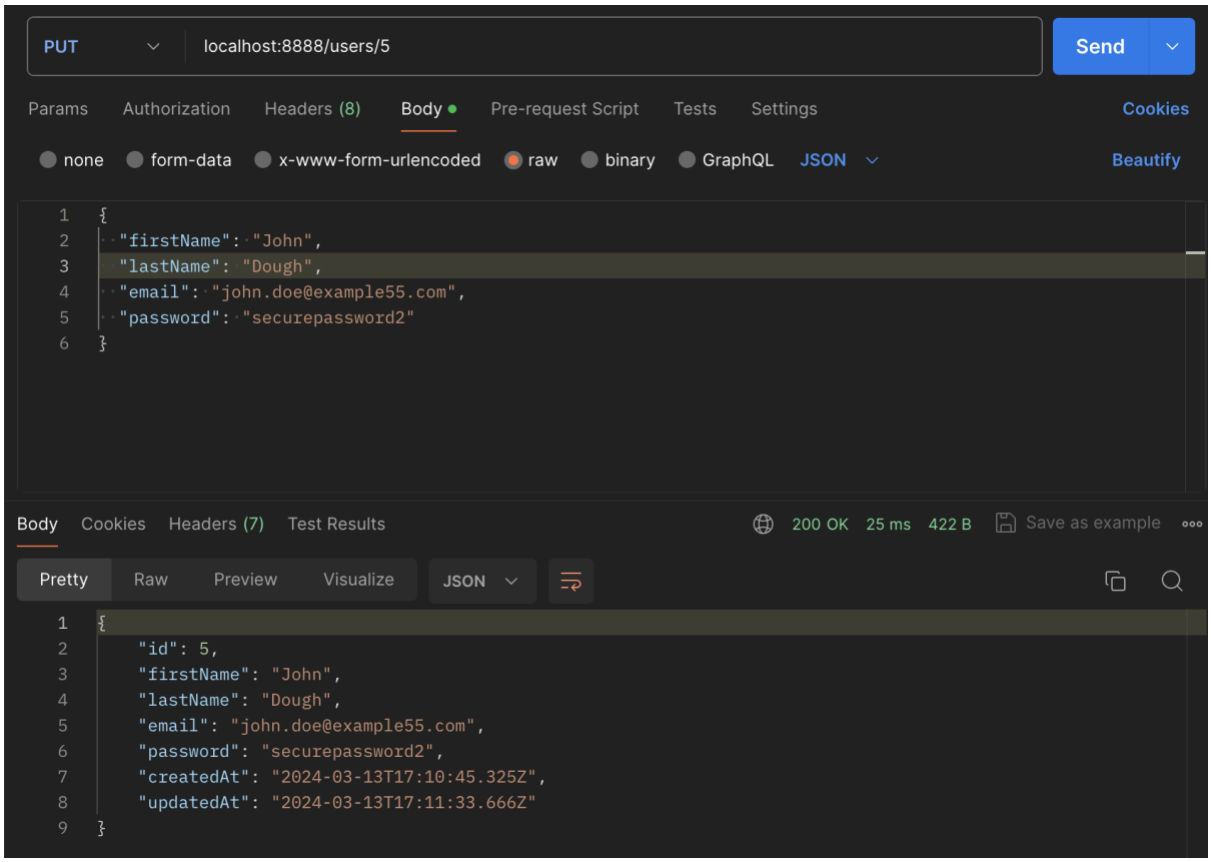
Для роутинга, обработки запросов был использован express JS:

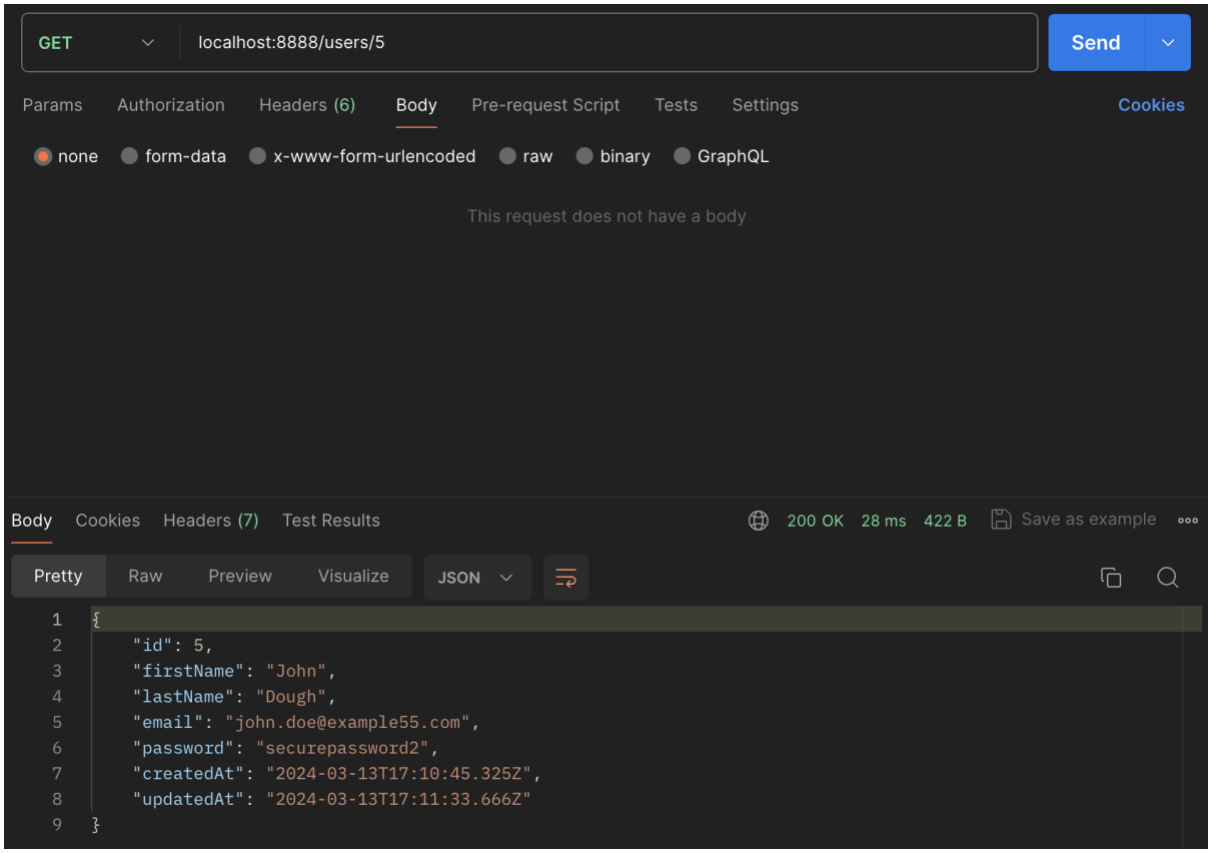
```
const express = require('express');
const router = express.Router();

const user = require("../controllers/user");
console.log("at routes");
router.post('/users/', user.createUser);
router.get('/users/:id', user.getUserById);
router.put('/users/:id', user.updateUserById);
router.delete('/users/:id', user.deleteUserById);
router.get('/users/', user.getUsers);
module.exports = router;
```

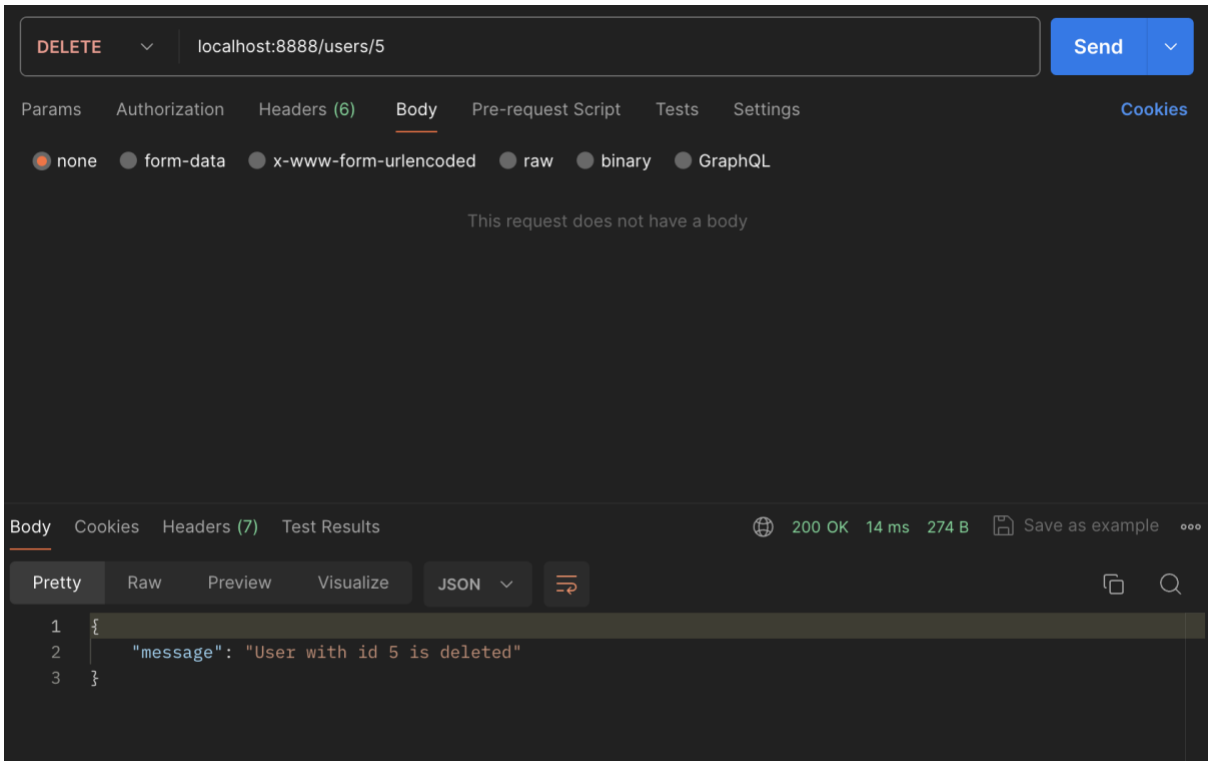
```
const express = require('express')
const users = require('./routes/user');
const port = 8888
const app = express()
app.use(express.json());

app.use('/', users);

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

Тестирование работоспособности с помощью postman:


POST
localhost:8888/users/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"firstName": "John",
"lastName": "Doe",
"email": "john.doe@example3.com",
"password": "securepassword2"
}
Body
Cookies
Headers (7)
Test Results
201 Created
64 ms
424 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 5,
"firstName": "John",
"lastName": "Doe",
"email": "john.doe@example3.com",
"password": "securepassword2",
"updatedAt": "2024-03-13T17:10:45.325Z",
"createdAt": "2024-03-13T17:10:45.325Z"
}


PUT
localhost:8888/users/5
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"firstName": "John",
"lastName": "Dough",
"email": "john.doe@example55.com",
"password": "securepassword2"
}
Body
Cookies
Headers (7)
Test Results
200 OK
25 ms
422 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 5,
"firstName": "John",
"lastName": "Dough",
"email": "john.doe@example55.com",
"password": "securepassword2",
"createdAt": "2024-03-13T17:10:45.325Z",
"updatedAt": "2024-03-13T17:11:33.666Z"
}


GET
localhost:8888/users/5
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (7)
Test Results
200 OK
28 ms
422 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 5,
"firstName": "John",
"lastName": "Dough",
"email": "john.doe@example55.com",
"password": "securepassword2",
"createdAt": "2024-03-13T17:10:45.325Z",
"updatedAt": "2024-03-13T17:11:33.666Z"
}


DELETE
localhost:8888/users/5
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (7)
Test Results
200 OK
14 ms
274 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"message": "User with id 5 is deleted"
}

Вывод

В ходе выполнения данной работы был создан boilerplate сервис с помощью Express JS и Sequelize