

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа № 2

Выполнил:

Казанков Илья К33402

Проверил: Добряков Д. И.

Санкт-Петербург

2024 г.



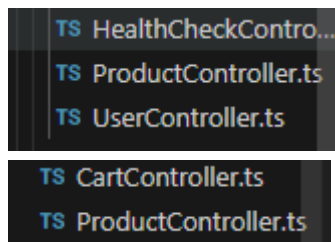
## Задача

- В рамках данной лабораторной работы Вам предложено выбрать один из нескольких вариантов. Выбранный вариант останется единым на весь курс и будет использоваться в последующих лабораторных работах.
- По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

## Ход работы

Для данной работы мною был выбран вариант с сервисом для буккроссинга.

Далее приведены созданные мною модели и Controller-ы для работы с данными:



Пользуясь уже реализованной частью создания моделей данных в boilerplate, создал оставшиеся модели данных, всё также используя библиотеку sequelize-typescript и декораторы.

```
src > database > TS index.ts > ...
1  import { Sequelize } from 'sequelize-typescript'
2
3  import User from './models/User'
4
5  const sequelize = new Sequelize({
6    database: process.env.DB_NAME || 'database123',
7    username: process.env.DB_USER || 'username123',
8    password: process.env.DB_PASSWORD || 'password123',
9    dialect: 'postgres',
10   host: process.env.DB_HOST || 'localhost',
11   port: Number(process.env.DB_PORT || '5432'),
12   repositoryMode: true,
13   logging: console.log,
14 })
15
16 sequelize.addModels([User])
17
18 sequelize.sync().then(() => {
19   console.log('sync sequelize')
20 })
```

Все созданные способы работы с данными подключаются к нашему серверу по соответствующим эндпоинтам, подключенным через общий Router.

```
const app = express();
const sequelize = new Sequelize({
  dialect: 'postgres',
  host: 'localhost',
  username: 'postgres',
  password: 'admin',
  database: 'postgres',
  models: [User, Product, Sale]
});

app.use(bodyParser.json());

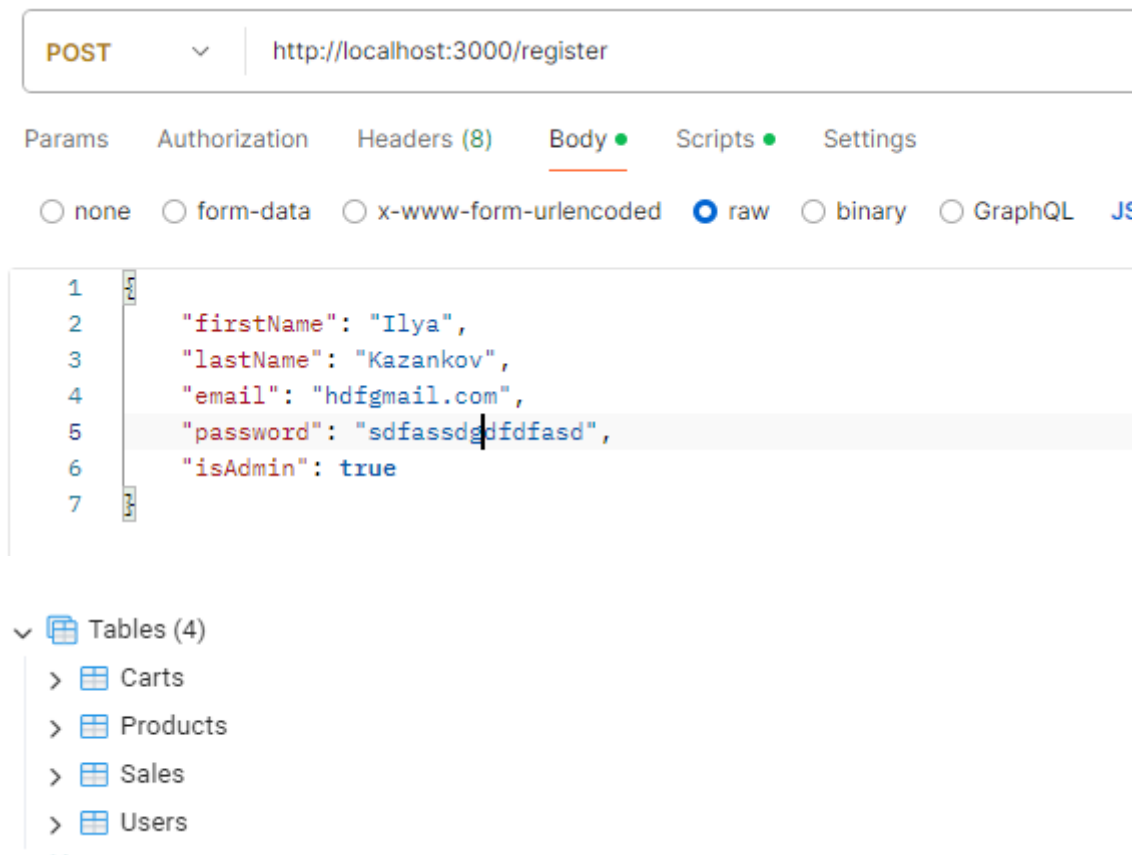
const userController = new UserController();

app.post('/register', userController.register);
app.post('/login', userController.login);
app.get('/health-check', (req, res) => HealthCheckController.check(req, res));

sequelize.sync().then(() => {
  app.listen(3000, () => {
    console.log('Server is running on port 3000');
  });
});

export default sequelize
```

В итоге при отправке запросов через Postman, наш сервер взаимодействует с выбранной базой данных PostgreSQL, создавая и изменяя выбранные нами записи.



## Вывод

В процессе работы создали RESTful API, используя express и typescript. Подключили несколько моделей данных, поработали с данными через отправку тестовых запросов.