**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина: Б**ек-энд разработка

Отчет

## Лабораторная работа №3: Микросервисы

Выполнила:

Пронина Александра

K33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

1.Создание проекта:

Я начала с создания нового каталога для микросервиса управления книгами. Перешла в этот каталог и инициализировала новый проект Node.js с помощью npm init -y.

**mkdir book-management-service**

**cd book-management-service**

**npm init -y**

2. Установка зависимостей:

Для работы моего микросервиса мне понадобились различные пакеты, такие как Express для создания сервера, Sequelize для работы с базой данных и bcryptjs для безопасного хранения паролей. Я установила их, используя команду npm install express sequelize sqlite3 sequelize-cli bcryptjs.

**npm install express sequelize sqlite3 sequelize-cli bcryptjs**

3. Настройка базы данных:

Сначала я инициализировала Sequelize с помощью npx sequelize-cli init, а затем создала модели для книги и пользователя с помощью команд npx sequelize-cli model:generate --name Book --attributes title:string,author:string,description:text и npx sequelize-cli model:generate --name User --attributes username:string,email:string,password:string. После этого я выполнила миграции для создания таблиц в базе данных с помощью npx sequelize-cli db:migrate.

**npx sequelize-cli init**

**npx sequelize-cli model:generate --name Book --attributes title:string,author:string,description:text**

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

**npx sequelize-cli db:migrate**

4. Настройка маршрутов:

Я создала маршруты для регистрации, входа и управления книгами. Для этого я написала код в файлах routes/auth.js и routes/books.js, который отвечает за соответствующие операции.

// routes/auth.js

const express = require('express');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const { User } = require('../models');

const router = express.Router();

router.post('/register', async (req, res) => {

try {

const { username, email, password } = req.body;

const hashedPassword = await bcrypt.hash(password, 10);

const user = await User.create({ username, email, password: hashedPassword });

res.status(201).json(user);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

router.post('/login', async (req, res) => {

try {

const { email, password } = req.body;

const user = await User.findOne({ where: { email } });

if (!user) {

throw new Error('User not found');

}

const isValidPassword = await bcrypt.compare(password, user.password);

if (!isValidPassword) {

throw new Error('Invalid password');

}

const token = jwt.sign({ userId: user.id }, 'secretkey', { expiresIn: '1h' });

res.json({ token });

} catch (error) {

res.status(401).json({ message: error.message });

}

});

module.exports = router;

// routes/books.js

const express = require('express');

const { Book } = require('../models');

const router = express.Router();

router.post('/', async (req, res) => {

try {

const { title, author, description } = req.body;

const book = await Book.create({ title, author, description });

res.status(201).json(book);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

module.exports = router;

5. Настройка сервера:

Чтобы запустить сервер, я создала файл server.js, где инициализировала Express приложение и подключила маршруты для регистрации, входа и управления книгами.

// server.js

const express = require('express');

const authRoutes = require('./routes/auth');

const bookRoutes = require('./routes/books');

const app = express();

app.use(express.json());

app.use('/api/auth', authRoutes);

app.use('/api/books', bookRoutes);

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

6. Запуск микросервиса:

И, наконец, чтобы запустить микросервис, я воспользовалась командой node server.js.