**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:

Пронина Александра

К33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

**Задача:**

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. Начнем с создания нового проекта Node.js и инициализации его с помощью npm. Запусти команду:

npm init

2. После инициализации проекта установим Express:

npm install express

3. Теперь нам нужно создать и настроить модель пользователя с помощью Sequelize. Выполним следующие команды:

npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string

4. Установим SQLite3 для работы с базой данных:

npm i sqlite3 -S

5. Затем сделаем миграцию базы данных:

npx sequelize db: migrate

После базовой настройки Express и Sequelize для работы с пользователями, нужно реализовать CRUD-методы и запросы для получения пользователя по id/email. Сначала реализуем CRUD-методы.

Ниже примеры кода для каждого метода:

// Создание нового пользователя

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

try {

const user = await User.create(req.body);

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

} catch (err) {

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

}

});

// Получение всех пользователей

app.get ('/users', async (req, res) => {

try {

const users = await User.findAll();

res.json(users);

} catch (err) {

res.status(500).json({ error: err.message });

}

});

// Получение пользователя по id

app.get('/users/:id', async (req, res) => {

try {

const user = await User.findByPk(req.params.id);

if (user) {

res.json(user);

} else {

res.status(404).json({ error: 'User not found' });

}

} catch (err) {

res.status(500).json({ error: err.message });

}

});

// Обновление пользователя

app.put('/users/:id', async (req, res) => {

try {

const [updated] = await User.update(req.body, {

where: { id: req.params.id }

});

if (updated) {

const updatedUser = await User.findByPk(req.params.id);

res.json({ updated: updatedUser });

} else {

res.status(404).json({ error: 'User not found' });

}

} catch (err) {

res.status(500).json({ error: err.message });

}

});

// Удаление пользователя

app.delete('/users/:id', async (req, res) => {

try {

const deleted = await User.destroy({

where: { id: req.params.id }

});

if (deleted) {

res.json({ deleted: true });

} else {

res.status(404).json({ error: 'User not found' });

}

} catch (err) {

res.status(500).json({ error: err.message });

}

});

Теперь для запроса пользователя по id/email:

app.get('/users/:idOrEmail', async (req, res) => {

try {

const user = await User.findOne({

where: {

[Sequelize.Op.or]: [

{ id: req.params.idOrEmail },

{ email: req.params.idOrEmail }

]

}

});

if (user) {

res.json(user);

} else {

res.status(404).json({ error: 'User not found' });

}

} catch (err) {

res.status(500).json({ error: err.message });

}

});

**Вывод:** В данной работе мы создали проект на Node.js с использованием Express для создания веб-приложения и Sequelize для работы с базой данных. Мы инициализировали проект с помощью npm, установили Express и настроили модель пользователя с помощью Sequelize. Затем мы реализовали CRUD-методы для работы с пользователями, а также запрос для получения пользователя по id/email.