САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2

Выполнил:

Трофимов Андрей

Группа K33402

Проверил: Добряков Д. И.

Санкт-Петербург

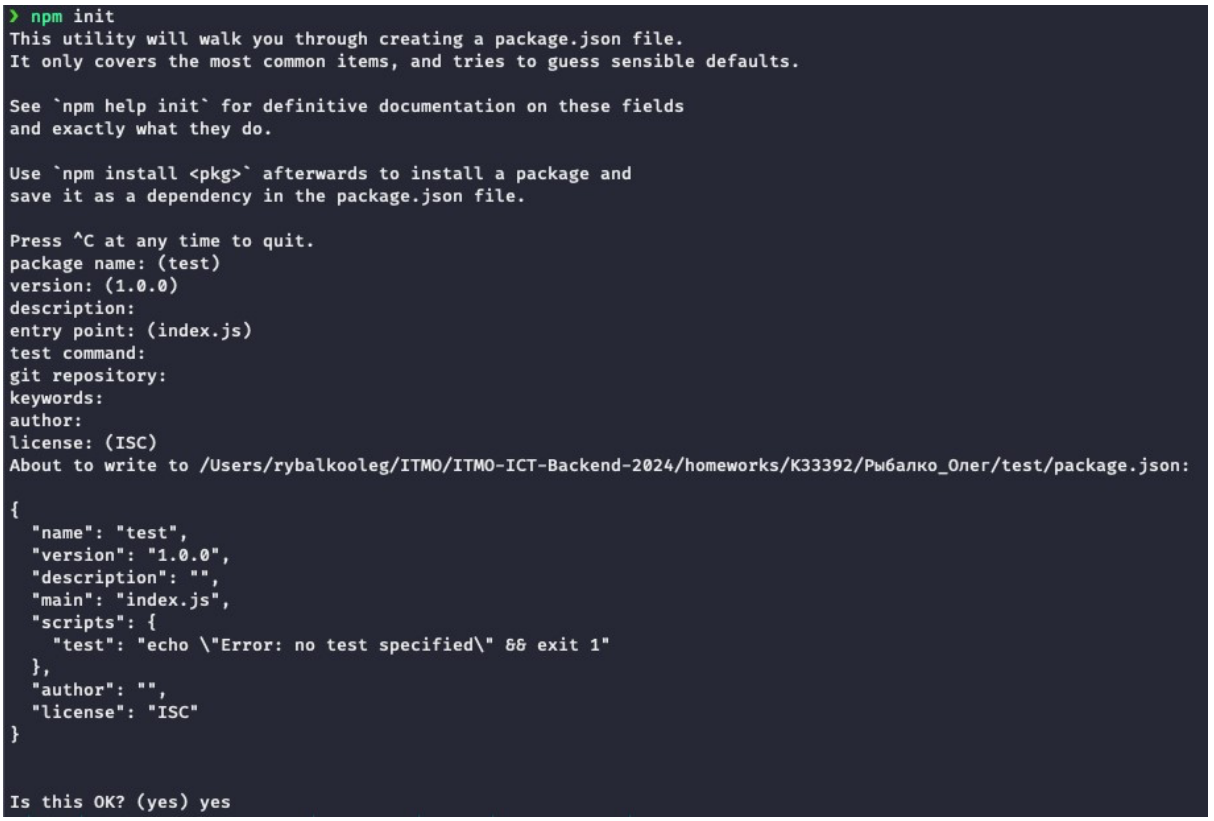
2024 г.

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. Инициализируем модуль: *npm init*



2. Установим зависимости:

   npm i express sequelize sqlite3 sequelize-cli
3. Инициализируем sequelize: *npx sequelize init*

4. Сгенерируем модель при помощи sequelize-cli

   npx sequelize-cli model:generate --name User --attributes "username:string, email:string, password:string, firstName:string, lastName:string, isAdmin:boolean"
5. Проведем миграцию
6. Создадим endpoint для получения всех пользователей

```
// Get all users
app.get('/users', async (req, res) => {
  try {
    res.json(await db.User.findAll())
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
})
```

7. Создадим endpoint для создания нового пользователя

```
// Create a new user
app.post('/users', async (req, res) => {
  try {
    const user = await db.User.create(req.body)
    res.status(200).json(user)
  } catch (error) {
    res.status(400).json({ error: error.message })
  }
})
```

8. Создадим endpoint для получения пользователя по id

```
// Get user by ID
app.get('/users/:id', async (req, res) => {
  try {
    const user = await db.User.findByPk(req.params.id)
    if (!user) {
      res.status(404).json({ error: 'User not found' })
    } else {
      res.json(user)
    }
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
})
```

9. Создадим endpoint для изменения данных пользователя

```
// Update user by ID
app.patch('/users/:id', async (req, res) => {
  try {
    const user = await db.User.findByPk(req.params.id)
    if (!user) {
      res.status(404).json({ error: 'User not found' })
    } else {
      await user.update(req.body)
      res.json(user)
    }
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
})
```

10. Создадим endpoint для удаление пользователя

```
// Delete user by ID
app.delete('/users/:id', async (req, res) => {
  try {
    const user = await db.User.findByPk(req.params.id)
    if (!user) {
      res.status(404).json({ error: 'User not found' })
    } else {
      await user.destroy()
      res.status(200).send()
    }
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
})
```

Вывод

В данной домашней работе удалось написать HTTP сервер, который обрабатывает запросы для CRUD-операций над пользователем при помощи библиотеки express и sequelize для работы с базой данных.