САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2

Выполнил:

Жигалова Анастасия Евгеньевна

Группа K33392

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

```
D:\back-end\HW2new>npm int
Unknown command: "int"

Did you mean this?
    npm init # Create a package.json file

To see a list of supported npm commands, run:
  npm help

D:\back-end\HW2new>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (hw2new)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\back-end\HW2new\package.json:

{
  "name": "hw2new",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

2. Установим зависимости:

   npm i express sequelize sqlite3 sequelize-cli

```
D:\back-end\HW2new>npm i express sequelize sqlite3 sequelize-cli
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs

added 316 packages, and audited 317 packages in 18s

40 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\back-end\HW2new>
```

3. Инициализируем sequelize:

   npx sequelize init

```
D:\back-end\HW2new>npx sequelize init

Sequelize CLI [Node: 20.11.1, CLI: 6.6.2, ORM: 6.37.1]

Created "config\config.json"
Successfully created models folder at "D:\back-end\HW2new\models".
Successfully created migrations folder at "D:\back-end\HW2new\migrations".
Successfully created seeders folder at "D:\back-end\HW2new\seeders".

D:\back-end\HW2new>
```

4. Сгенерируем модель при помощи sequelize-cli

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

```
D:\back-end\HW2new>npx sequelize-cli model:generate --name User --attributes "firstName:string, lastName:string, email:string, password:string"

Sequelize CLI [Node: 20.11.1, CLI: 6.6.2, ORM: 6.37.1]

New model was created at D:\back-end\HW2new\models\user.js .
New migration was created at D:\back-end\HW2new\migrations\20240313141302-create-user.js .
```

5. Выполним миграцию:

Изменим файл config.json

```
{
  "development": {
    "username": null,
    "password": null,
    "database": "database_development",
    "host": null,
    "dialect": "sqlite",
    "storage": "db.sqlite"
  },
  "test": {
    "username": "root",
    "password": null,
    "database": "database_test",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "production": {
    "username": "root",
    "password": null,
    "database": "database_production",
    "host": "127.0.0.1",
    "dialect": "mysql"
  }
}
```

npx sequelize-cli db:migrate

```
D:\back-end\HW2new>npx sequelize-cli db:migrate

Sequelize CLI [Node: 20.11.1, CLI: 6.6.2, ORM: 6.37.1]

Loaded configuration file "config\config.json".
Using environment "development".
== 20240313141302-create-user: migrating =======
== 20240313141302-create-user: migrated (0.013s)


D:\back-end\HW2new>
```

6. Создадаем endpoint для получения всех пользователей:

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

  try {
    const users = await db.User.findAll()
    res.json(users);

  } catch (error) {
    res.status(500).json({ error: error.message })
  }
}),
```

7. Создадаем endpoint для получения пользователя по id:

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

  try {
    const user = await db.User.findByPk(req.params.id)

    if (user) {
      return res.send(user.toJSON())
    } else {
      return res.send({ message: "User not found" })
    }

  } catch (error) {
    res.status(500).json({ error: error.message })
  }
}),
```

8. Создадаем endpoint для создания нового пользователя:

```
app.post('/users', async (req, res) => { //Создание нового пользователя

  try {
    const newUser = await db.User.create(req.body)
    res.status(200).json(newUser)

  } catch (error) {
    res.status(400).json({ error: error.message })
  }
}),
```

9. Создадаем endpoint для удаления пользователя по id:

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

  try {
    const user = await db.User.findByPk(req.params.id)

    if (user) {
      await user.destroy()
      res.status(200).send()
    } else {
      res.status(404).json({ message: 'User not found' })
    }

  } catch (error) {
    res.status(500).json({ error: error.message })
  }
}),
```

10. Создадаем endpoint для данных пользователя по id:

```
app.patch('/users/:id', async (req, res) => { //Изменение данных пользователя

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

    if (user) {
      await user.update(req.body);
      res.json(user);
    } else {
      res.status(404).json({ message: 'User not found' });
    }

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

Вывод

В данной домашней работе был написать HTTP сервер, который обрабатывает запросы для CRUD-операций над пользователем, для этого были использованы библиотеки express и sequelize для работы с базой данных.