САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа № 2

Выполнил:

Кононов Степан

Группа

K32392

Проверил:
Добряков Д. И.

Санкт-Петербург

2024г.

Знакомство с ORM Sequelize

Задание:

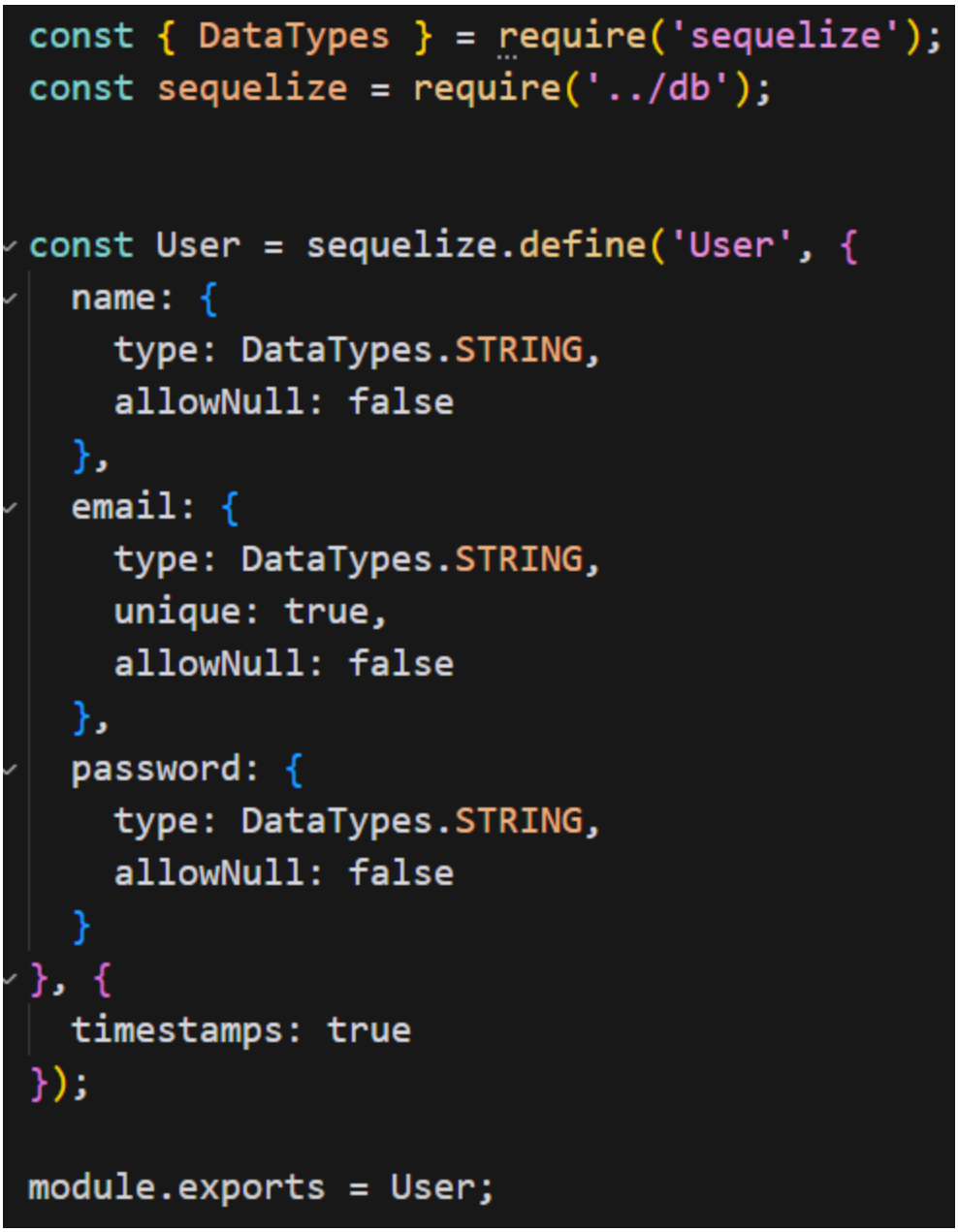
- Продумать свою собственную модель пользователя
- Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize
- Написать запрос для получения пользователя по id/email

Ход работы

1. Разработка модели пользователя

Для начала была создана модель `User`, которая включает следующие поля:

- `id` (авто-сгенерированный ID пользователя)
- `name` (имя пользователя, строка, обязательное поле)
- `email` (email пользователя, строка, обязательное и уникальное поле)
- `password` (пароль пользователя, строка, обязательное поле)



```
const { DataTypes } = require('sequelize');
const sequelize = require('../db');


const User = sequelize.define('User', {
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  email: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false
  }
}, {
  timestamps: true
});

module.exports = User;
```

2. Реализация CRUD-методов

В рамках этой работы были разработаны методы для выполнения операций CRUD:

- **Create (Создание)**: Метод `POST /users` для добавления нового пользователя. Данные принимаются в формате JSON и сохраняются в базе данных.

```
router.post('/users', async (req, res) => {
  const { name, email, password } = req.body;
  try {
    const user = await User.create({ name, email, password });
    res.status(201).json(user);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});
```

- **Read (Чтение)**: Метод `GET /users/:identifier` для получения информации о пользователе по ID или email. Если идентификатор не является числом, то производится поиск по email.

```
router.get('/users/:identifier', async (req, res) => {
  const identifier = req.params.identifier;
  try {
    let user;
    if (isNaN(identifier)) {
      user = await User.findOne({ where: { email: identifier } });
    } else {
      user = await User.findByPk(identifier);
    }

    if (user) {
      res.json(user);
    } else {
      res.status(404).json({ error: 'User not found' });
    }
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

- **Update (Обновление)**: Метод `PUT /users/:id` для обновления данных существующего пользователя. Пользователь может изменить имя, email и пароль.

```
router.put('/users/:id', async (req, res) => {
  const { name, email, password } = req.body;
  try {
    const user = await User.findByPk(req.params.id);
    if (user) {
      user.name = name;
      user.email = email;
      user.password = password;
      await user.save();
      res.json(user);
    } else {
      res.status(404).json({ error: 'User not found' });
    }
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});
```

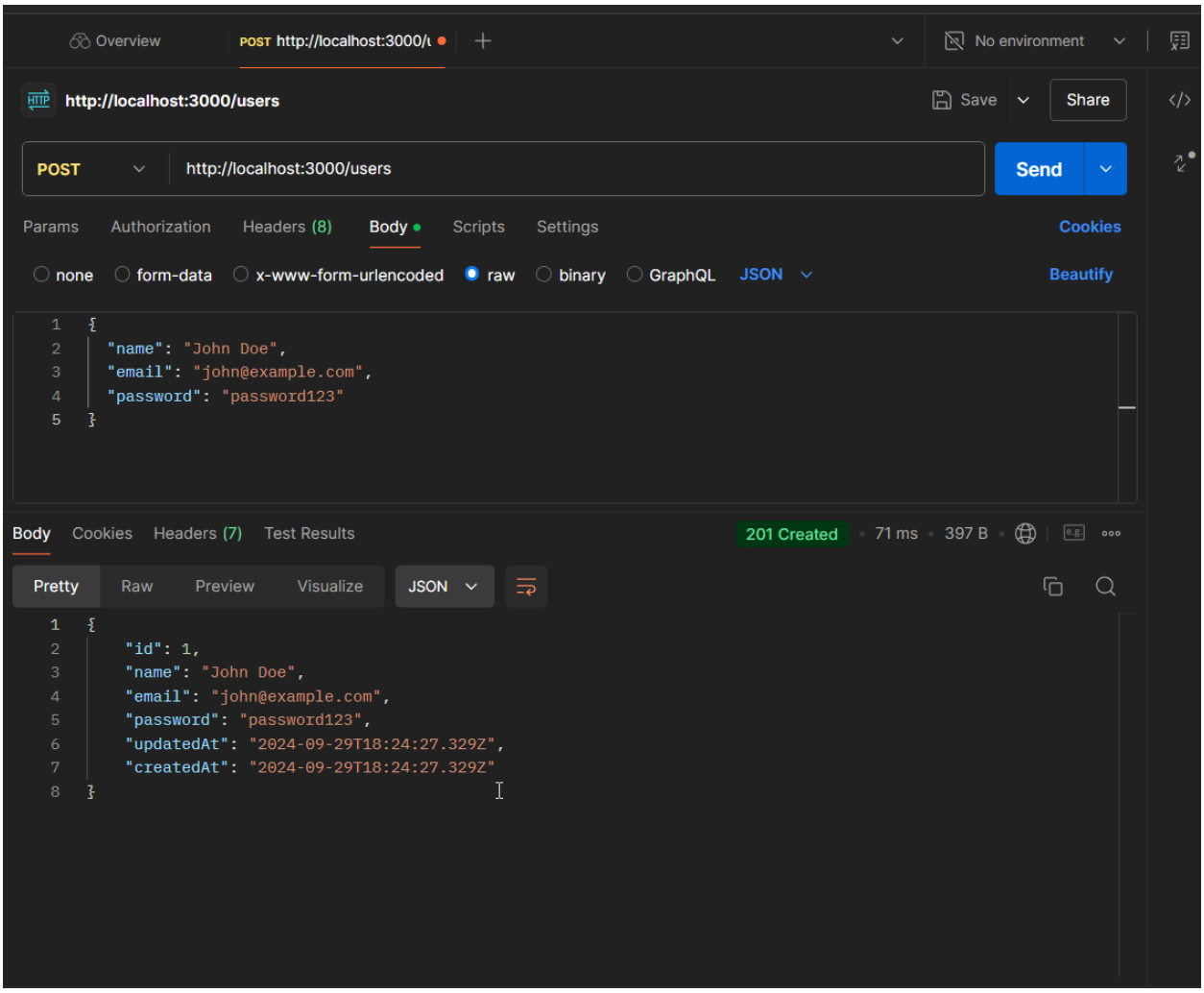
- **Delete (Удаление)**: Метод `DELETE /users/:id` для удаления пользователя по его ID. Если пользователь найден, он удаляется из базы данных.

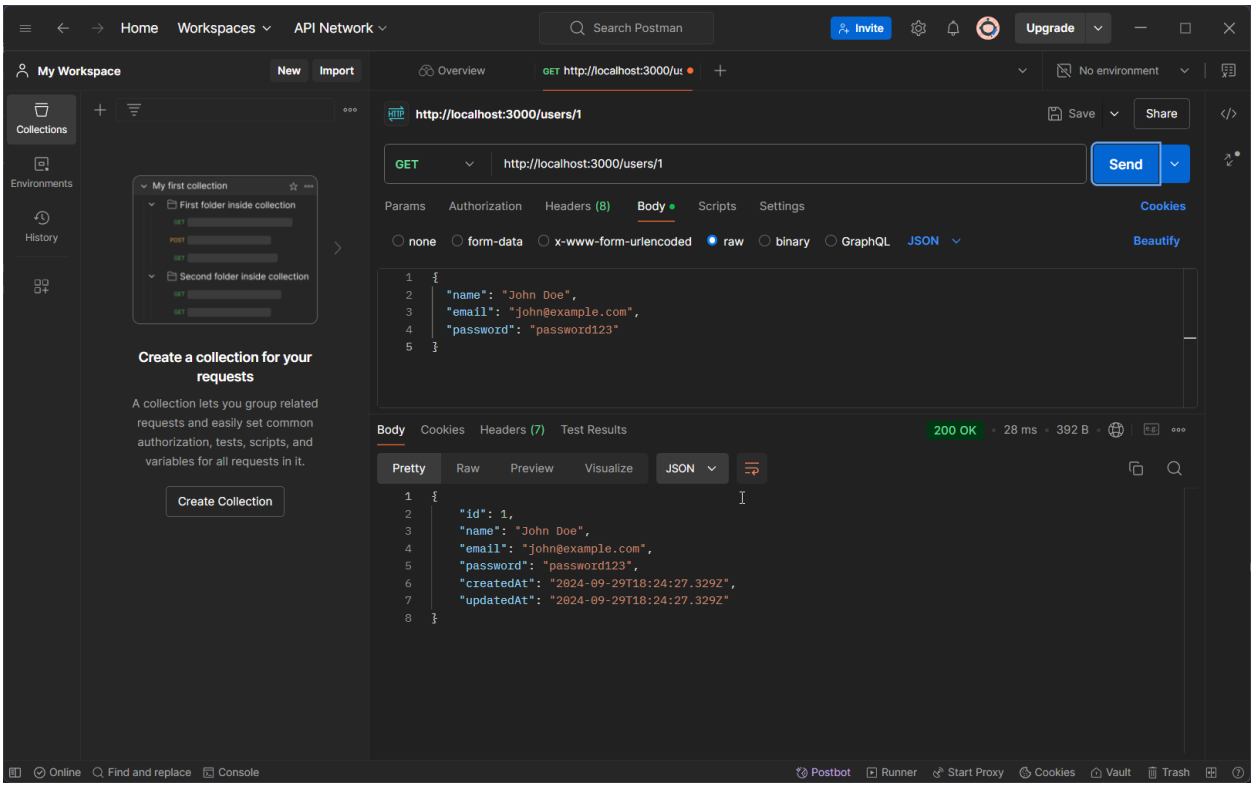
```
router.delete('/users/:id', async (req, res) => {
  try {
    const user = await User.findByPk(req.params.id);
    if (user) {
      await user.destroy();
      res.json({ message: 'User deleted' });
    } else {
      res.status(404).json({ error: 'User not found' });
    }
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

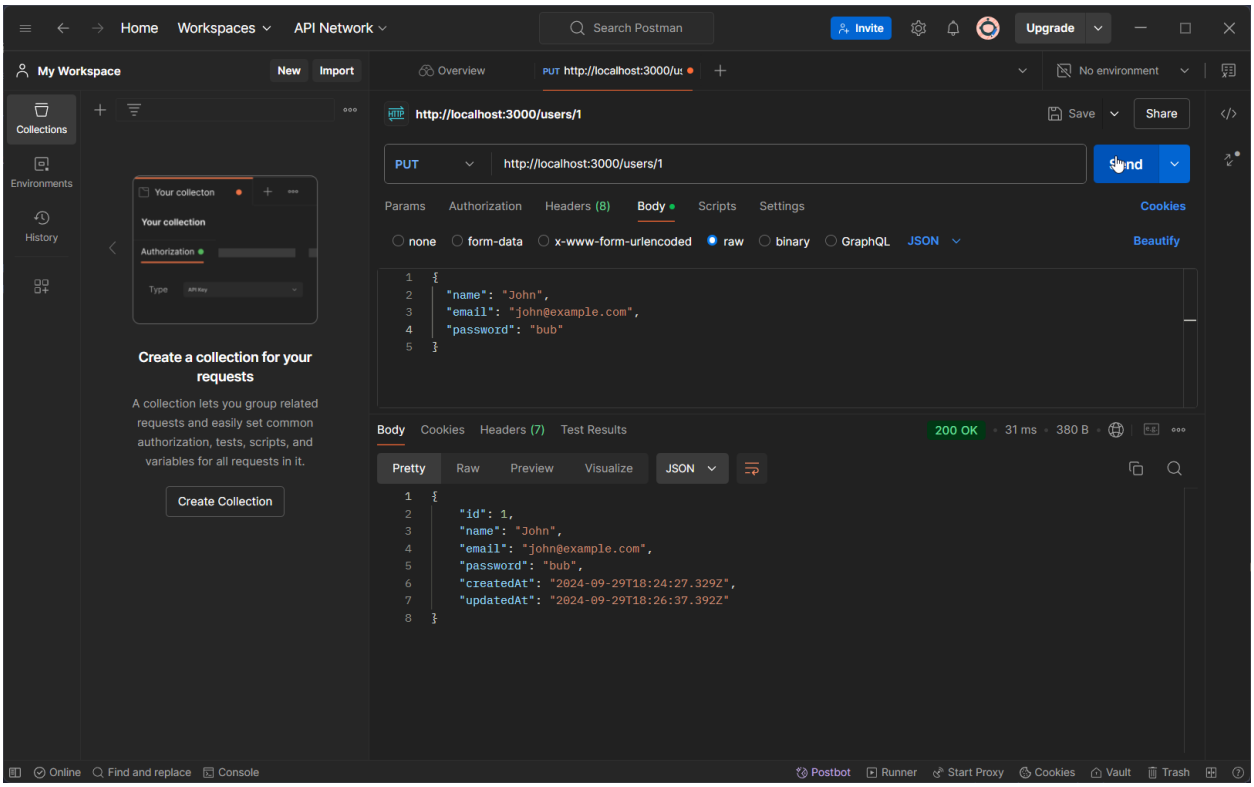
3. Запрос для получения пользователя

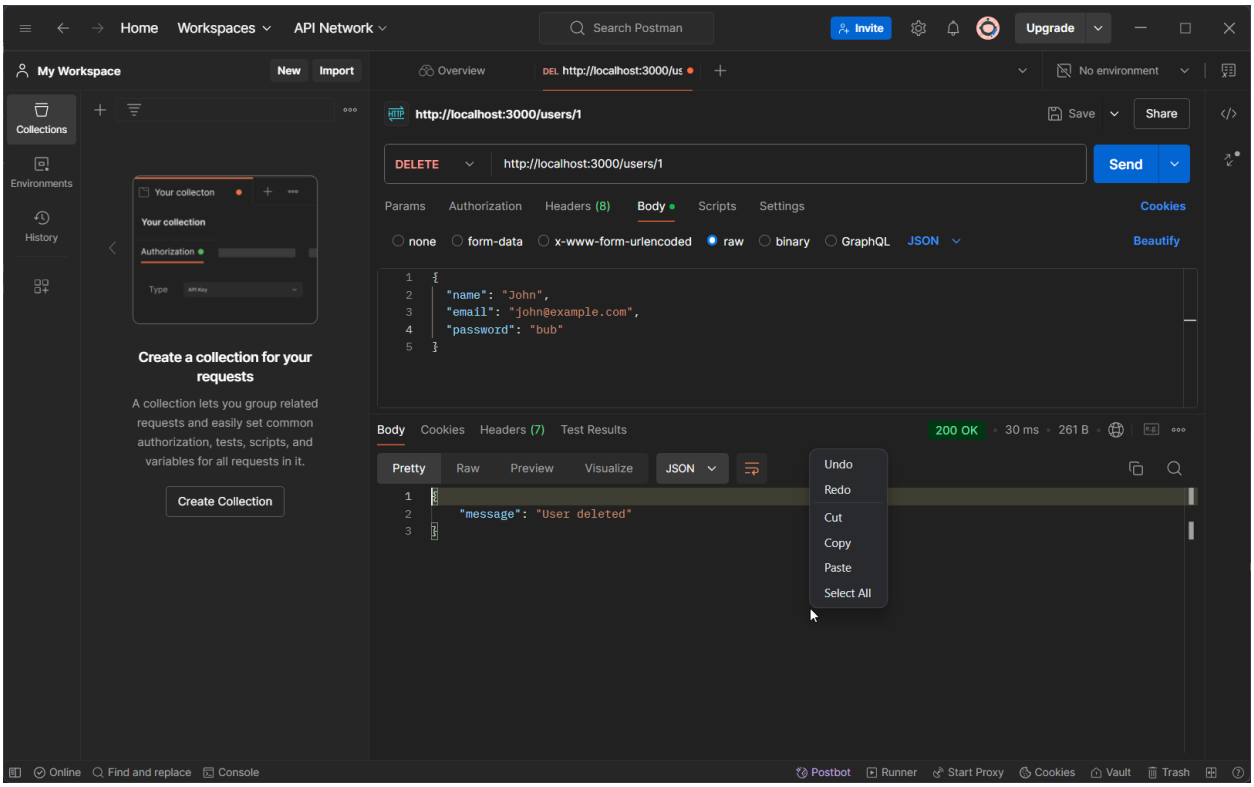
Запрос для получения пользователя был реализован через метод `GET /users/:identifier`. В зависимости от переданного параметра (ID или email) происходит соответствующий поиск в базе данных.

4. Проверка запросов через Postman


Overview
POST http://localhost:3000/
No environment
http://localhost:3000/users
Save
Share
POST
http://localhost:3000/users
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "John Doe",
"email": "john@example.com",
"password": "password123"
}
Body
Cookies
Headers (7)
Test Results
201 Created
71 ms
397 B
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "John Doe",
"email": "john@example.com",
"password": "password123",
"updatedAt": "2024-09-29T18:24:27.329Z",
"createdAt": "2024-09-29T18:24:27.329Z"
}


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET http://localhost:3000/us
No environment
Collections
Environments
History
http://localhost:3000/users/1
Save
Share
GET
http://localhost:3000/users/1
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "John Doe",
"email": "john@example.com",
"password": "password123"
}
My first collection
First folder inside collection
Second folder inside collection
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (7)
Test Results
200 OK
28 ms
392 B
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "John Doe",
"email": "john@example.com",
"password": "password123",
"createdAt": "2024-09-29T18:24:27.329Z",
"updatedAt": "2024-09-29T18:24:27.329Z"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
PUT http://localhost:3000/us
No environment
Collections
Environments
History
http://localhost:3000/users/1
Save
Share
PUT
http://localhost:3000/users/1
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "John",
"email": "john@example.com",
"password": "bub"
}
Your collecton
Your collection
Authorization
Type
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (7)
Test Results
200 OK
31 ms
380 B
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "John",
"email": "john@example.com",
"password": "bub",
"createdAt": "2024-09-29T18:24:27.329Z",
"updatedAt": "2024-09-29T18:26:37.392Z"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash



Заключение

В результате выполнения задачи была успешно реализована модель пользователя с CRUD-методами, а также добавлена возможность получения информации о пользователе по ID или email.