

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 2: Знакомство с Sequelize и Express

Выполнил:

Золотухин Артем

К33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

Задача

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

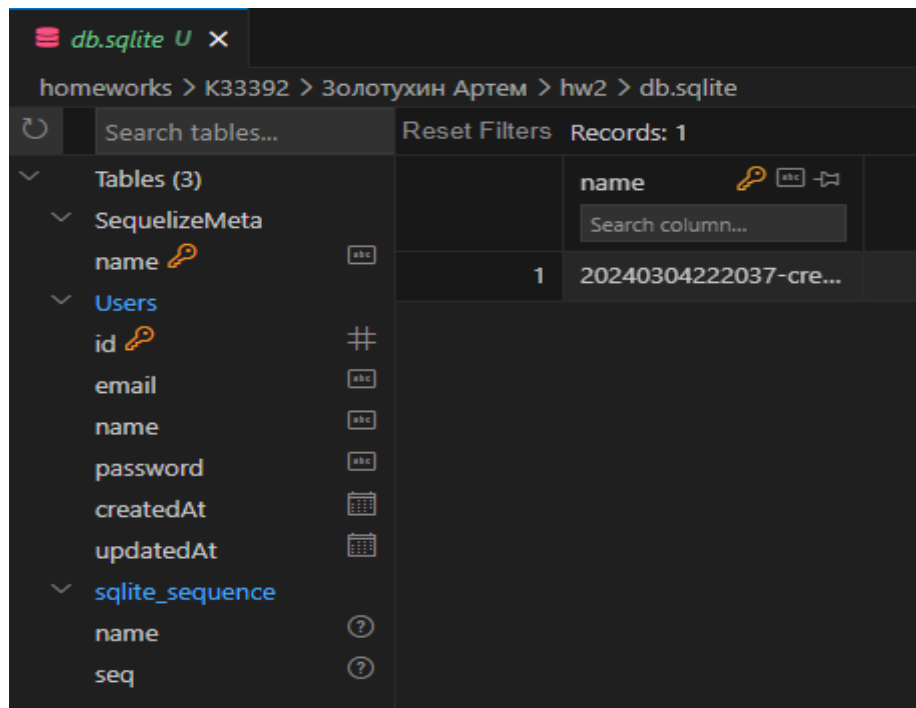
Сначала я настроил Sequelize и прописал модель данных юзера.

```
1 'use strict'
2 const { Model } = require('sequelize')
3 module.exports = (sequelize, DataTypes) => {
4   class User extends Model {
5     static associate(models) {}
6   }
7   User.init(
8     {
9     id: {
10      type: DataTypes.INTEGER,
11      autoIncrement: true,
12      primaryKey: true,
13    },
14    email: {
15      type: DataTypes.STRING,
16      unique: true,
17      validate: {
18        isEmail: true,
19      },
20    },
21    name: DataTypes.STRING,
22    password: {
23      type: DataTypes.STRING,
24    },
25    createdAt: {
26      type: DataTypes.DATE,
27      defaultValue: DataTypes.NOW,
28    },
29    updatedAt: {
30      type: DataTypes.DATE,
31      defaultValue: DataTypes.NOW,
32    },
33  },
34  {
35    sequelize,
36    modelName: 'User',
37    tableName: 'Users',
38    timestamps: true,
39  }
40 )
41 return User
42 }
43
```

Index.js моделей, сгенерированный командой “npx sequelize-cli init” оставляю нетронутым, также не забываем прописать “npx sequelize-cli db:migrate” для создания миграций. Далее я прописываю эти самые миграции для моей модели пользователя:

```
1  'use strict'
2
3  /** @type {import('sequelize-cli').Migration} */
4  module.exports = {
5    async up(queryInterface, Sequelize) {
6      await queryInterface.createTable('Users', {
7        id: {
8          allowNull: false,
9          autoIncrement: true,
10         primaryKey: true,
11         type: Sequelize.INTEGER,
12       },
13
14       email: {
15         type: Sequelize.STRING,
16         unique: true,
17         validate: {
18           isEmail: true,
19         },
20       },
21       name: {
22         type: Sequelize.STRING,
23       },
24       password: {
25         type: Sequelize.STRING,
26       },
27       createdAt: {
28         allowNull: false,
29         type: Sequelize.DATE,
30       },
31       updatedAt: {
32         allowNull: false,
33         type: Sequelize.DATE,
34       },
35     })
36   },
37
38   async down(queryInterface, Sequelize) {
39     await queryInterface.dropTable('Users')
40   },
41 }
42
```

Файл конфига оставляю нетронутым, так как по дефолту там настроено подключение к sqlite, а чтобы подключиться к другой RDBMS нужно немного изменить конфиг. После всех этих махинаций у нас должен появиться файл бд:

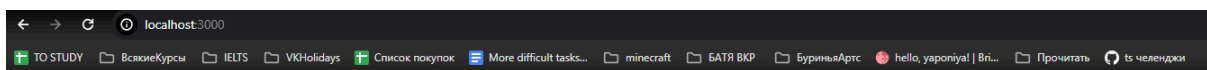


Далее настраиваем сервер и роутинг, используя `express`. Для начала создам сам сервер на порту 3000, для этого я сначала создам файл `.env`, в котором, пока что, будет храниться только порт, на котором будет запущен сервер. Вдобавок напишу скрипт запуска сервера на `nodemon`, который включит для сервера `hotreload`:

```
"scripts":  
{  
  "devStart": "nodemon server.js"  
},
```

Далее настроим сервер так, чтобы при переходе на его URL открывалась стартовая страница (ради приличия), на которую я мог бы передать какой-то текст:

```
1  const express = require('express')
2
3  const app = express()
4  app.set('view engine', 'ejs')
5  app.use(express.json())
6  app.use(express.urlencoded({ extended: true }))
7
8  app.get('/', (req, res) => {
9    res.render('index', {
10      text: 'If you see this text that means that everything is working just fine',
11    })
12  })
13
14  const PORT = process.env.PORT || 3000
15  app.listen(PORT, () => console.log(`Server running on port ${PORT}`))
16
```



Home page

If you see this text that means that everything is working just fine

Далее нужно добавить рауты по которым можно создать/вернуть/обновить и удалить пользователя. Для этого я создам отдельный файл user.js в папке routes. Там я создам все нужные мне пути:

```

1  const express = require('express')
2  const router = express.Router()
3  const { User } = require('../models')
4  const { Op } = require('sequelize')
5
6  router.post('/', async (req, res) => {
7    try {
8      let user = undefined
9      if (req.body) {
10        user = await User.create(req.body)
11      } else {
12        user = await User.create({
13          email: 'artem.zolotukhin.303@gmail.com',
14          name: 'Artem',
15          password: '110100zz',
16        })
17      }
18      res.json(user)
19    } catch (error) {
20      res.status(400).json({ error: error.message })
21    }
22  })
23
24  router.get('/', async (req, res) => {
25    try {
26      const users = await User.findAll()
27      res.send(users)
28    } catch (error) {
29      res.status(400).json({ error: error.message })
30    }
31  })
32
33  router
34    .route('/:id')
35    .get(async (req, res) => {
36      try {
37        const user = await User.findOne({
38          where: {
39            [Op.or]: [{ id: req.params.id }, { email: req.params.id }],
40          },
41        })
42        if (user) {
43          res.json(user)
44        } else {
45          res.status(404).json({ error: 'User not found' })
46        }
47      } catch (error) {
48        res.status(400).json({ error: error.message })
49      }
50    })
51    .put(async (req, res) => {
52      try {
53        const user = await User.findByPk(req.params.id)
54        if (user) {
55          await user.update(req.body)
56          res.json(user)
57        } else {
58          res.status(404).json({ error: 'User not found' })
59        }
60      } catch (error) {
61        res.status(400).json({ error: error.message })
62      }
63    })
64    .delete(async (req, res) => {
65      try {
66        const user = await User.findByPk(req.params.id)
67        if (user) {
68          await user.destroy()
69          res.status(204).send()
70        } else {
71          res.status(404).json({ error: 'User not found' })
72        }
73      } catch (error) {
74        res.status(400).json({ error: error.message })
75      }
76    })
77
78  module.exports = router
79

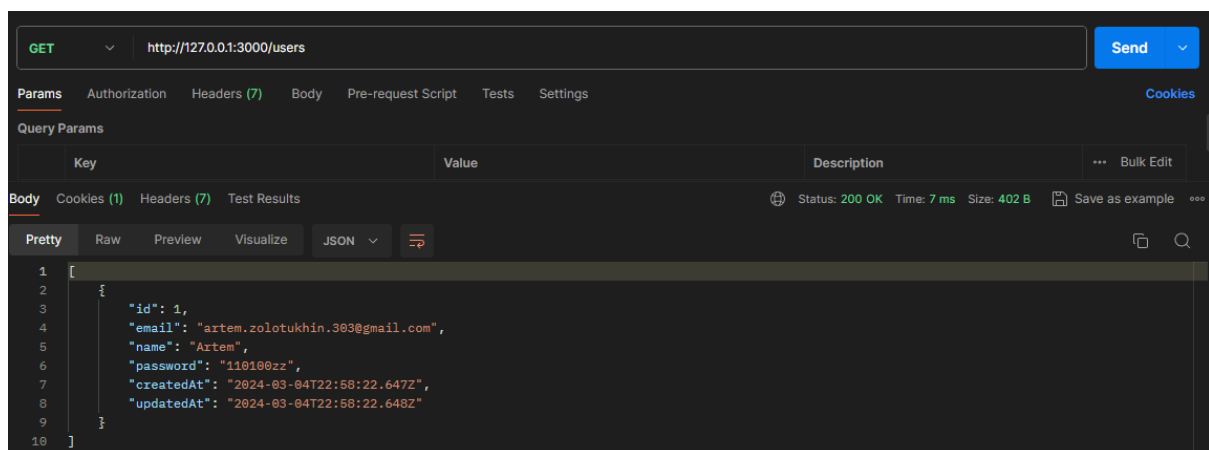
```

Потом нужно подключить этот router к нашему серверу:

```
1 const userRouter = require('./routes/users')
2 app.use('/users', userRouter)
```

Также я создал seeders, чтобы по дефолту заполнять свою бд данными, но я так ими и не воспользовался.

Теперь, когда все подключено, проверим что все работает:



GET <http://127.0.0.1:3000/users> Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

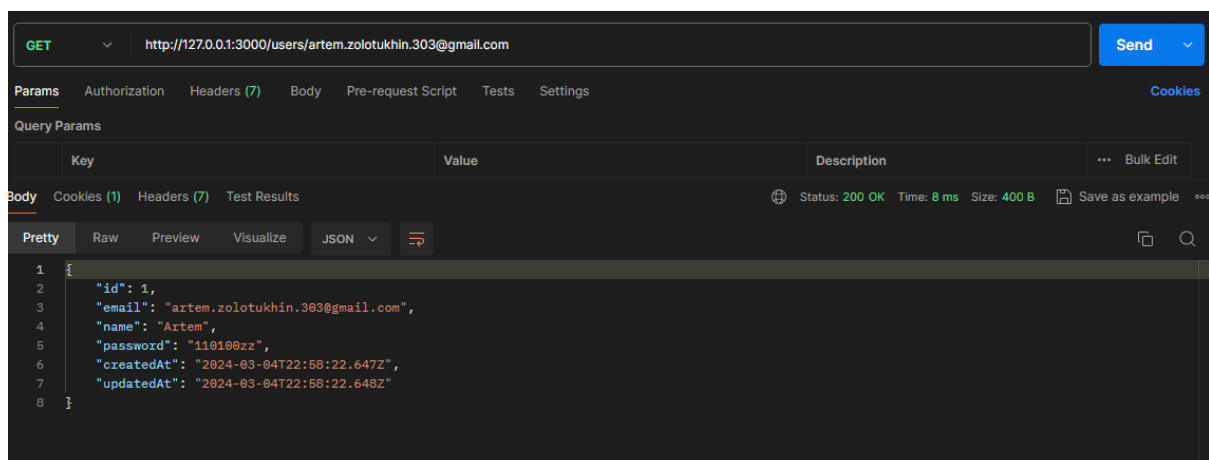
Query Params

Key	Value	Description	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies (1) Headers (7) Test Results Status: 200 OK Time: 7 ms Size: 402 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "email": "artem.zolotukhin.303@gmail.com",
5     "name": "Artem",
6     "password": "110100zz",
7     "createdAt": "2024-03-04T22:58:22.647Z",
8     "updatedAt": "2024-03-04T22:58:22.648Z"
9   }
10 ]
```



GET <http://127.0.0.1:3000/users/artem.zolotukhin.303@gmail.com> Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies (1) Headers (7) Test Results Status: 200 OK Time: 8 ms Size: 400 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "email": "artem.zolotukhin.303@gmail.com",
4   "name": "Artem",
5   "password": "110100zz",
6   "createdAt": "2024-03-04T22:58:22.647Z",
7   "updatedAt": "2024-03-04T22:58:22.648Z"
8 }
```

GET

http://127.0.0.1:3000/users/1

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

	Key	Value	Description	...	Bulk Edit
--	-----	-------	-------------	-----	-----------

BodyCookies (1)Headers (7)Test Results

Status: 200 OKTime: 24 msSize: 400 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 1,
3   "email": "artem.zolotukhin.303@gmail.com",
4   "name": "Artem",
5   "password": "110100zz",
6   "createdAt": "2024-03-04T22:58:22.647Z",
7   "updatedAt": "2024-03-04T22:58:22.648Z"
8 }
```

GET

http://127.0.0.1:3000/users/2

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

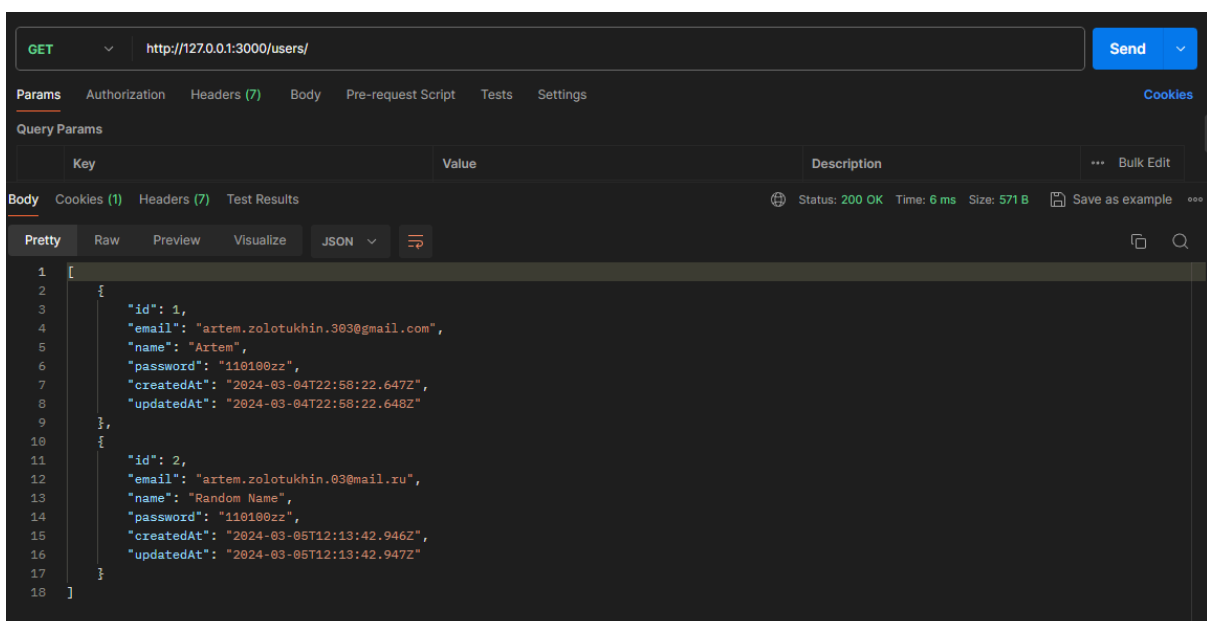
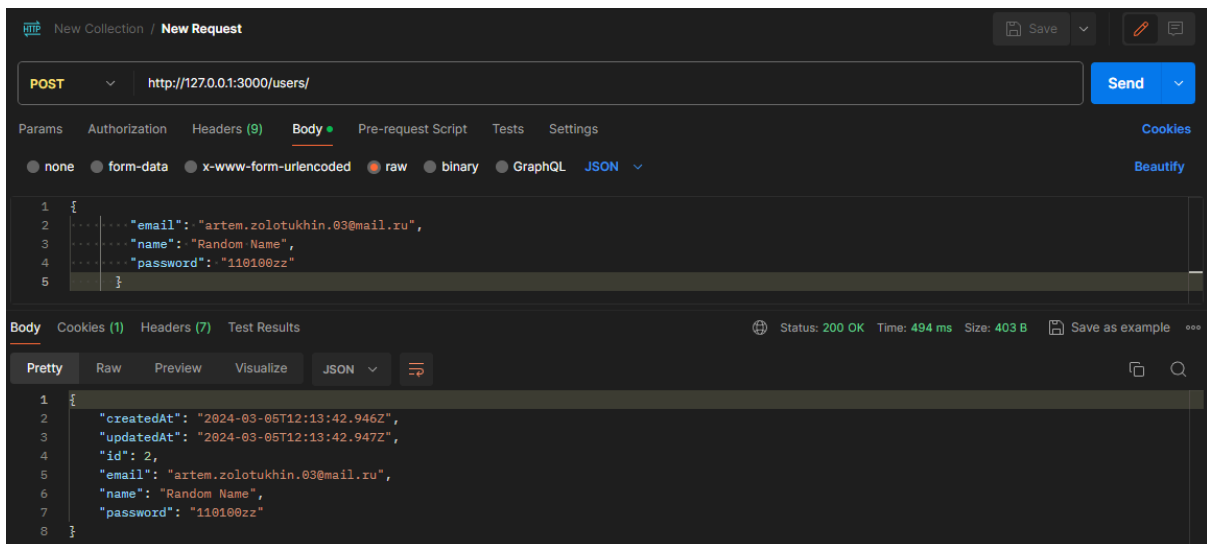
	Key	Value	Description	...	Bulk Edit
--	-----	-------	-------------	-----	-----------

BodyCookies (1)Headers (7)Test Results

Status: 404 Not FoundTime: 8 msSize: 268 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "error": "User not found"
3 }
```

Также работают функции обновления и удаления, но они не важны для этой работы, так что не буду утяжелять отчет скриншотами.

Вывод

В ходе данной работы я научился использовать такую orm как Sequelize, а также писать простые сервера при помощи фреймворка Express.