

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Практическая работа №2: Создание своего сервера

Выполнил:  
Жаров Александр  
К33402

Проверил:  
Добряков Д. И.

Санкт-Петербург

2024 г.

## Задача

- 1) Придумать собственную модель пользователя
- 2) Реализовать набор из CRUD методов для работы с пользователем
- 3) Написать запрос для получения пользователя по id или email

## Ход работы

- 1) Устанавливаем необходимые зависимости



```
1 {
2   "name": "hw2",
3   "version": "1.0.0",
4   "description": "",
5   "main": "server.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "",
10  "license": "ISC",
11  "dependencies": {
12    "cors": "^2.8.5",
13    "express": "^4.18.3",
14    "mysql2": "^3.9.2",
15    "sequelize": "^6.37.1",
16    "sqlite3": "^5.1.7"
17  }
18 }
19
```

- 2) Инициализируем server на express и подключаемся к бд

```

1  const express = require("express");
2  const cors = require("cors"); 4.5k (gzipped: 1.9k)
3
4  const app = express();
5
6  const db = require("../app/models");
7
8  db.sequelize
9    .sync()
10   .then(() => {
11     console.log("Synced db.");
12   })
13   .catch((err) => {
14     console.log("Failed to sync db: " + err.message);
15   });
16
17 var corsOptions = {
18   origin: "http://localhost:8081",
19 };
20
21 app.use(cors(corsOptions));
22 app.use(express.json());
23 app.use(express.urlencoded({ extended: true }));
24
25 app.get("/", (req, res) => {
26   res.json({ message: "Welcome to application." });
27 });
28
29 require("../app/routes/user.routes")(app);
30
31 const PORT = process.env.PORT || 8080;
32 app.listen(PORT, () => {
33   console.log(`Server is running on port ${PORT}.`);
34 });
35

```

3) Создаем конфигурационный файл для бд на SQLite

```

config > JS db.config.js > ...
1  module.exports = {
2    HOST: "localhost",
3    USER: "sasha",
4    PASSWORD: "1234",
5    DB: "mydb",
6    dialect: "sqlite",
7    storage: "./db/database.sqlite",
8  };
9

```

4) Создаем модель пользователя и передаем ее в бд

```

1  module.exports = (sequelize, Sequelize) => {
2    const User = sequelize.define("user", {
3      name: {
4        type: Sequelize.STRING,
5        allowNull: false,
6      },
7      age: {
8        type: Sequelize.INTEGER,
9        allowNull: false,
10     },
11     email: {
12       type: Sequelize.STRING,
13       allowNull: false,
14       unique: true,
15     },
16     password: {
17       type: Sequelize.STRING,
18       allowNull: false,
19     },
20   });
21
22   return User;
23 };
24

```

```

1  const Sequelize = require("sequelize");
2  const dbConfig = require("../config/db.config.js");
3
4  const sequelize = new Sequelize({
5    dialect: dbConfig.dialect,
6    storage: dbConfig.storage,
7  });
8
9  const db = {};
10
11  db.Sequelize = Sequelize;
12  db.sequelize = sequelize;
13
14  db.users = require("./user.model.js")(sequelize, Sequelize);
15
16  module.exports = db;
17

```

5) Создаем CRUD операции для взаимодействия с бд

```

exports.create = (req, res) => {
  if (
    !req.body.name ||
    !req.body.age ||
    !req.body.email ||
    !req.body.password
  ) {
    res.status(400).send({
      message: "All fields are required!",
    });
    return;
  }

  const user = {
    name: req.body.name,
    age: req.body.age,
    email: req.body.email,
    password: req.body.password,
  };

  User.create(user)
    .then((data) => {
      res.send(data);
    })
    .catch((err) => {
      res.status(500).send({
        message: err.message || "Some error occurred while creating the User.",
      });
    });
};

```

```
exports.findAll = (req, res) => {  
  User.findAll()  
    .then((data) => {  
      res.send(data);  
    })  
    .catch((err) => {  
      res.status(500).send({  
        message: err.message || "Some error occurred while retrieving users.",  
      });  
    });  
};
```

```
exports.findOne = (req, res) => {  
  const id = req.params.id;  
  
  User.findById(id)  
    .then((data) => {  
      if (data) {  
        res.send(data);  
      } else {  
        res.status(404).send({  
          message: `User with id=${id} not found.`,  
        });  
      }  
    })  
    .catch((err) => {  
      res.status(500).send({  
        message: "Error retrieving user with id=" + id,  
      });  
    });  
};
```

```

exports.update = (req, res) => {
  const id = req.params.id;

  User.update(req.body, {
    where: { id: id },
  })
    .then((num) => {
      if (num == 1) {
        res.send({
          message: "User was updated successfully.",
        });
      } else {
        res.send({
          message: `Cannot update user with id=${id}. Maybe user was not found or req.body is empty!`,
        });
      }
    })
    .catch((err) => {
      res.status(500).send({
        message: "Error updating user with id=" + id,
      });
    });
};

```

```

exports.delete = (req, res) => {
  const id = req.params.id;

  User.destroy({
    where: { id: id },
  })
    .then((num) => {
      if (num == 1) {
        res.send({
          message: "User was deleted successfully!",
        });
      } else {
        res.send({
          message: `Cannot delete user with id=${id}. Maybe user was not found!`,
        });
      }
    })
    .catch((err) => {
      res.status(500).send({
        message: "Could not delete user with id=" + id,
      });
    });
};

```

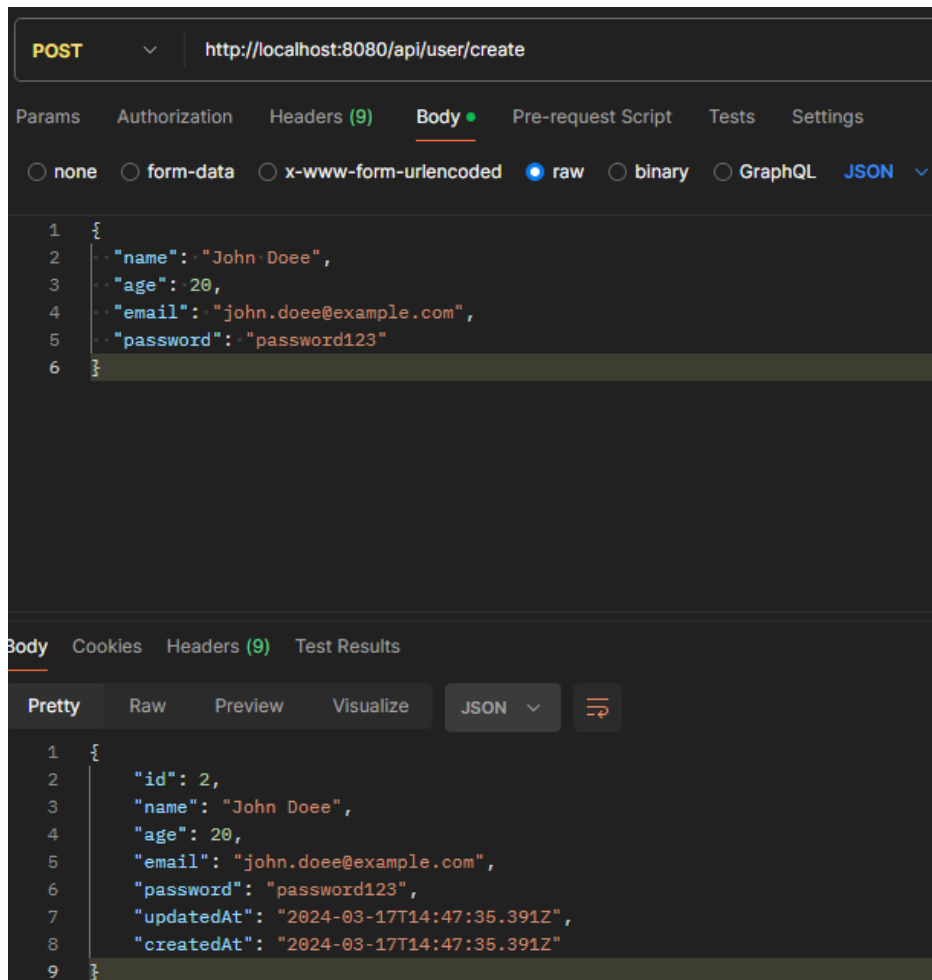
6) Создаем модуль с роутами

```

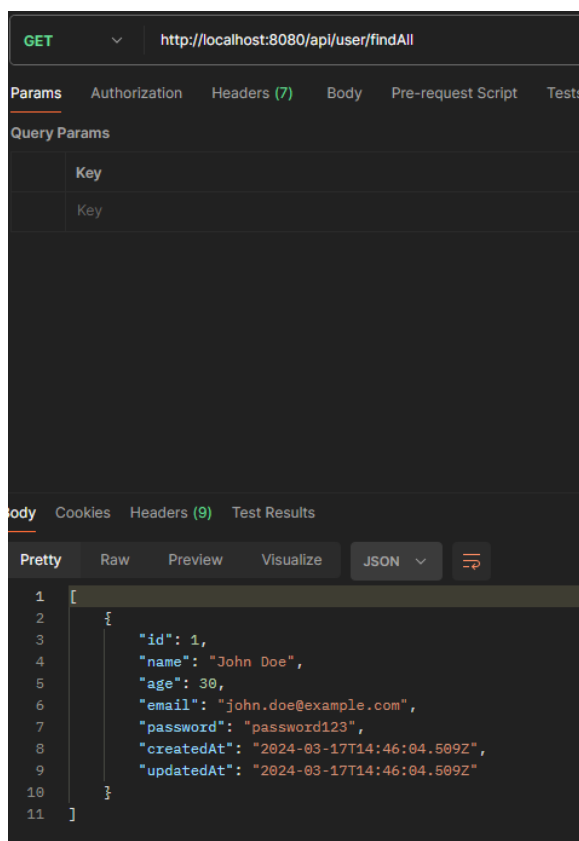
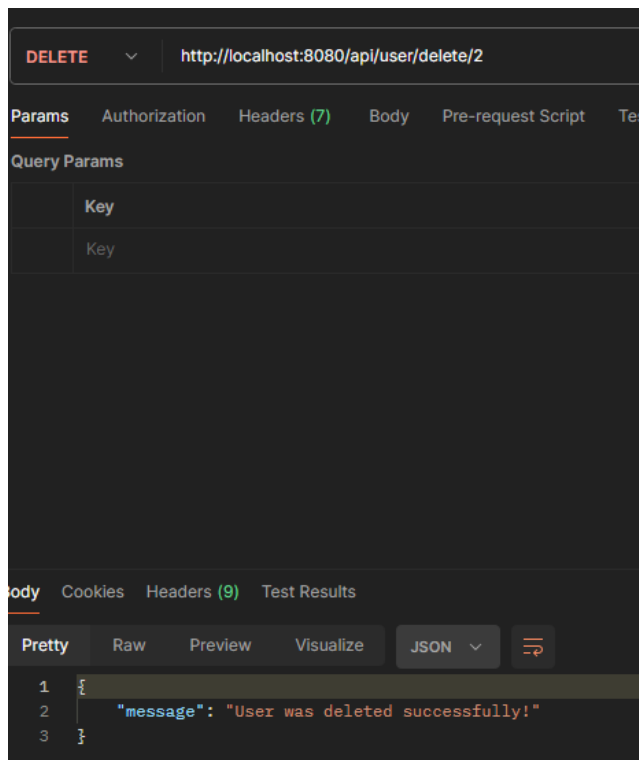
1  module.exports = (app) => {
2      const users = require("../controllers/user.controller.js");
3
4      var router = require("express").Router();
5
6      router.post("/create", users.create);
7      router.get("/findAll", users.findAll);
8      router.get("/find/:id", users.findOne);
9      router.put("/update/:id", users.update);
10     router.delete("/delete/:id", users.delete);
11
12     app.use("/api/user", router);
13 };
14

```

7) Проверяем работу в postman







## Вывод

В ходе работы я написал собственный сервер, создал базу данных, где хранятся пользователи и написал апи для этой бд.