САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:
Кулёмин С. А.
Группа K33401

Проверил:
Добряков Д. И.

Санкт-Петербург
2022 г.

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

Модель пользователя

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class User extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here
    }
  }
  User.init( attributes: {
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    username: DataTypes.STRING,
    password: DataTypes.STRING
  }, options: {
    sequelize,
    modelName: 'User',
  });
  return User;
};
```

Работа с sequelize

npx sequelize-cli init

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

npx sequelize-cli db:migrate

Приложение

```
const express = require('express')
const db = require('./models')
const bodyParser = require("body-parser");

const app = express()
const port = 3000

app.use(bodyParser.json())

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
})

app.get('/', (req, res) => {
    res.send('Hello World!')
})
```

CRUD-методы

Create

```
//CREATE
app.post('/add', async (req, res) => {
    const user = await db.User.create(req.body)
    res.send(user.toJSON())
})
```

Read

```
//READ
app.get('/get', async (req, res) => {
    const users = await db.User.findAll()
    res.send(users)
})

app.get('/users/id/:id', async (req, res) => {
    const user = await db.User.findByPk(req.params.id)
    if (user) {
        res.send(user.toJSON())
    } else {
        res.status(404).send({'msg': 'user not found'})
    }

})

app.get('/users/username/:username', async (req, res) => {
    const user = await db.User.findOne({
        where: {
            username: req.params.username
        }
    })
    if (user) {
        res.send(user.toJSON())
    } else {
        res.status(404).send({'msg': 'user not found'})
    }

})
```

Update

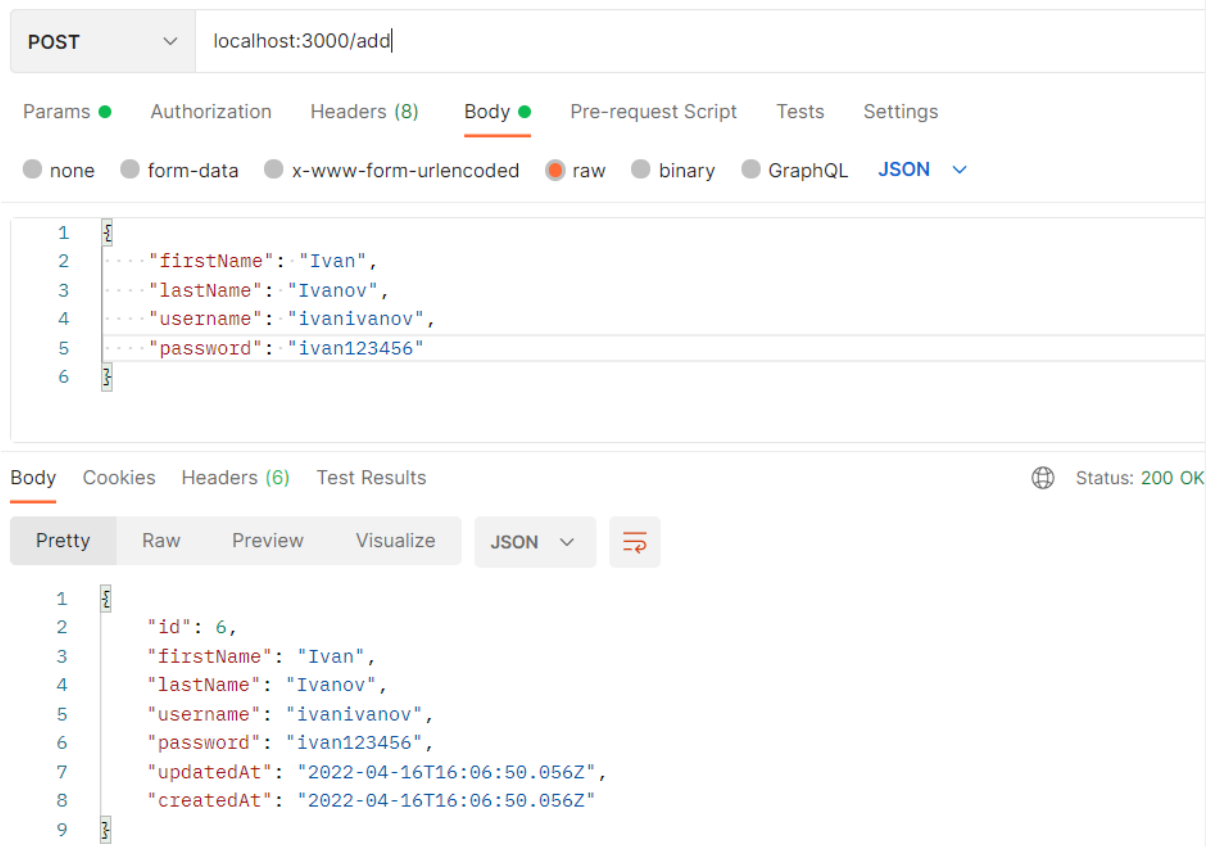
```
//UPDATE
app.put('update/:id', async (req, res) => {
    const user = await db.User.findByPk(req.params.id)
    if (user) {
        await db.User.update(req.body, {
            where: {
                id: req.params.id
            }
        })
    }
})
```

Delete

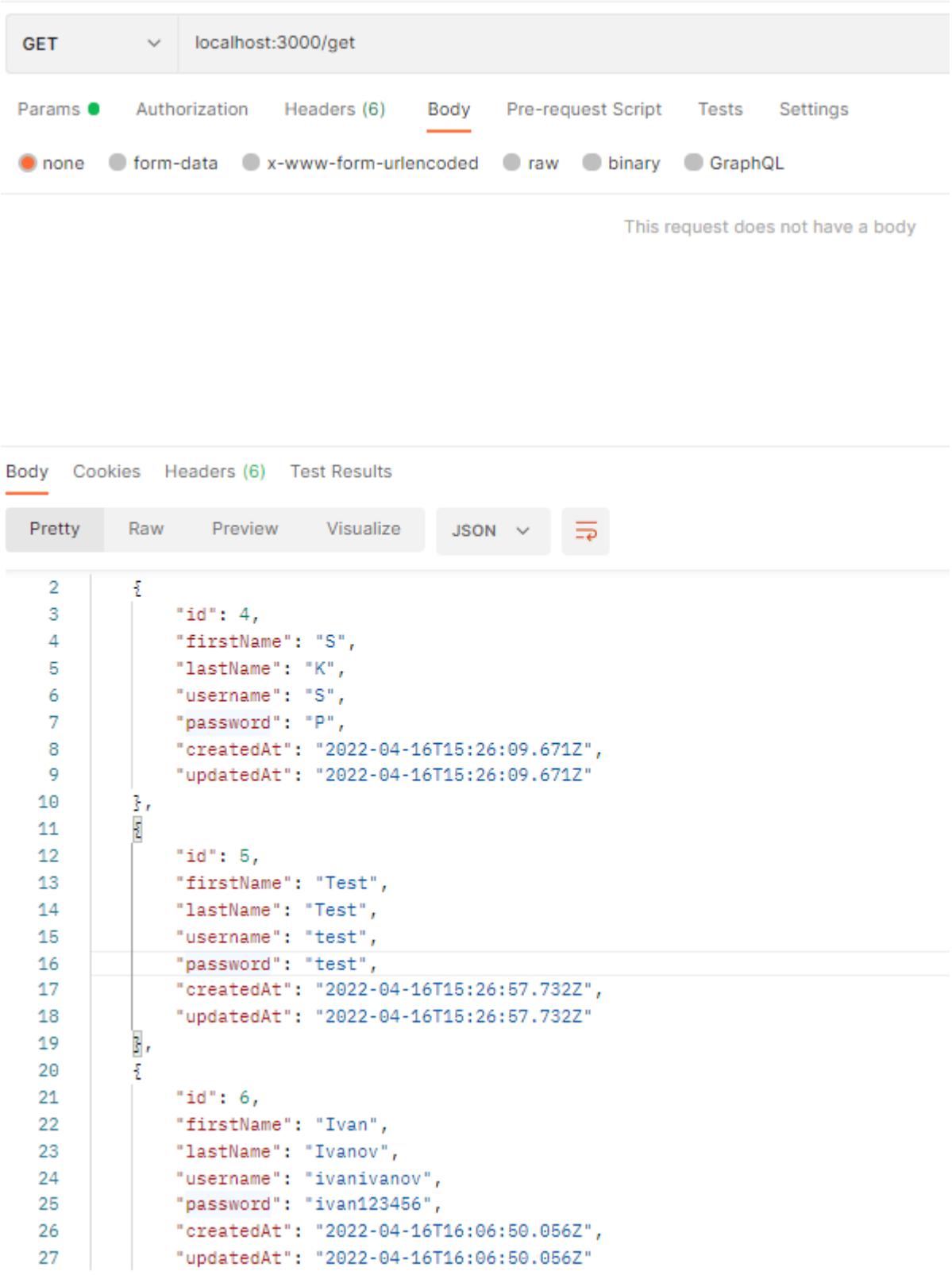
```
//DELETE
app.delete('/delete/:id', async (req, res) => {
    const user = await db.User.findByPk(req.params.id)
    if (user) {
        user.destroy();
        res.sendStatus( statusCode: 200).send({'msg': 'user deleted'})
    }
    res.status(404).send({'msg': 'user not found'})
})
```

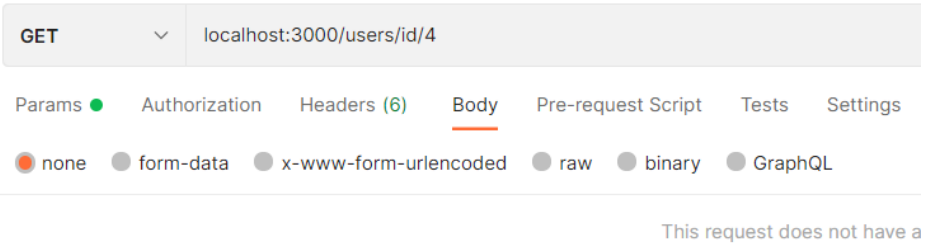
Проверка работы

Добавление пользователя

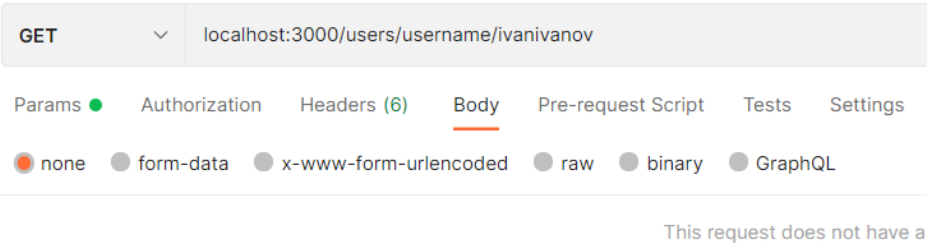


Чтение списка пользователей



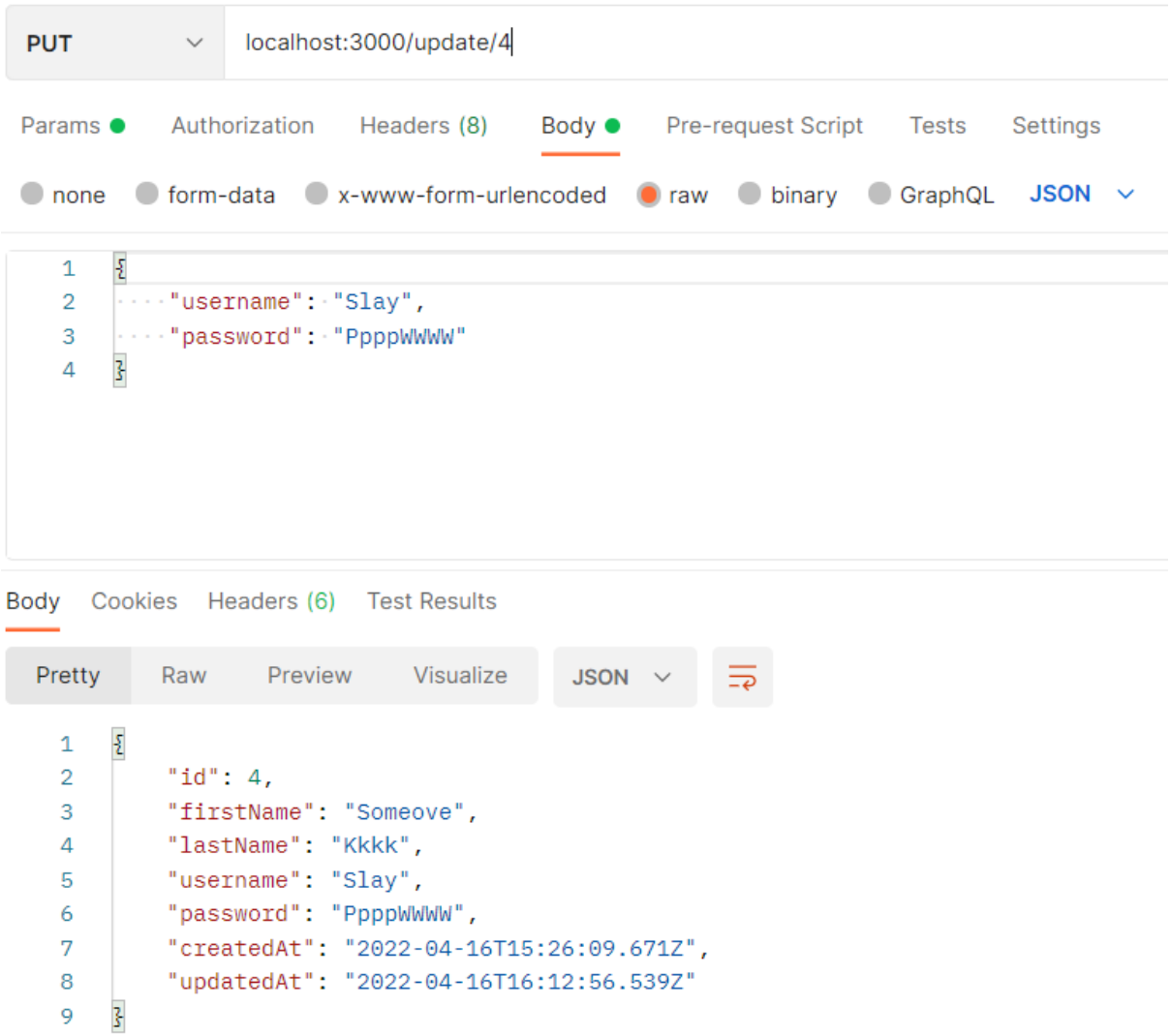

GET
localhost:3000/users/id/4
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a


Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"id": 4,
"firstName": "S",
"lastName": "K",
"username": "S",
"password": "P",
"createdAt": "2022-04-16T15:26:09.671Z",
"updatedAt": "2022-04-16T15:26:09.671Z"
}

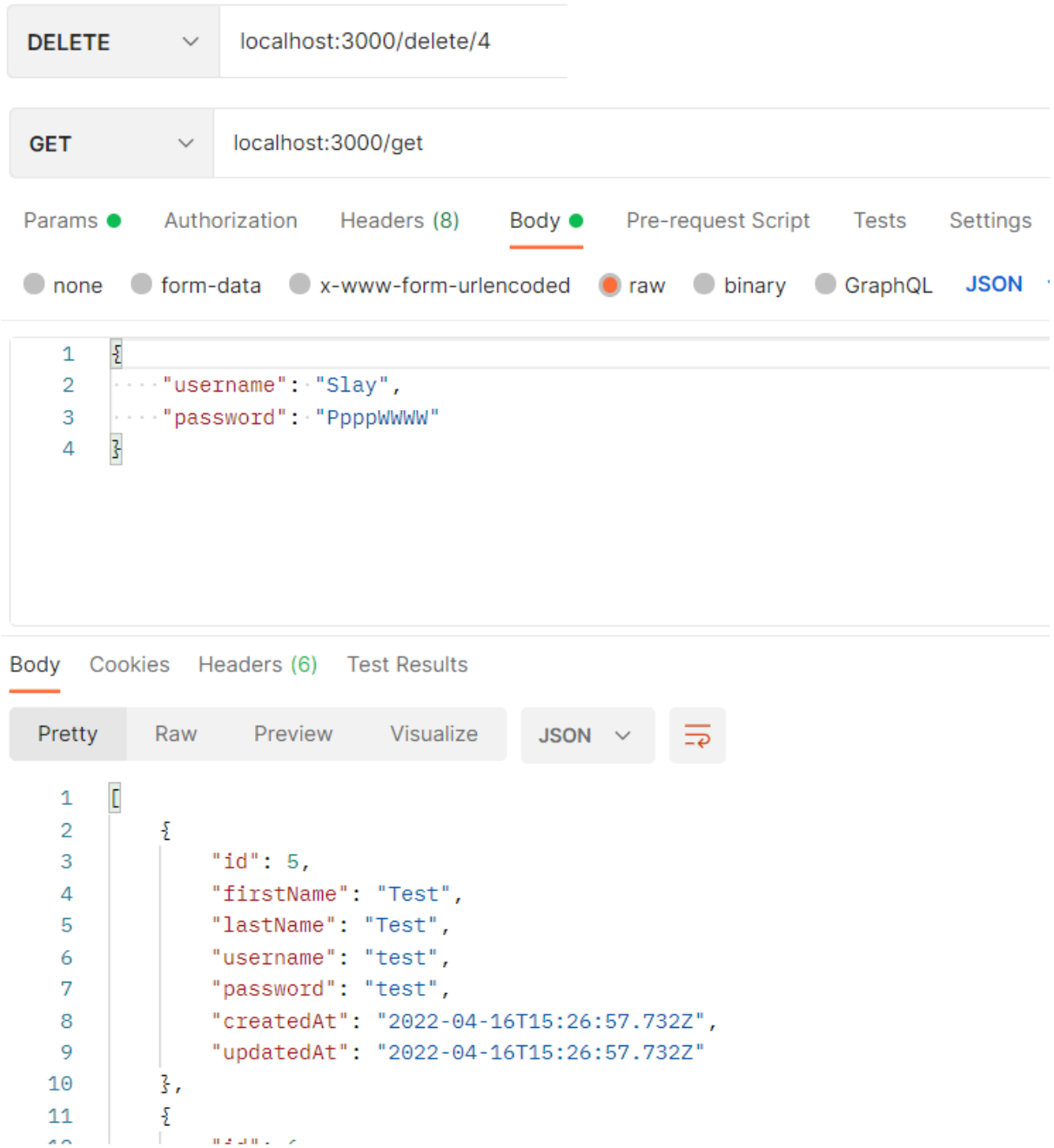

GET
localhost:3000/users/username/ivanivanov
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a


Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"id": 6,
"firstName": "Ivan",
"lastName": "Ivanov",
"username": "ivanivanov",
"password": "ivan123456",
"createdAt": "2022-04-16T16:06:50.056Z",
"updatedAt": "2022-04-16T16:06:50.056Z"
}

Обновление пользователя



Удаление пользователя



Вывод

В ходе выполнения работы я ознакомился с основами express и sequelize. Кроме того, был создан скрипт для управления базой данных (создания, удаления, обновления записей).