САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:
Вахрушева Ксения

Группа:
K33412

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 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

Ход задания:

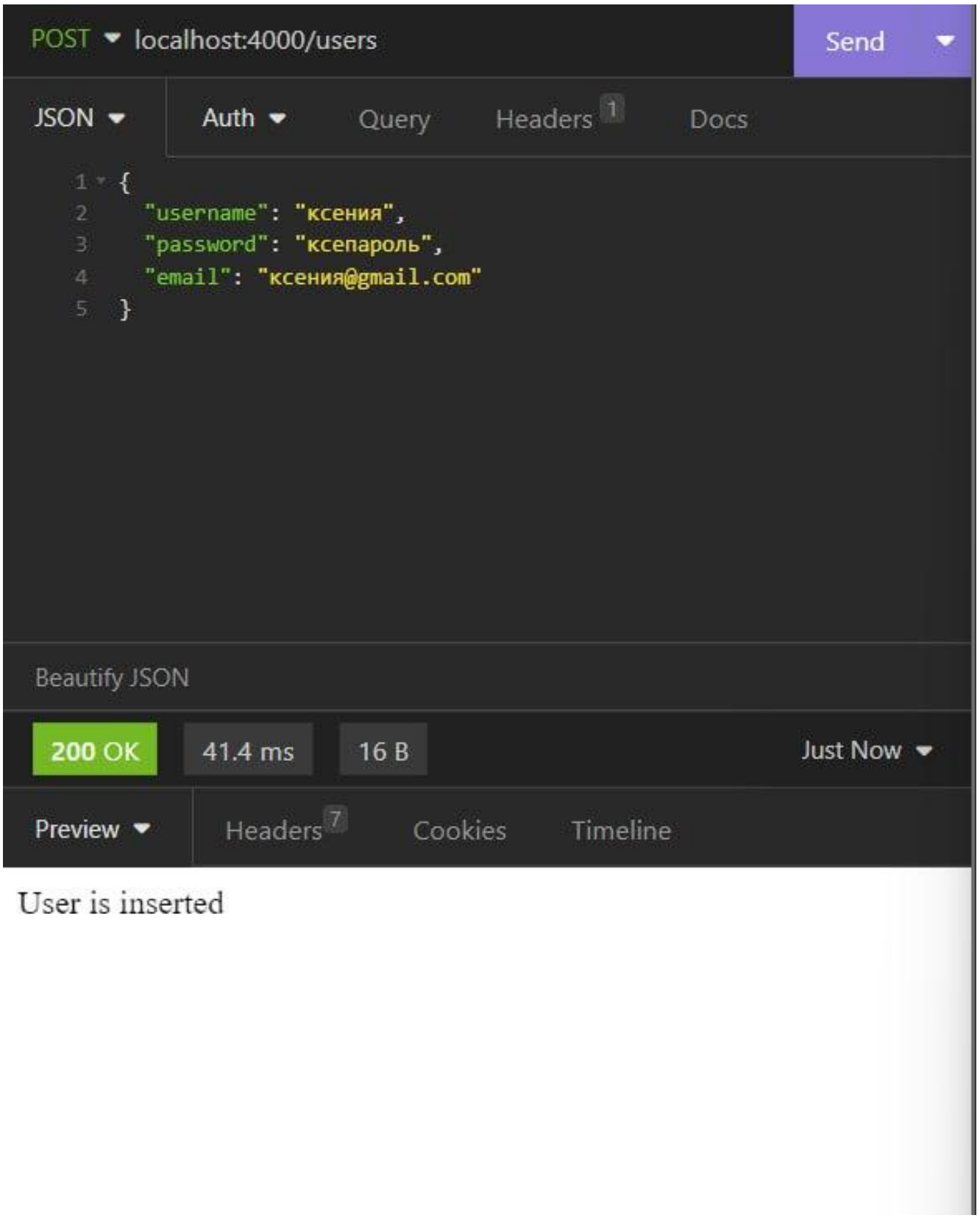
1) Модель пользователя:

```
//lets says if we have user table in our db, so we are creating user model


const {Model, DataTypes} = require('sequelize');
const sequelize = require('./database');

class User extends Model{}


//here initilizing the user and it takes to two parameters
// entities of db and second our database name
User.init({
    username: {
        type:DataTypes.STRING
    },
    password:{
        type:DataTypes.STRING
    },
    email:{
        type:DataTypes.STRING
    }
},{
    sequelize,
    modelName: 'user', // this is to give name to our db tabel
    timestamps: false  // for not showing timing in our table
})

module.exports = User;
```

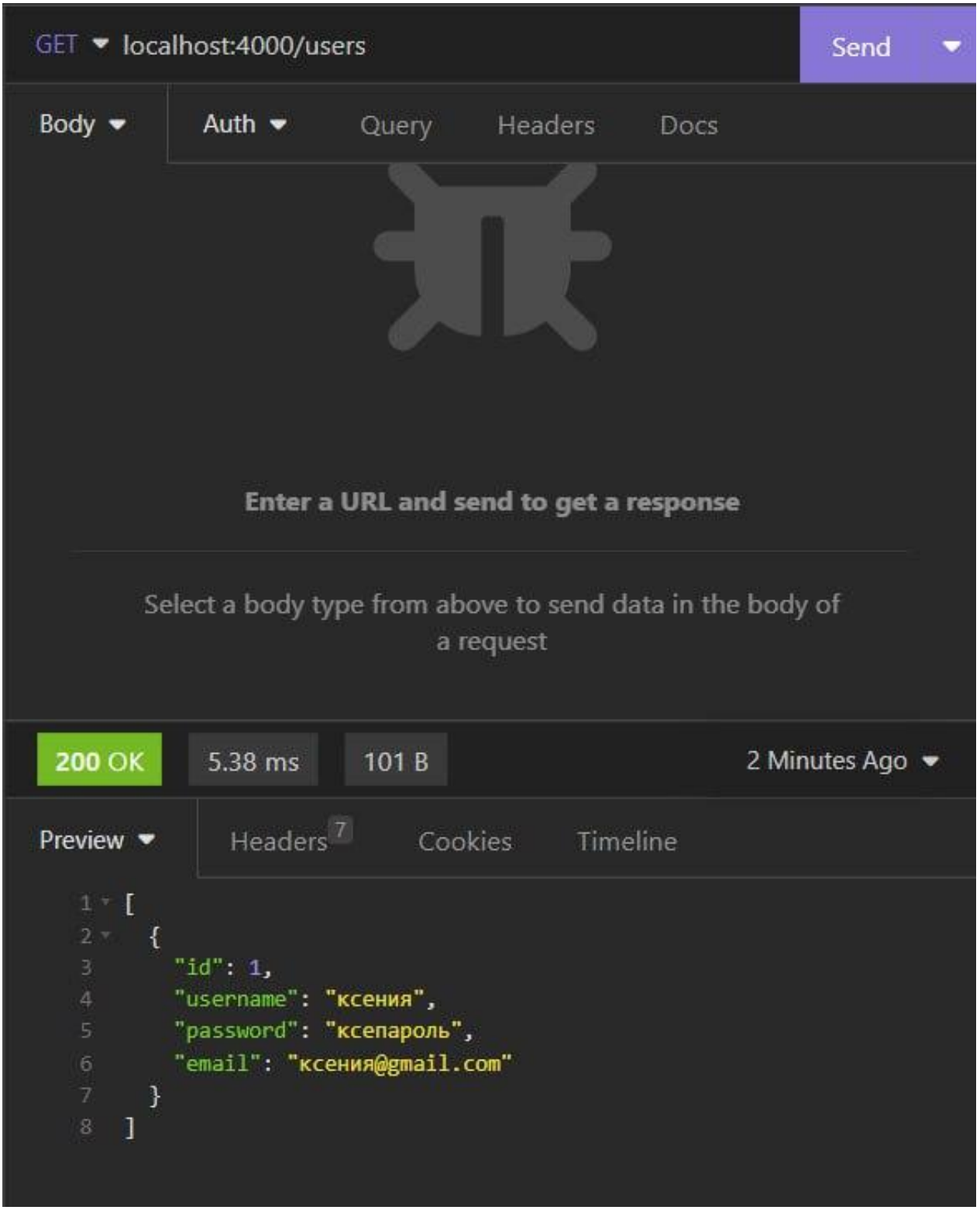
CRUD-методы:

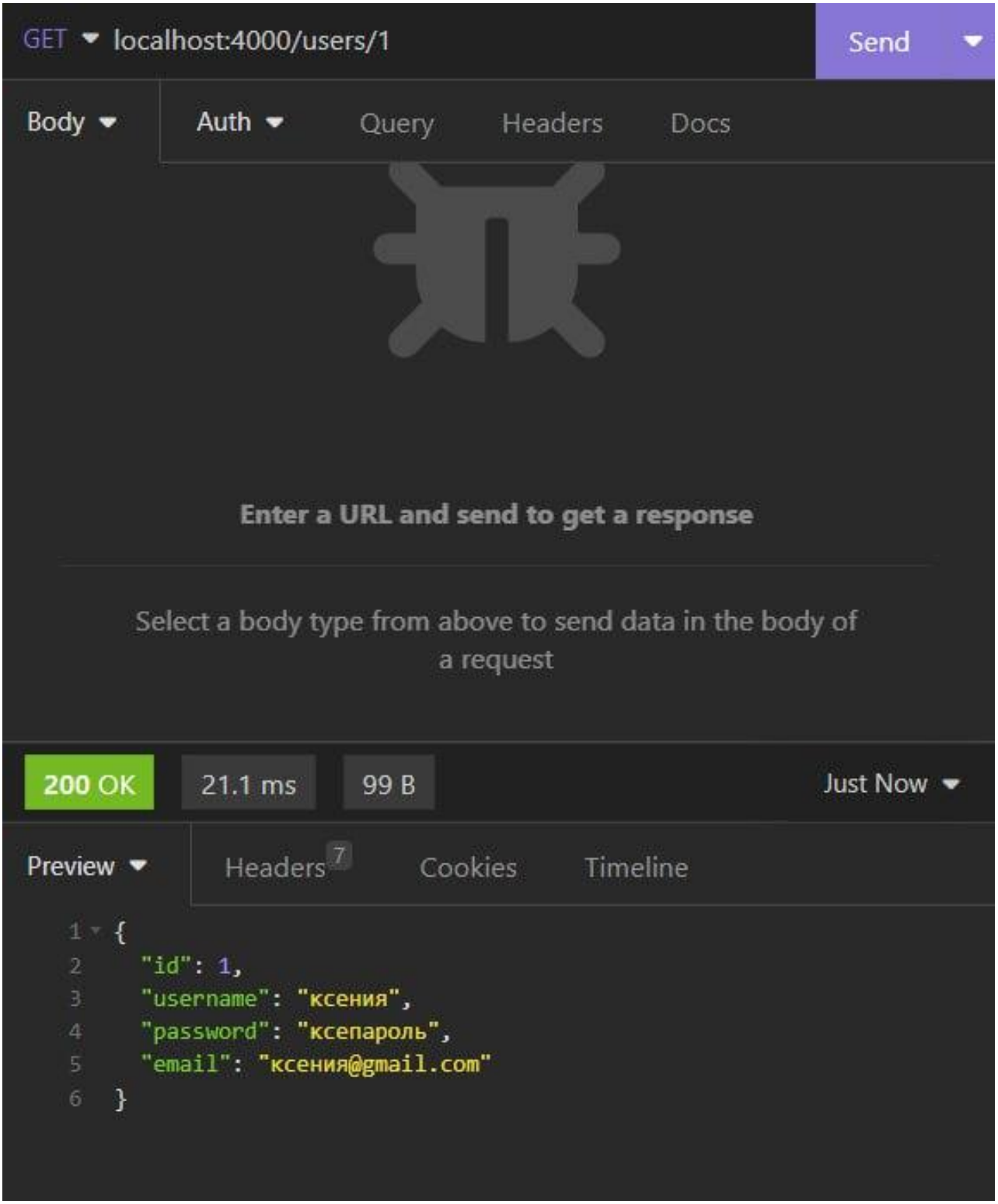
```
const express = require("express");

//for important our instance file we do as below
const sequelize = require('./database');//importing the file
const User = require('./User');
//for running our database we do as below
sequelize.sync({force: true}).then(()=> console.log("db is ready"))
//force: true. is used here to delete old data from table and create table again every time

const app = express()

app.use(express.json()); // this to make express aware what data is coming. unless it will put null in
app.post('/users', async (req,res)=>{
    await User.create(req.body)
        res.send("User is inserted")

});
//for getting all users in table
app.get('/users', async (req, res)=>{
    const myuser = await User.findAll()
        res.send(myuser);
    })
//for getting specific user in table
app.get('/users/:id',async (req,res)=>{
    const requestedId = req.params.id;
    const requestedUser = await User.findOne({where: {id: requestedId} })
    res.send(requestedUser)
})
```

2) Создание:

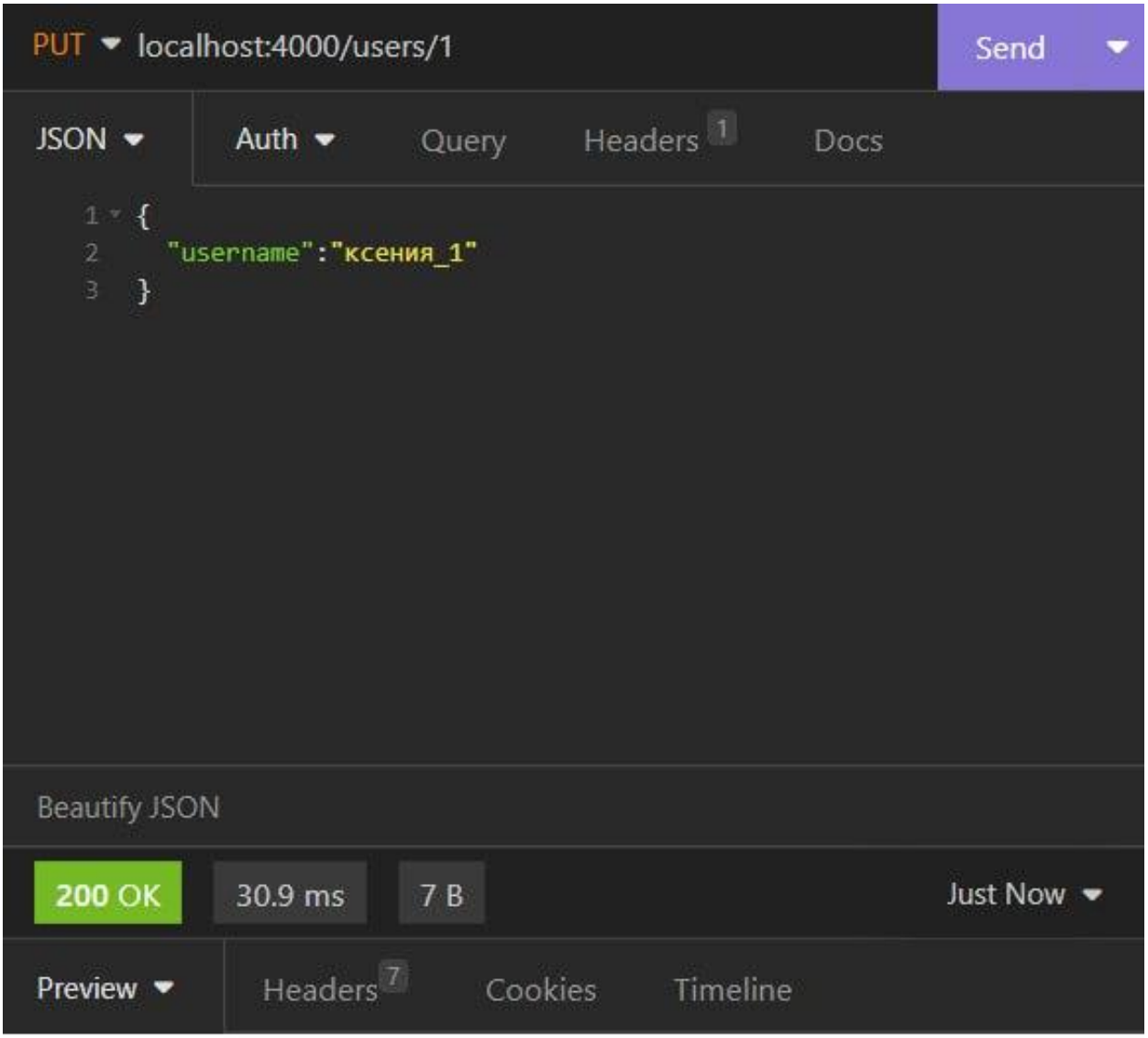

POST
localhost:4000/users
Send
JSON
Auth
Query
Headers 1
Docs
1 {
2 "username": "ксения",
3 "password": "ксепароль",
4 "email": "ксения@gmail.com"
5 }
Beautify JSON
200 OK
41.4 ms
16 B
Just Now
Preview
Headers 7
Cookies
Timeline
User is inserted

3) Чтение:


GET
localhost:4000/users
Send
Body
Auth
Query
Headers
Docs
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request
200 OK
5.38 ms
101 B
2 Minutes Ago
Preview
Headers 7
Cookies
Timeline
[
{
"id": 1,
"username": "ксения",
"password": "ксепароль",
"email": "ксения@gmail.com"
}
]

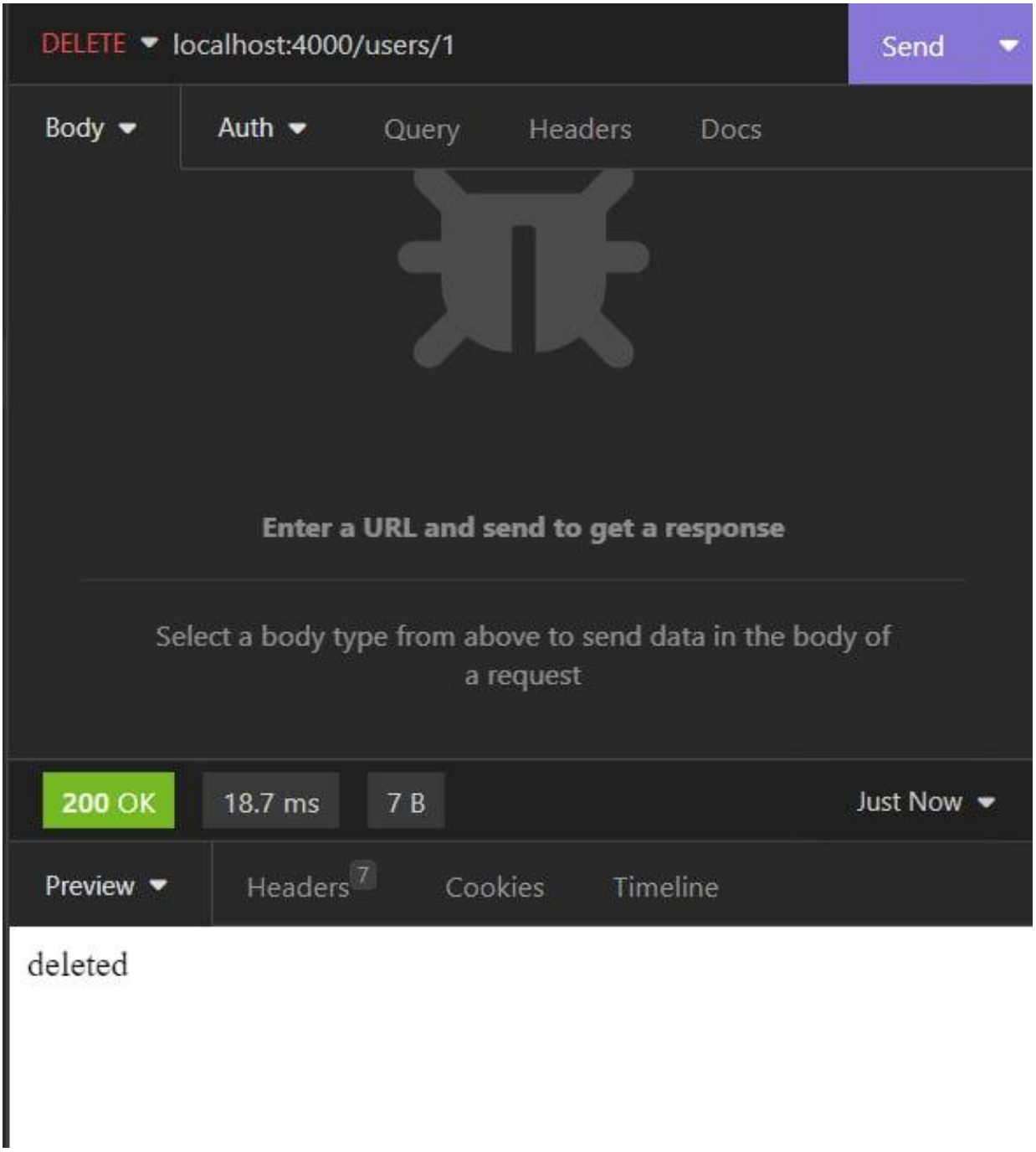

GET
localhost:4000/users/1
Send
Body
Auth
Query
Headers
Docs
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request
200 OK
21.1 ms
99 B
Just Now
Preview
Headers 7
Cookies
Timeline
{
"id": 1,
"username": "ксения",
"password": "ксепароль",
"email": "ксения@gmail.com"
}

4) Изменение:



Updated

5) Удаление:



Вывод:

В ходе выполнения задания была придумана собственная модель пользователя, реализован набор из CRUD-методов для работы с пользователями средствами Express и Sequelize, написан запрос для получения пользователя по id/email.