

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по домашней работе № 2 «Знакомство с ORM Sequelize»
по дисциплине «Бэкенд-разработка»

Автор: Власов М. И.

Факультет: ИКТ

Группа: К33402

Преподаватель: Добряков Д. И.

Дата: 18.03.22



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2022

1. Продумать свою собственную модель пользователя

```
matvey@matvey-Aspire-A315-55KG:~/ITMO-ICT-Backend-2022/homeworks/K33402/Vlasov_Matvey/HW2$ npx sequelize-cli model:generate --name User --attributes email:string,firstName:string,lastName:string,password:string
Sequelize CLI [Node: 16.13.0, CLI: 6.4.1, ORM: 6.17.0]

New model was created at /home/matvey/ITMO-ICT-Backend-2022/homeworks/K33402/Vlasov_Matvey/HW2/models/user.js .
New migration was created at /home/matvey/ITMO-ICT-Backend-2022/homeworks/K33402/Vlasov_Matvey/HW2/migrations/20220318070515-create-user.js .
```

```
'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({
    email: DataTypes.STRING,
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    password: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'User',
  });
};
```

2. Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize

```
app.post('/users/create', async (req, res) => {
  if(!req.body.email || !req.body.firstName || !req.body.lastName || !req.body.password) return res.sendStatus(400)
  const user = await db.User.create(req.body)
  res.send(user.toJSON())
})
```

POST ▼ http://127.0.0.1:3000/users/create Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	email	test2@example.com			
<input checked="" type="checkbox"/>	firstName	Test2			
<input checked="" type="checkbox"/>	lastName	Test2			
<input checked="" type="checkbox"/>	password	testpass			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 🌐 200 OK 61 ms 389 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 📄 🔍

```
1 {
2   "id": 2,
3   "email": "test2@example.com",
4   "firstName": "Test2",
5   "lastName": "Test2",
6   "updatedAt": "2022-03-18T08:08:05.908Z",
}
```

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

  if (user) {
    await user.update(req.body)
    res.send(user.toJSON())
  } else {
    res.send({"error": "user is not found"})
  }
})
```

PUT

http://127.0.0.1:3000/users/update/3

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	email	test3new@example.com			
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

200 OK 66 ms 392 B

Save Response

Pretty

Raw

Preview


Visualize

JSON

```
1  {
2    "id": 3,
3    "firstName": "test3",
4    "lastName": "test3",
5    "email": "test3new@example.com",
6    "createdAt": "2022-03-18T08:09:08.161Z",
```

```
app.delete('/users/delete/:id', async (req, res) => {
  const user = await db.User.findByPk(req.params.id)

  if (user) {
    await user.destroy()
    res.sendStatus(200)
  } else {
    res.send({"error": "user is not found"})
  }
})
```



DELETE  http://127.0.0.1:3000/users/delete/3

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

Body Cookies Headers (7) Test Results  200 OK 93 ms 229 B 

Pretty Raw Preview Visualize Text  

1 OK



GET  http://127.0.0.1:3000/users/3

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

Body Cookies Headers (7) Test Results  200 OK 15 ms 264 B 

Pretty Raw Preview Visualize JSON  

```
1 {  
2   "error": "user is not found"  
3 }
```

3. Написать запрос для получения пользователя по id/email

```
app.get('/users/:id', async (req, res) => {
  const user = await db.User.findByPk(req.params.id)

  if (user) {
    res.send(user.toJSON())
  } else {
    res.send({"error": "user is not found"})
  }
})

app.get('/users/email/:email', async (req, res) => {
  const user = await db.User.findOne({ where: { email: req.params.email } })

  if (user) {
    res.send(user.toJSON())
  } else {
    res.send({"error": "user is not found"})
  }
})
```

← → ↻ localhost:3000/users/1

{"id":1,"firstName":"Test1","lastName":"Test1","email":"test1@example.com","createdAt":"2022-03-18T08:07:21.514Z","updatedAt":"2022-03-18T08:07:21.514Z"}

← → ↻ localhost:3000/users/email/test2@example.com

{"id":2,"firstName":"Test2","lastName":"Test2","email":"test2@example.com","createdAt":"2022-03-18T08:08:05.908Z","updatedAt":"2022-03-18T08:08:05.908Z"}

Вывод: в ходе работы мы познакомились с ORM Sequelize и создали модель пользователя, а также набор CRUD-методов для работы с ним.