

# PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

Davi Schneid - [davi.Schneid@gmail.com](mailto:davi.Schneid@gmail.com)

23/07/2024

# Agenda

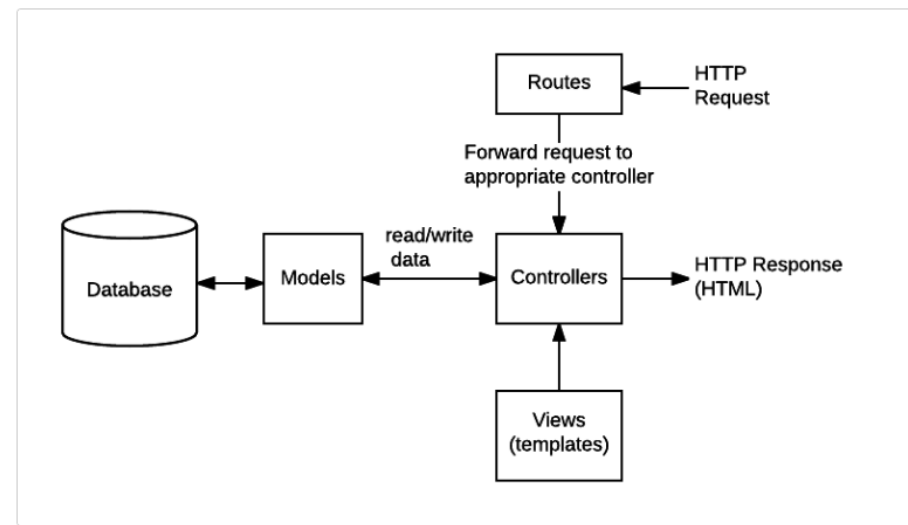
- ▶ Routers
  - ▶ Criar as rotas
- ▶ Controllers
  - ▶ Criar os controllers
- ▶ Estrutura do projeto
  - ▶ Analisar estrutura do projeto
- ▶ Swagger
  - ▶ Criar documentação para cada rota

# Routers

- ▶ Definem como uma aplicação responde ao HTTP
  - ▶ De acordo GET, POST, PUT, DELETE
  - ▶ Conexão (URI) e um método HTTP + function call-back
- ▶ Benefícios
  - ▶ Modularidade
  - ▶ Manutenibilidade
  - ▶ Reutilização

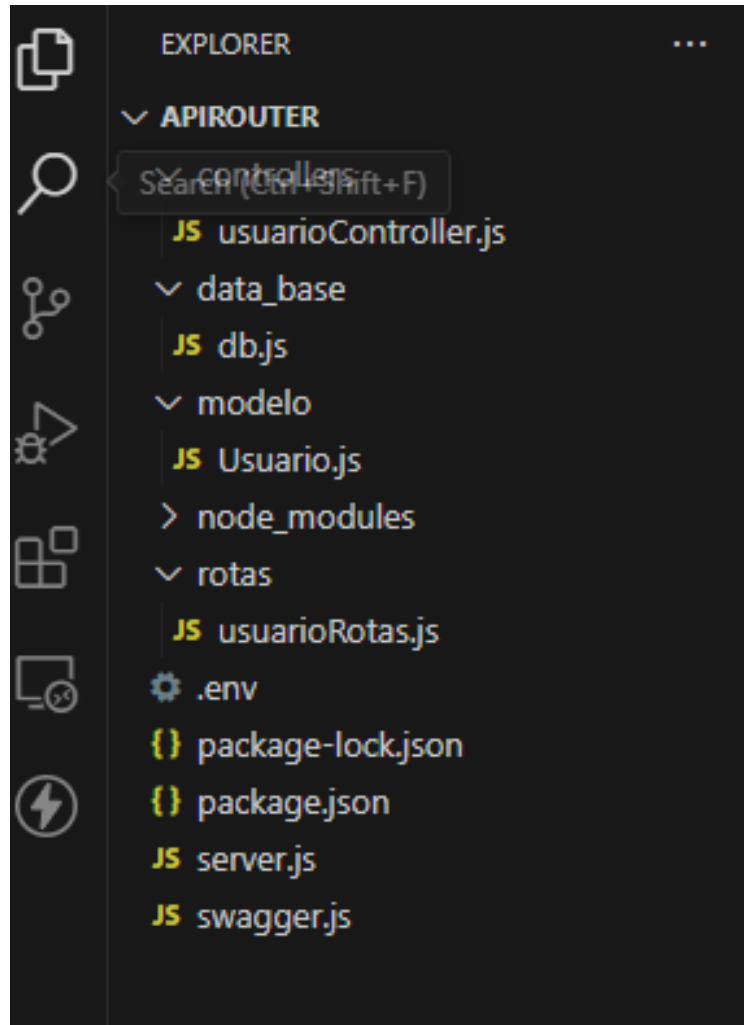
# Controller

- ▶ Gerenciar Requisições e Respostas
  - ▶ Recebem as requisições HTTP
  - ▶ Processam essas requisições
    - ▶ Validar dados + aplicar regra de negócio
    - ▶ Interagem com a camada de banco de dados
  - ▶ Retornam repostas aos clientes
- ▶ Interagir com Modelos
  - ▶ Acessam e manipulam dados
- ▶ Organização e Modularização
  - ▶ Código organizado
  - ▶ Separa lógico de negócio do roteamento
  - ▶ Manutenção
  - ▶ Escalabilidade



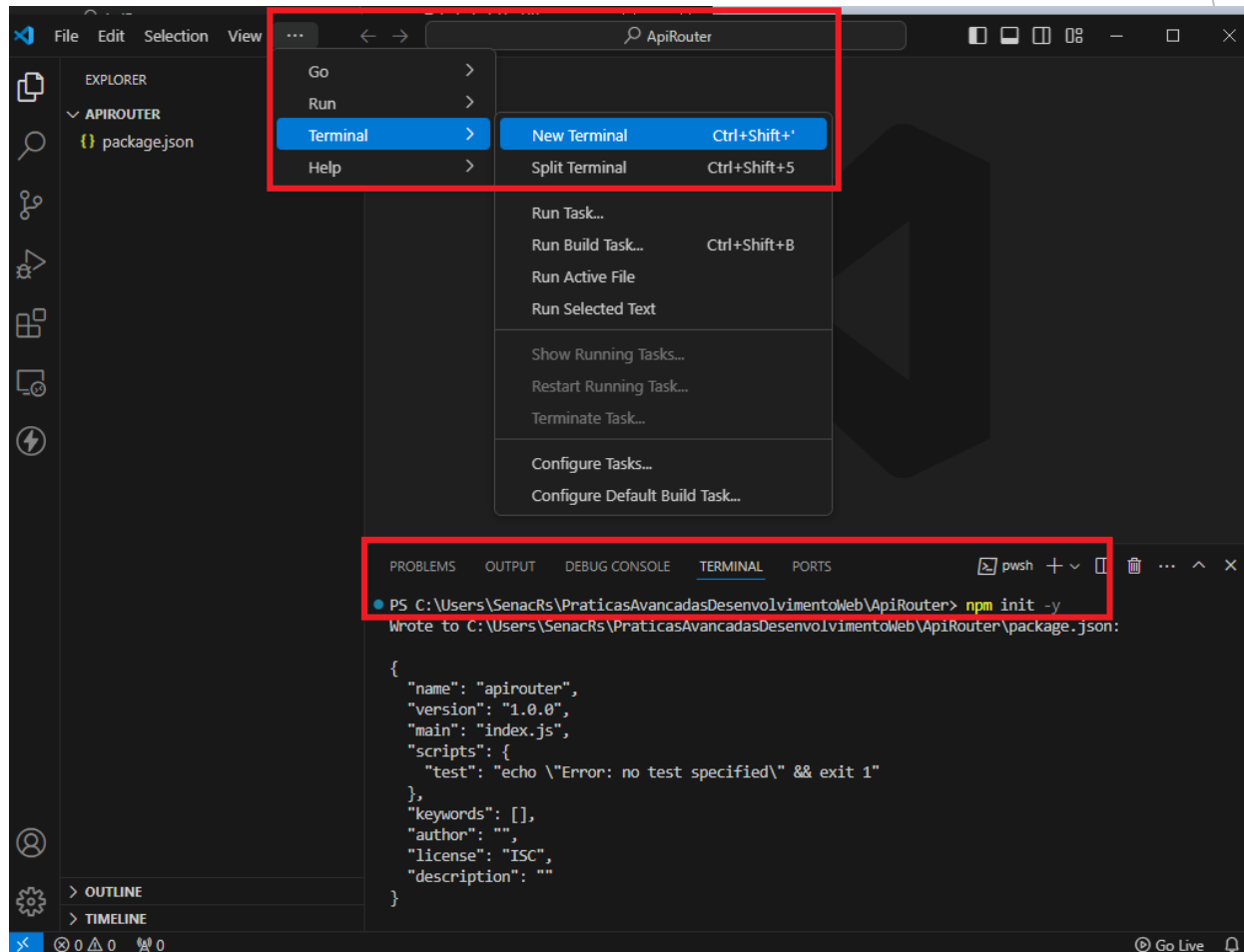
# API

- ▶ A estrutura da aplicação deve ficar assim



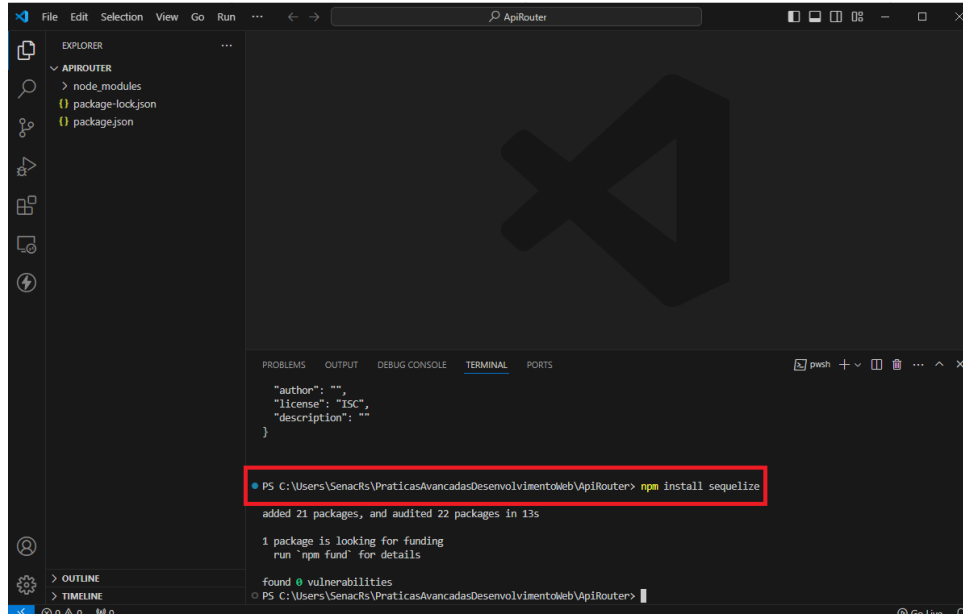
# API

- ▶ Criar uma nova pasta com nome ApiRouter no diretório de aplicações
- ▶ Acessar a pasta ApiRouter executar o comando: `npm init -y`



# API

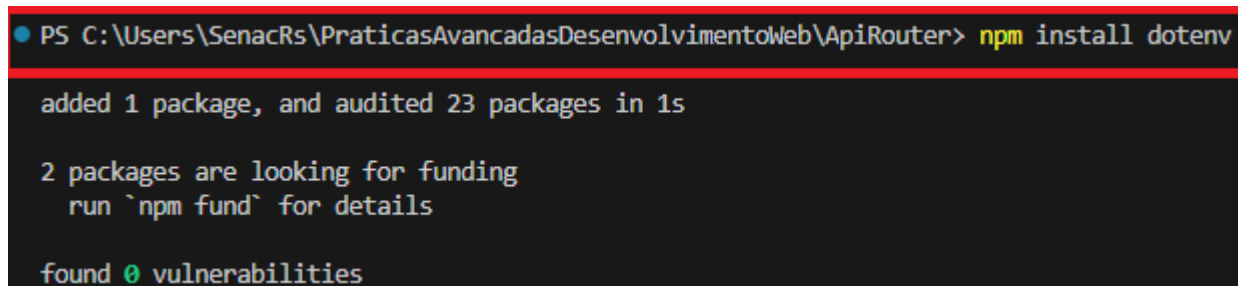
- ▶ Instalar o Sequelize, executar o comando: `npm install sequelize`



The screenshot shows the Visual Studio Code interface with a project named 'ApiRouter'. The Explorer sidebar on the left shows the file structure with 'node\_modules', 'package-lock.json', and 'package.json'. The Terminal panel at the bottom displays the command `npm install sequelize` being executed. The output shows that 21 packages were added and 22 packages were audited in 13 seconds. It also mentions that 1 package is looking for funding and that 0 vulnerabilities were found.

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install sequelize
added 21 packages, and audited 22 packages in 13s
1 package is looking for funding
run `npm fund` for details
found 0 vulnerabilities
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter>
```

- ▶ Instalar o Dotenv, executando o comando: `npm install dotenv`



The screenshot shows a terminal window with the command `npm install dotenv` being executed. The output shows that 1 package was added and 23 packages were audited in 1 second. It also mentions that 2 packages are looking for funding and that 0 vulnerabilities were found.

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install dotenv
added 1 package, and audited 23 packages in 1s
2 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
```

# API

- ▶ Criar o arquivo .env na raiz do projeto
- ▶ Instalar o Mysql no projeto executando o comando: `npm install mysql2`

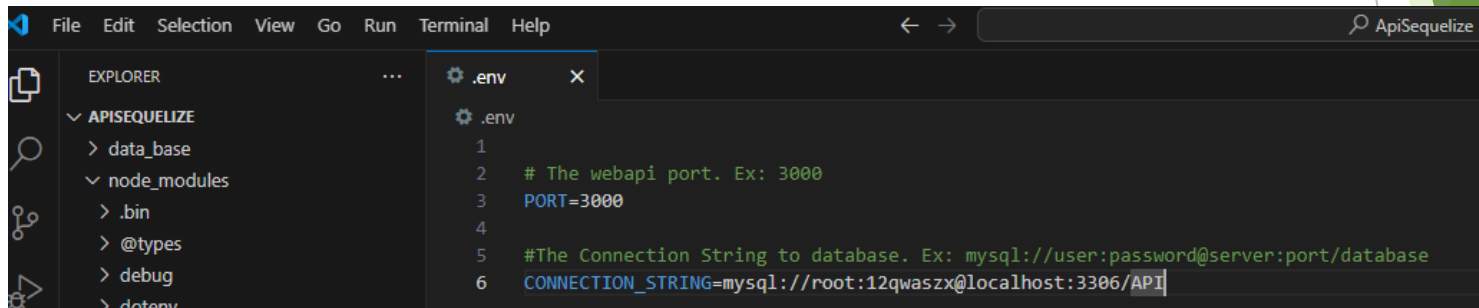
```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install mysql2

added 13 packages, and audited 36 packages in 3s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

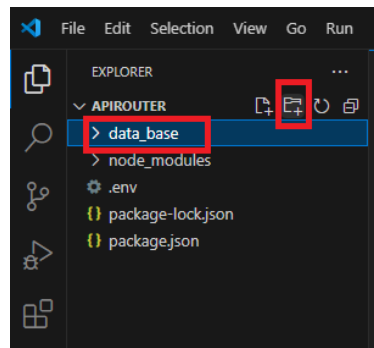
- ▶ Adicionar os parâmetros de conexão do Mysql dentro do arquivo



The screenshot shows the Visual Studio Code interface with the Explorer view on the left showing the project structure under 'APISEQUELIZE'. The main editor displays the '.env' file with the following content:

```
1
2 # The webapi port. Ex: 3000
3 PORT=3000
4
5 #The Connection String to database. Ex: mysql://user:password@server:port/database
6 CONNECTION_STRING=mysql://root:12qwaszx@localhost:3306/API
```

- ▶ Criar a pasta data\_base dentro da pasta ApiRouter



The screenshot shows the Visual Studio Code Explorer view. The 'APIROUTER' folder is expanded, and a new folder named 'data\_base' is being created, highlighted with a red box. The file explorer also shows the '.env', 'package-lock.json', and 'package.json' files.



# API

- ▶ Criar arquivo db.js dentro da pasta data\_base
  - ▶ Escrever o código abaixo

```
JS db.js  X
data_base > JS db.js > ...
1  //importa as configuracoes do arquivo de configuracao
2  require("dotenv").config();
3
4  //Importa o modulo do Sequelize
5  const Sequelize = require('sequelize');
6
7  const sequelize = new Sequelize(process.env.CONNECTION_STRING, {dialect: 'mysql'});
8
9  module.exports = sequelize;
```

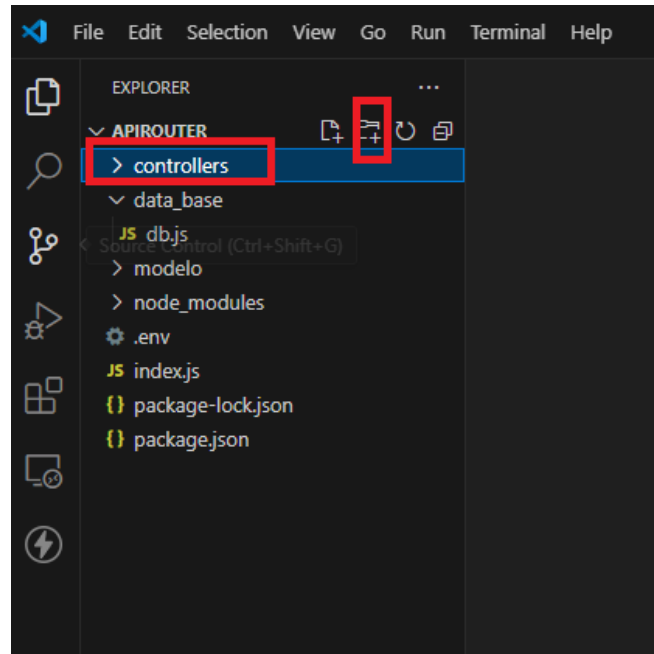
# API

- ▶ Criar uma pasta chamada modelo na raiz do projeto
  - ▶ Criar o arquivo usuario.js dentro da pasta modelo
  - ▶ Escrever o código abaixo

```
1
2  const Sequelize = require('sequelize');
3  const database = require('../data_base/db');
4
5  const Usuario = database.define('usuario', {
6    id: {
7      type: Sequelize.INTEGER,
8      autoIncrement: true,
9      allowNull: false,
10     primaryKey: true
11   },
12   nome: {
13     type: Sequelize.STRING,
14     allowNull: false
15   },
16   idade: {
17     type: Sequelize.INTEGER,
18     allowNull: false
19   },
20   cidade: {
21     type: Sequelize.STRING,
22     allowNull: false
23   }
24 }, {
25   // Configurações do modelo
26   timestamps: true, // Habilita createdAt e updatedAt
27   hooks: {
28     beforeCreate: (usuario, options) => {
29       const now = new Date();
30       const threeHoursLater = new Date(now.getTime() - 3 * 60 * 60 * 1000);
31       usuario.createdAt = threeHoursLater;
32       usuario.updatedAt = threeHoursLater;
33     },
34     beforeUpdate: (usuario, options) => {
35       const now = new Date();
36       const threeHoursLater = new Date(now.getTime() - 3 * 60 * 60 * 1000);
37       usuario.updatedAt = threeHoursLater;
38     }
39   }
40 }
41 ])
42 module.exports = Usuario;
```

# API

- ▶ Criar uma pasta controllers na raiz do projeto



- ▶ Criar o arquivo usuarioController.js dentro da pasta controllers

# API

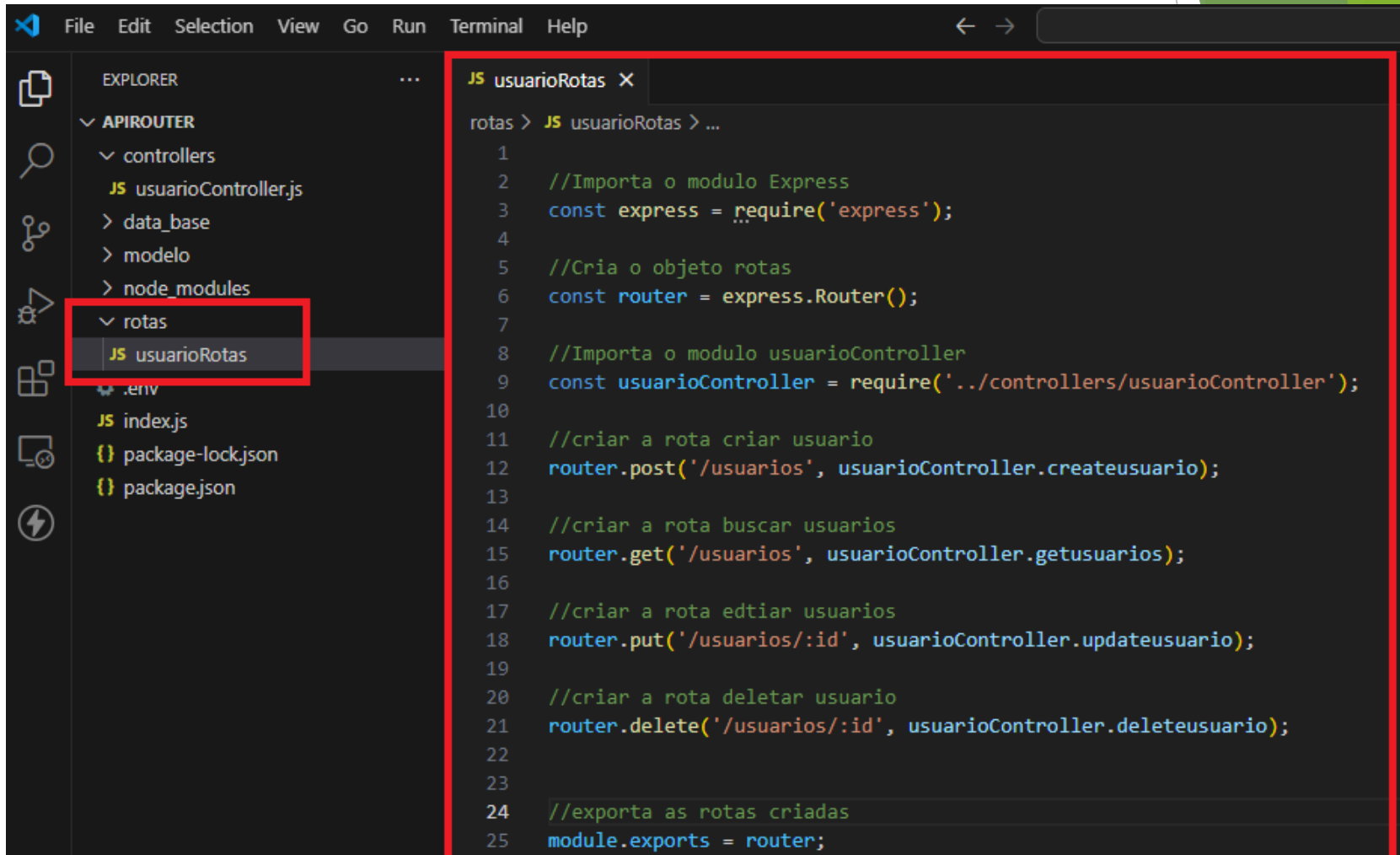
- Escrever o código abaixo no usuarioController.js

```
JS usuarioController.js X
controllers > JS usuarioController.js > ...

1
2  // Importa o objeto usuario
3  const Usuario = require('../modelo/Usuario');
4
5  // Criar um novo usuário
6  exports.createusuario = async (req, res) => {
7    const { nome, idade, cidade } = req.body;
8    try {
9      const novoUsuario = await Usuario.create({ nome, idade, cidade });
10     res.status(201).json(novoUsuario);
11   } catch (err) {
12     res.status(500).json({ error: 'Erro ao criar usuário' });
13   }
14 };
15
16 // Obter todos os usuários
17 exports.getusuarios = async (req, res) => {
18   try {
19     const usuarios = await Usuario.findAll();
20     res.status(200).json(usuarios);
21   } catch (err) {
22     res.status(500).json({ error: 'Erro ao obter usuários' });
23   }
24 };
25
```

# API

- ▶ Criar a pasta rotas na raiz da aplicação
- ▶ Criar o JS usuariosRotas e escreva o código abaixo



```
File Edit Selection View Go Run Terminal Help

EXPLORER
  APIROUTER
    controllers
      JS usuarioController.js
    > data_base
    > modelo
    > node_modules
    rotas
      JS usuarioRotas
  .env
  JS index.js
  {} package-lock.json
  {} package.json

JS usuarioRotas X
rotas > JS usuarioRotas > ...
1
2 //Importa o modulo Express
3 const express = require('express');
4
5 //Cria o objeto rotas
6 const router = express.Router();
7
8 //Importa o modulo usuarioController
9 const usuarioController = require('../controllers/usuarioController');
10
11 //criar a rota criar usuario
12 router.post('/usuarios', usuarioController.createusuario);
13
14 //criar a rota buscar usuarios
15 router.get('/usuarios', usuarioController.getusuarios);
16
17 //criar a rota edtiar usuarios
18 router.put('/usuarios/:id', usuarioController.updateusuario);
19
20 //criar a rota deletar usuario
21 router.delete('/usuarios/:id', usuarioController.deleteusuario);
22
23
24 //exporta as rotas criadas
25 module.exports = router;
```

# API

- ▶ Configurar o servidor
- ▶ Criar o arquivo server.js na raiz da aplicação, escrever o código abaixo

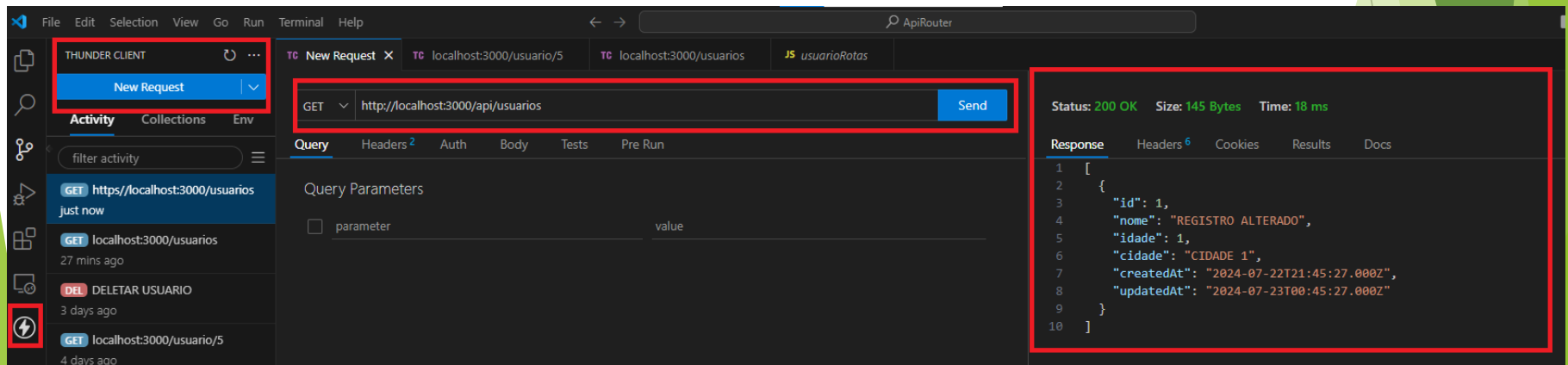
```
JS server.js  X  JS usuarioRotas  TC New Request
JS server.js > ...
1  const express = require('express');
2  const sequelize = require('./data_base/db');
3  const usuariosRotas = require('./rotas/usuarioRotas');
4
5  const app = express();
6  const PORT = process.env.PORT;
7
8  app.use(express.json());
9  app.use('/api', usuariosRotas);
10
11  sequelize.sync().then(() => {
12    app.listen(PORT, () => {
13      console.log(`Servidor rodando na porta ${PORT}`);
14    });
15  });
16
```

# API

- ▶ Inicializar a aplicação executando o comando: `node .\server.js`

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> node .\server.js
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'usuarios' AND TABLE_SCHEMA = 'API'
Executing (default): SHOW INDEX FROM `usuarios`
Servidor rodando na porta 3000
```

- ▶ Abrir o Thunder
  - ▶ Criar novo request buscar usuarios
  - ▶ Método get na URL: `http://localhost:3000/api/usuarios`



# API

- ▶ Criar request para rota criar usuário

- ▶ Método post na URL: <http://localhost:3000/api/usuarios>

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/api/usuarios`. The request body is a JSON object with the following content:

```
{
  "nome": "Davi Schneid",
  "idade": 43,
  "cidade": "Canoas"
}
```

The response status is `201 Created`, with a size of `137 Bytes` and a time of `50 ms`. The response body is a JSON object with the following content:

```
{
  "id": 3,
  "nome": "Davi Schneid",
  "idade": 43,
  "cidade": "Canoas",
  "updatedAt": "2024-07-23T08:05:05.986Z",
  "createdAt": "2024-07-23T08:05:05.986Z"
}
```

- ▶ Visualizar o log do servidor executando as operações

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> node .\server.js
Executing (default): SELECT TABLE NAME FROM INFORMATION SCHEMA.TABLES WHERE TABLE TYPE = 'BASE TABLE' AND TABLE NAME = 'usuarios' AND TABLE SCHEMA = 'API'
Executing (default): SHOW INDEX FROM `usuarios`
Servidor rodando na porta 3000
Executing (default): SELECT `id`, `nome`, `idade`, `cidade`, `createdAt`, `updatedAt` FROM `usuarios` AS `usuario`;
Executing (default): INSERT INTO `usuarios` (`id`,`nome`,`idade`,`cidade`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?);
```



# API

## ► Criar request para rota editar usuário

- Método put na URL: `http://localhost:3000/api/usuarios/1`

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** `http://localhost:3000/api/usuarios/1`
- Body (JSON):**

```
{  "nome": "REGISTRO ALTERADO",  "idade": 43,  "cidade": "POA"}
```
- Status:** 200 OK
- Size:** 139 Bytes
- Time:** 79 ms
- Response (JSON):**

```
{  "id": 1,  "nome": "REGISTRO ALTERADO",  "idade": 43,  "cidade": "POA",  "createdAt": "2024-07-22T21:45:27.000Z",  "updatedAt": "2024-07-23T11:50:47.087Z"}
```

## ► Criar request para rota deletar usuário

- Método delete na URL: `http://localhost:3000/api/usuarios/3`

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** `http://localhost:3000/api/usuarios/3`
- Status:** 204 No Content
- Size:** 0 Bytes
- Time:** 24 ms
- Response:** (Empty response body)

# API

- ▶ Instalar Swagger na aplicação
  - ▶ Executar o comando: `npm install swagger-ui-express`

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install swagger-ui-express
added 2 packages, and audited 108 packages in 4s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- ▶ Executar o comando: `npm install swagger-jsdoc`

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiRouter> npm install swagger-jsdoc
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check
ve and powerful.
npm warn deprecated glob@7.1.6: Glob versions prior to v9 are no longer supported

added 31 packages, and audited 139 packages in 11s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

# API

- ▶ Criar o arquivo swagger.js na raiz da aplicação
  - ▶ Escrever o código abaixo

```
JS swagger.js > ...
1
2 //importa o modulo do swagger jsdoc
3 const swaggerJsdoc = require('swagger-jsdoc');
4
5 //importa o modulo com a interface grafica do swagger
6 const swaggerUi = require('swagger-ui-express');
7
8 const options = {
9   definition: {
10     openapi: '3.0.0',
11     info: {
12       title: 'APIROUTER',
13       version: '1.0.0',
14       description: 'Uma aplicacao com as rotas usando Sequelize',
15     },
16     servers: [
17       {
18         url: 'http://localhost:3000',
19       },
20     ],
21   },
22   // Caminho para os arquivos de rotas
23   apis: ['./rotas/*.js'],
24 };
25
26 const swaggerSpec = swaggerJsdoc(options);
27
28 const setupSwagger = (app) => {
29   app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));
30 };
31
32 module.exports = setupSwagger;
```

# API

- Importar o módulo Swagger no server.js

```
JS server.js > ...
1
2  const express = require('express');
3  const sequelize = require('./data_base/db');
4  const usuariosRotas = require('./rotas/usuarioRotas');
5
6
7  const setupSwagger = require('./swagger');
8
9  const app = express();
10 const PORT = process.env.PORT;
11
12 app.use(express.json());
13 app.use('/api', usuariosRotas);
14
15
16 // Configurar Swagger
17 setupSwagger(app);
18
19 sequelize.sync().then(() => {
20   app.listen(PORT, () => {
21     console.log(`Servidor rodando na porta ${PORT}`);
22   });
23 });
```

# API

- ▶ Adicionar a documentação no usuarioRotas.js de acordo com a rota.
  - ▶ Rota post

```
10
11 //criar a rota criar usuario
12
13 /**
14  * @swagger
15  * /usuarios:
16  *   post:
17  *     summary: Cria um novo usuário
18  *     tags: [Usuario]
19  *     requestBody:
20  *       required: true
21  *       content:
22  *         application/json:
23  *           schema:
24  *             type: object
25  *             properties:
26  *               nome:
27  *                 type: string
28  *               idade:
29  *                 type: integer
30  *               cidade:
31  *                 type: string
32  *             responses:
33  *               201:
34  *                 description: Usuario criado
35  *               500:
36  *                 description: Erro ao criar usuario
37  */
38 router.post('/usuarios', usuarioController.createusuario);
39
```

# Swagger

## Usuário

POST

/usuarios

Cria um novo usuário

Parameters

Cancel

No parameters

Request body

required

application/json

```
{  "nome": "string",  "idade": 0,  "cidade": "string"}  
```

Execute

Clear

### Responses

Curl

```
curl -X 'POST' \  'http://localhost:3000/api/usuarios' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "nome": "string",  "idade": 0,  "cidade": "string"}'
```

Request URL

```
http://localhost:3000/api/usuarios
```

Server response

Code

Details

201

Response body

```
{  "id": 0,  "nome": "string",  "idade": 0,  "cidade": "string",  "updatedAt": "2024-07-23T13:25:43.636Z",  "createdAt": "2024-07-23T13:25:43.636Z"}  
```

Download

# API

- Adicionar a documentação na rota get/usuarios

```
42 //criar a rota buscar usuarios
43 /**
44  * @swagger
45  * tags:
46  *   name: Usuarios
47  *   description: Busca todos os usuários
48  */
49
50 /**
51  * @swagger
52  * /usuarios:
53  *   get:
54  *     summary: Retorna a lista de todos os usuários
55  *     tags: [Usuarios]
56  *     responses:
57  *       200:
58  *         description: A lista de usuários
59  *         content:
60  *           application/json:
61  *             schema:
62  *               type: array
63  *               items:
64  *                 type: object
65  */
66 router.get('/usuarios', usuarioController.getusuarios);
```

# Swagger

## Usuarios

Busca todos os usuários

GET

/usuarios

Retorna a lista de todos os usuários

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3000/api/usuarios' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:3000/api/usuarios
```

Server response

Code

Details

200

Response body

```
{
  "id": 4,
  "nome": "XXXXXXXXXXXXXXXXXXXX",
  "idade": 43,
  "cidade": "CanoasXXXXXXXXXX",
  "createdAt": "2024-07-23T08:12:50.000Z",
  "updatedAt": "2024-07-23T11:47:19.000Z",
},
{
  "id": 5,
  "nome": "DDDDDDZZZZZZ",
  "idade": 43,
  "cidade": "DDDDDDZZZZZZ",
  "createdAt": "2024-07-23T08:48:20.000Z",
  "updatedAt": "2024-07-23T11:48:45.000Z",
},
{
  "id": 6,
  "nome": "REGISTRO ALTERADO AGORA",
  "idade": 43,
  "cidade": "POA",
  "createdAt": "2024-07-23T13:17:23.000Z",
  "updatedAt": "2024-07-23T13:17:23.000Z",
},
{
  "id": 7,
  "nome": "JJJJJJJJJJ",
  "idade": 43
}
```

Download

Response headers

```
connection: keep-alive
content-length: 843
content-type: application/json; charset=utf-8
date: Tue, 23 Jul 2024 16:28:00 GMT
etag: W/"34b-GfB3jOLYRULpxCA0ka/hK8+HePY"
keep-alive: timeout=5
x-powered-by: Express
```

Responses



# API

- Adicionar a documentação na rota put/usuarios/:id

```
74 //criar a rota editar usuarios
75 /**
76  * @swagger
77  * /usuarios/{id}:
78  *   put:
79  *     summary: Atualiza um usuário existente
80  *     tags: [Usuario]
81  *     parameters:
82  *       - in: path
83  *         name: id
84  *         schema:
85  *           type: string
86  *           required: true
87  *           description: ID do usuário
88  *     requestBody:
89  *       required: true
90  *       content:
91  *         application/json:
92  *           schema:
93  *             type: object
94  *             properties:
95  *               nome:
96  *                 type: string
97  *               idade:
98  *                 type: integer
99  *               cidade:
100  *                 type: string
101  *     responses:
102  *       200:
103  *         description: Usuário atualizado
104  *       404:
105  *         description: Usuário não encontrado
106  *       500:
107  *         description: Erro ao atualizar usuário
108  *     /
109 router.put('/usuarios/:id', usuarioController.updateusuario);
```

# Swagger

**PUT** /usuarios/{id} Atualiza um usuário existente

Parameters

Cancel

Reset

Name	Description
id * required	ID do usuário
string (path)	<input type="text" value="8"/>

Request body required

application/json

```
{  "nome": "ALTERADO",  "idade": 1,  "cidade": "ALTERADO"}
```

Execute

Clear

Responses

Curl

```
curl -X 'PUT' \  'http://localhost:3000/api/usuarios/8' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "nome": "ALTERADO",  "idade": 1,  "cidade": "ALTERADO"  }'
```

Request URL

```
http://localhost:3000/api/usuarios/8
```

Server response

Code	Details
------	---------

# API

- Adicionar a documentação na rota delete/usuarios/:id

```
106 //criar a rota deletar usuario
107 /**
108  * @swagger
109  * /usuarios/{id}:
110  *   delete:
111  *     summary: Deleta um usuário existente
112  *     tags: [Usuario]
113  *     parameters:
114  *       - in: path
115  *         name: id
116  *         schema:
117  *           type: string
118  *           required: true
119  *           description: ID do usuário
120  *     responses:
121  *       204:
122  *         description: Usuário deletado
123  *       404:
124  *         description: Usuário não encontrado
125  *       500:
126  *         description: Erro ao deletar usuário
127  */
128 router.delete('/usuarios/:id', usuarioController.deleteusuario);
```

# Swagger

**DELETE** /usuarios/{id} Deleta um usuário existente

Parameters

Name	Description
<b>id</b> * required string (path)	ID do usuário

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:3000/api/usuarios/9' \
-H 'accept: */*'
```

Request URL

```
http://localhost:3000/api/usuarios/9
```

Server response

Code	Details
204	<div><div>Response headers</div><pre>connection: keep-alive date: Tue, 23 Jul 2024 16:37:35 GMT keep-alive: timeout=5 x-powered-by: Express</pre></div>

Responses

Code	Description	Links
204	Usuário deletado	No links
404	Usuário não encontrado	No links
500	Erro ao deletar usuário	No links