

# PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

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

22/07/2024

# Agenda

- ▶ Object-Relational Mapping Sequelizeize -> <https://sequelize.org/>



## Sequelize

Sequelize is a modern TypeScript and Node.js ORM for Oracle, Postgres, MySQL, MariaDB, SQLite and SQL Server, and more. Featuring solid transaction support, relations, eager and lazy loading, read replication and more.

[Getting Started](#)[API Reference](#)[Upgrade to v6](#)[Support us](#)

- ▶ Criar API com Sequelize.
- ▶ Criar funcionalidades
  - ▶ Criar tabela
  - ▶ Inserir registros
  - ▶ Buscar registros
  - ▶ Editar registros
  - ▶ Excluir registros

# O que é Sequelize?

- ▶ Object-Relational Mapping
- ▶ Projetado para bancos de dados SQL.
- ▶ MySQL, PostgreSQL, SQLite e MSSQL.
- ▶ Cada BD tem seu dialect
  - ▶ 

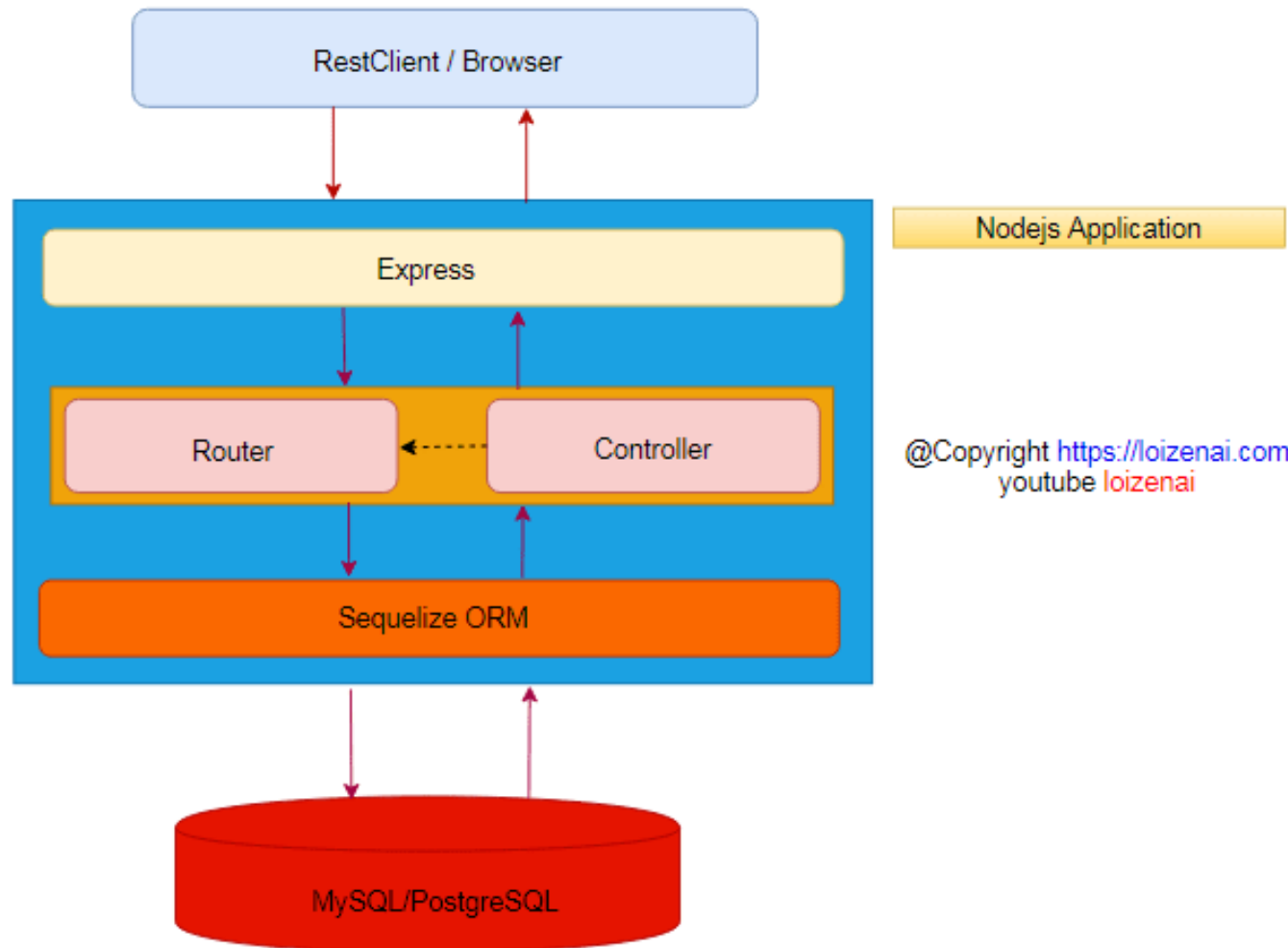
```
const sequelize = new Sequelize(process.env.CONNECTION_STRING, {dialect: 'mysql'});
```
  - ▶ 

```
const sequelize = new Sequelize(process.env.CONNECTION_STRING, {dialect: 'postgres'});
```
- ▶ Não é compatível com bancos de dados NoSQL

# Principais recursos

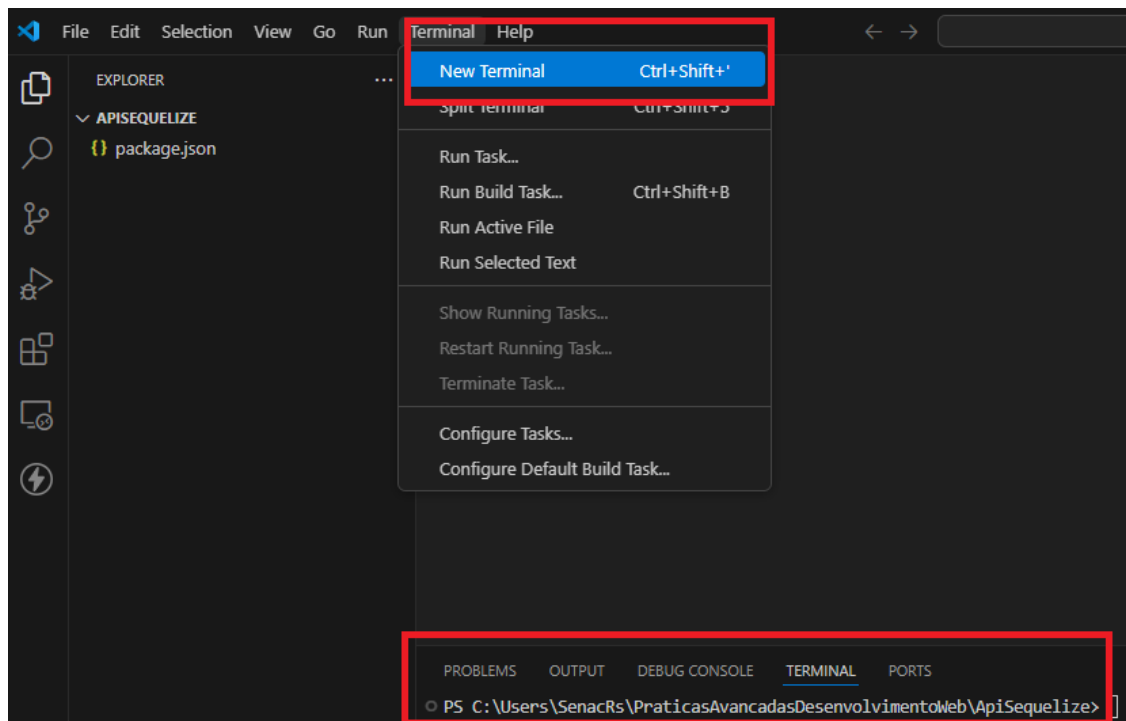
- ▶ **Modelos:** Representações das tabelas do banco de dados.
- ▶ **Consultas:** Métodos para criar, ler, atualizar e excluir registros.
- ▶ **Associações:** Relacionamentos entre tabelas (one-to-one, one-to-many, many-to-many)
- ▶ **Validações e Restrições:** Verificações automáticas de integridade dos dados.
- ▶ **Hooks:** Funções de callback executadas em determinados momentos do ciclo de vida do modelo.
- ▶ **Migrations:** Controle de versão do esquema do banco de dados.

# Arquitetura com Sequelize



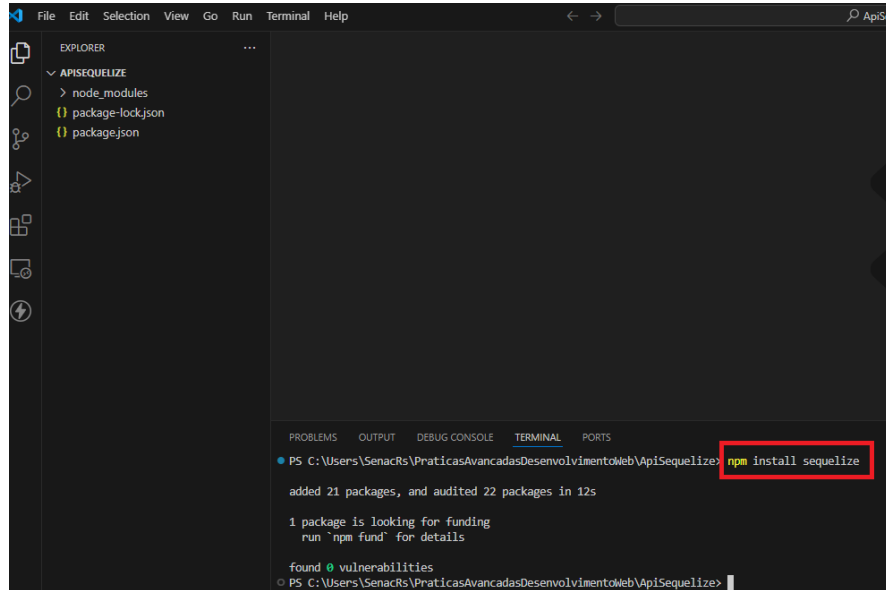
# API

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



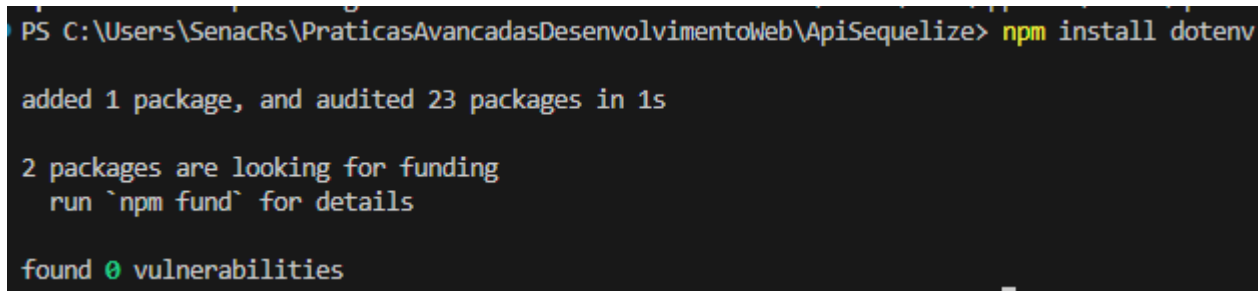
# API

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



A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'APISEQUELIZE' with a 'node\_modules' folder and 'package-lock.json' and 'package.json' files. The Terminal panel at the bottom shows the command `npm install sequelize` being executed. The output indicates that 21 packages were added and audited in 12 seconds, and that 1 package is looking for funding. The command prompt is currently at `PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiSequelize>`.

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



A screenshot of a terminal window with a black background and green text. The prompt is `PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiSequelize>`. The command `npm install dotenv` has been entered. The output shows that 1 package was added and audited in 1 second, and that 2 packages are looking for funding. The terminal ends with the message `found 0 vulnerabilities`.

# API

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

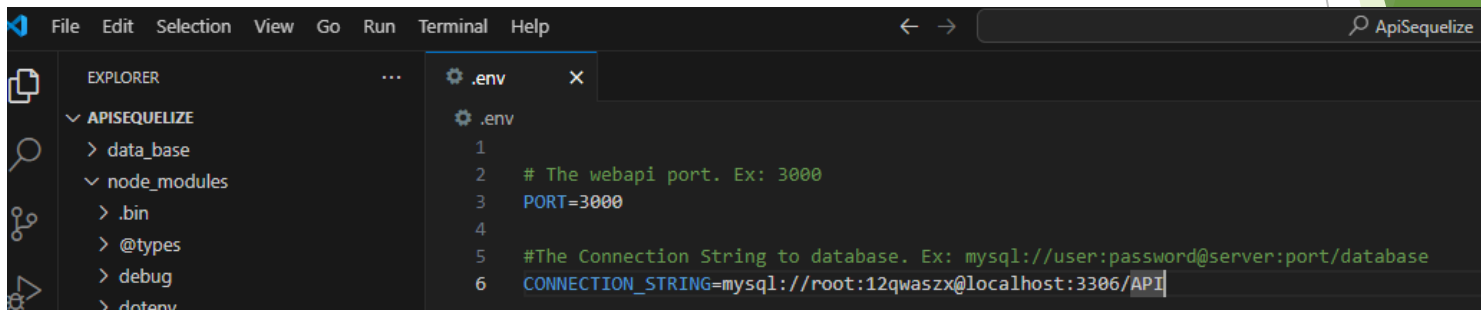
```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiSequelize> npm install mysql2

added 13 packages, and audited 36 packages in 5s

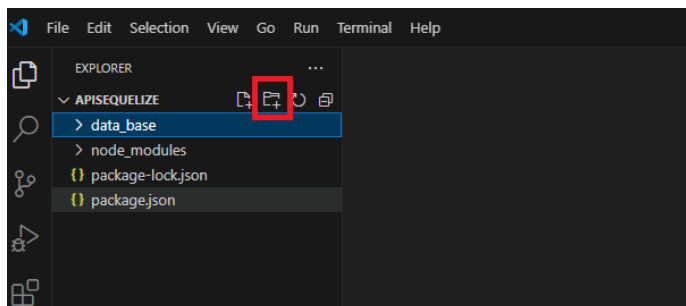
2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

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



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





# 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

```
JS usuario.js X
modelo > JS usuario.js > [e] Usuario
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: false // Desativando createdAt e updatedAt
27 });
28
29 module.exports = Usuario;
```

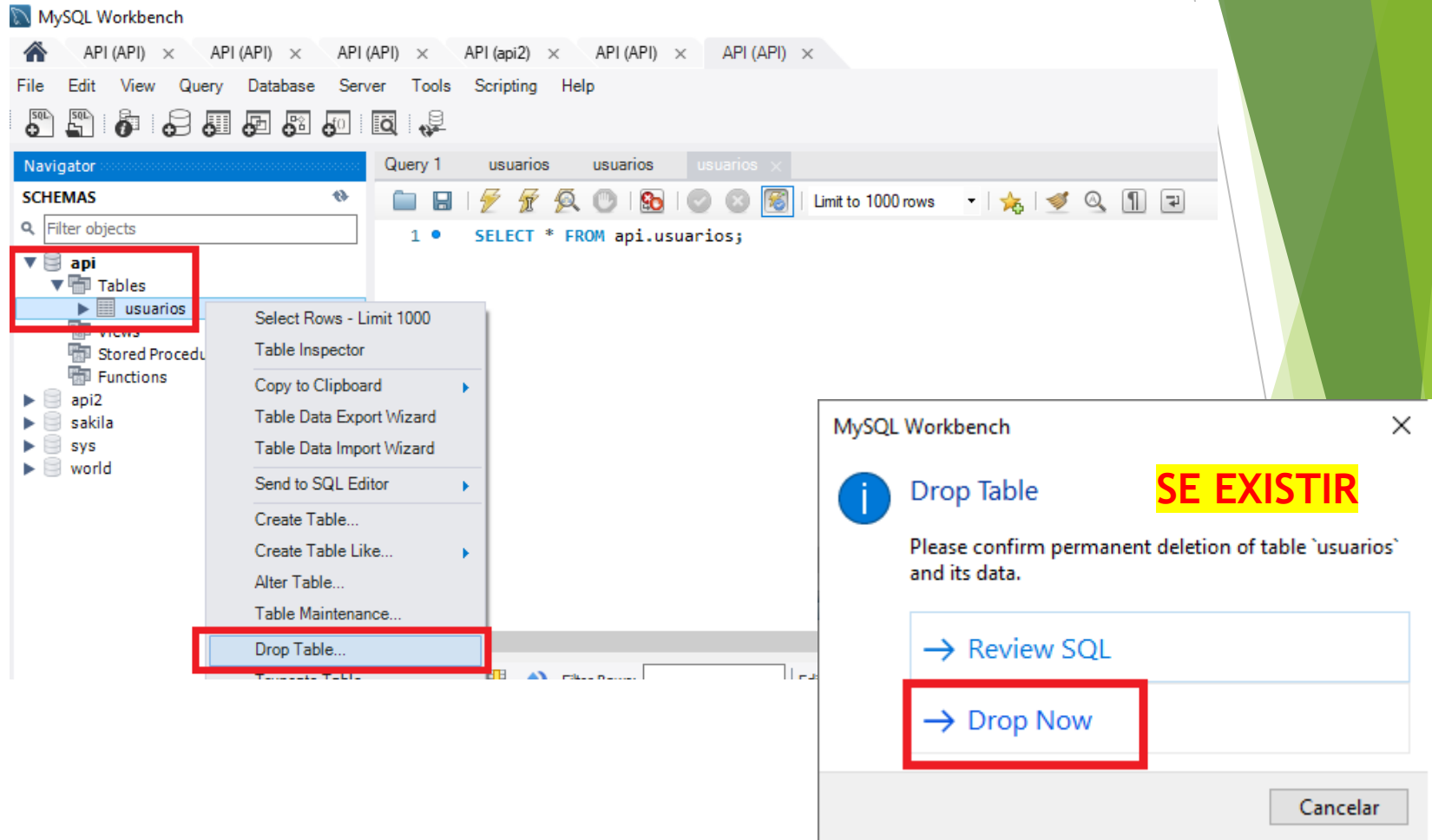
# API

- ▶ Criar arquivo index.js na raiz do projeto
- ▶ Escrever o código abaixo

```
JS index.js > ...
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejrãp mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5     const database = require('./data_base/db');
6     const Usuario = require('./modelo/Usuario');
7
8     try {
9         const resultado = await database.sync();
10        console.log(resultado);
11    } catch (error) {
12        console.log(error);
13    }
14 })();
15
```

# Mysql Workbench

- ▶ Abrir o Mysql Workbench
- ▶ Conectar no schema API



# API

- ▶ Executar a aplicação com o comando: `node .\index.js`

```
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiSequelize> node .\index.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`
<ref *1> Sequelize {
  options: {
    dialect: 'mysql',
    dialectModule: null,
    dialectModulePath: null,
    host: 'localhost',
    protocol: 'tcp',
    define: {},
    query: {},

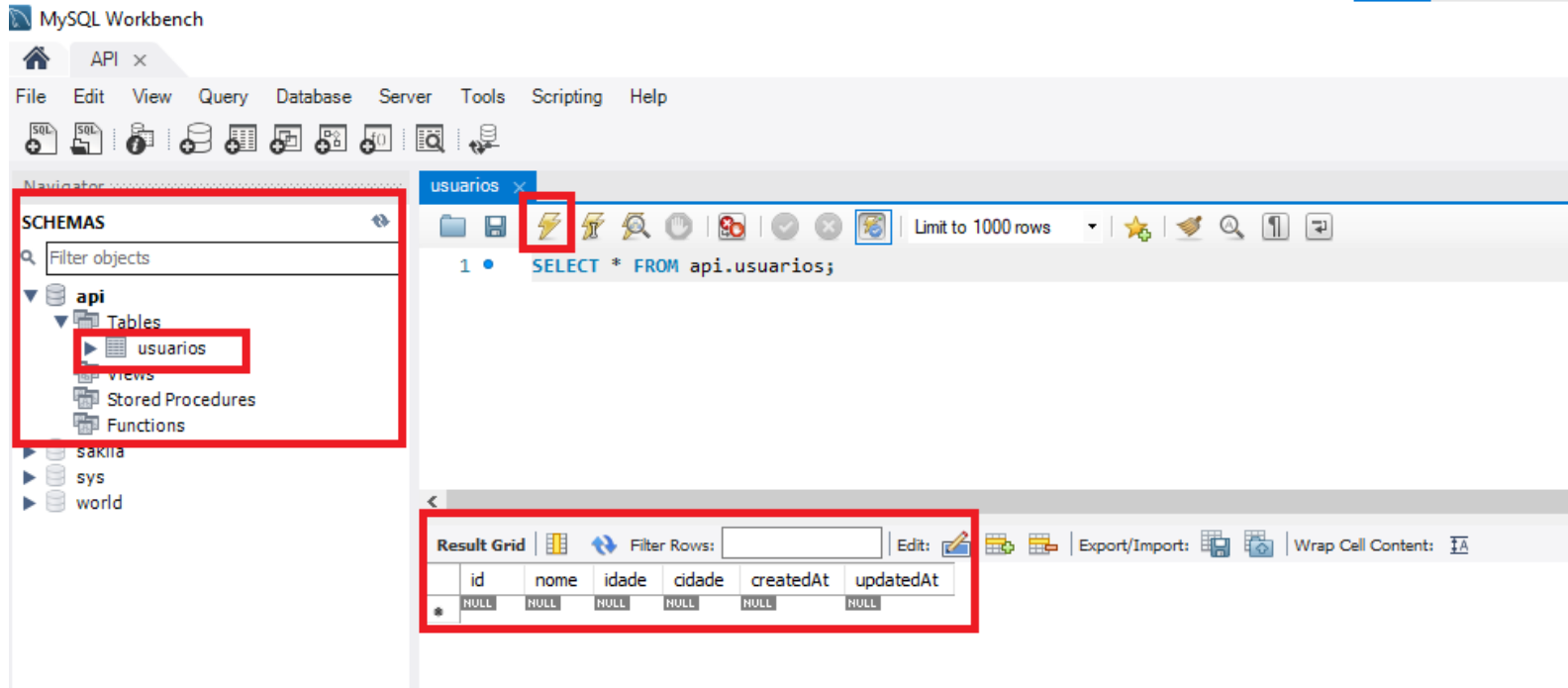
```

Continua ...

```
    },
    lib: {
      createConnection: [Function (anonymous)],
      connect: [Function (anonymous)],
      Connection: [class Connection extends EventEmitter],
      ConnectionConfig: [class ConnectionConfig],
      createPool: [Function (anonymous)],
      createPoolCluster: [Function (anonymous)],
      createQuery: [Function: createQuery],
      Pool: [class Pool extends EventEmitter],
      PoolCluster: [class PoolCluster extends EventEmitter],
      createServer: [Function (anonymous)],
      PoolConnection: [Function],
      authPlugins: [Object],
      escape: [Function: escape],
      escapeId: [Function: escapeId],
      format: [Function: format],
      raw: [Function: raw],
      createConnectionPromise: [Getter],
      createPoolPromise: [Getter],
      createPoolClusterPromise: [Getter],
      Types: [Getter],
      Charsets: [Getter],
      CharSetToEncoding: [Getter],
      setMaxParserCache: [Function (anonymous)],
      clearParserCache: [Function (anonymous)]
    }
  }
}
PS C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\ApiSequelize>
```

# Mysql Workbench

- Validar a criação da tabela pelo Sequelize



- Criou a tabela
- Default createdAt , updatedAt

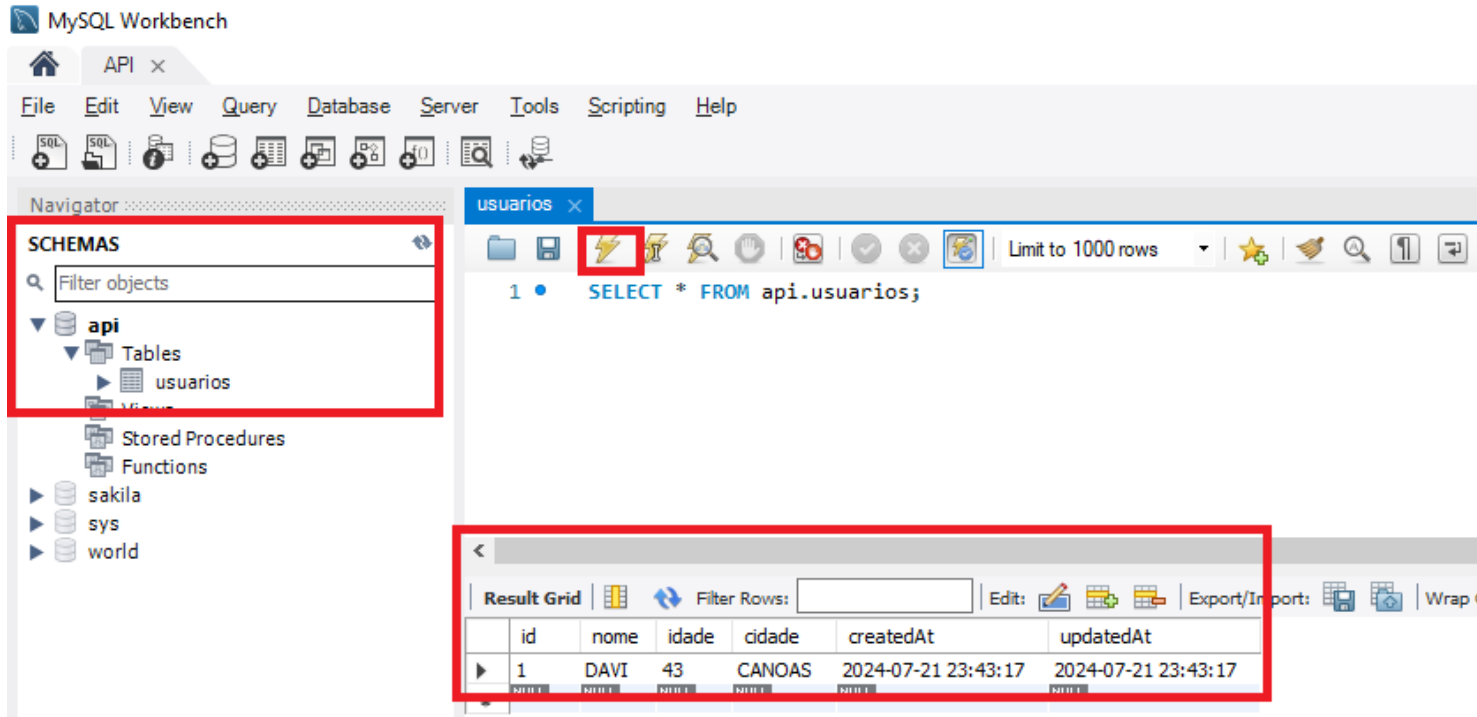
# API

- ▶ Inserir registro usando o Sequelize
- ▶ Adicionar o código destacado em vermelho no arquivo index.js

```
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejrapp mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5     const database = require('./data_base/db');
6     const Usuario = require('./modelo/Usuario');
7
8     try {
9         const resultado = await database.sync();
10        console.log(resultado);
11    } catch (error) {
12        console.log(error);
13    }
14
15    //insere registro via Sequelize
16    const resultadoCreate = await Usuario.create({
17        nome: 'REGISTRO 1',
18        idade: 1,
19        cidade: 'CIDADE 1'
20    })
21    console.log(resultadoCreate);
22
23
24 })();
25
```

# Mysql Workbench

- ▶ Validar a inserção do registro na tabela



- ▶ Registro criado com valor createdAt e updateAt de forma automática





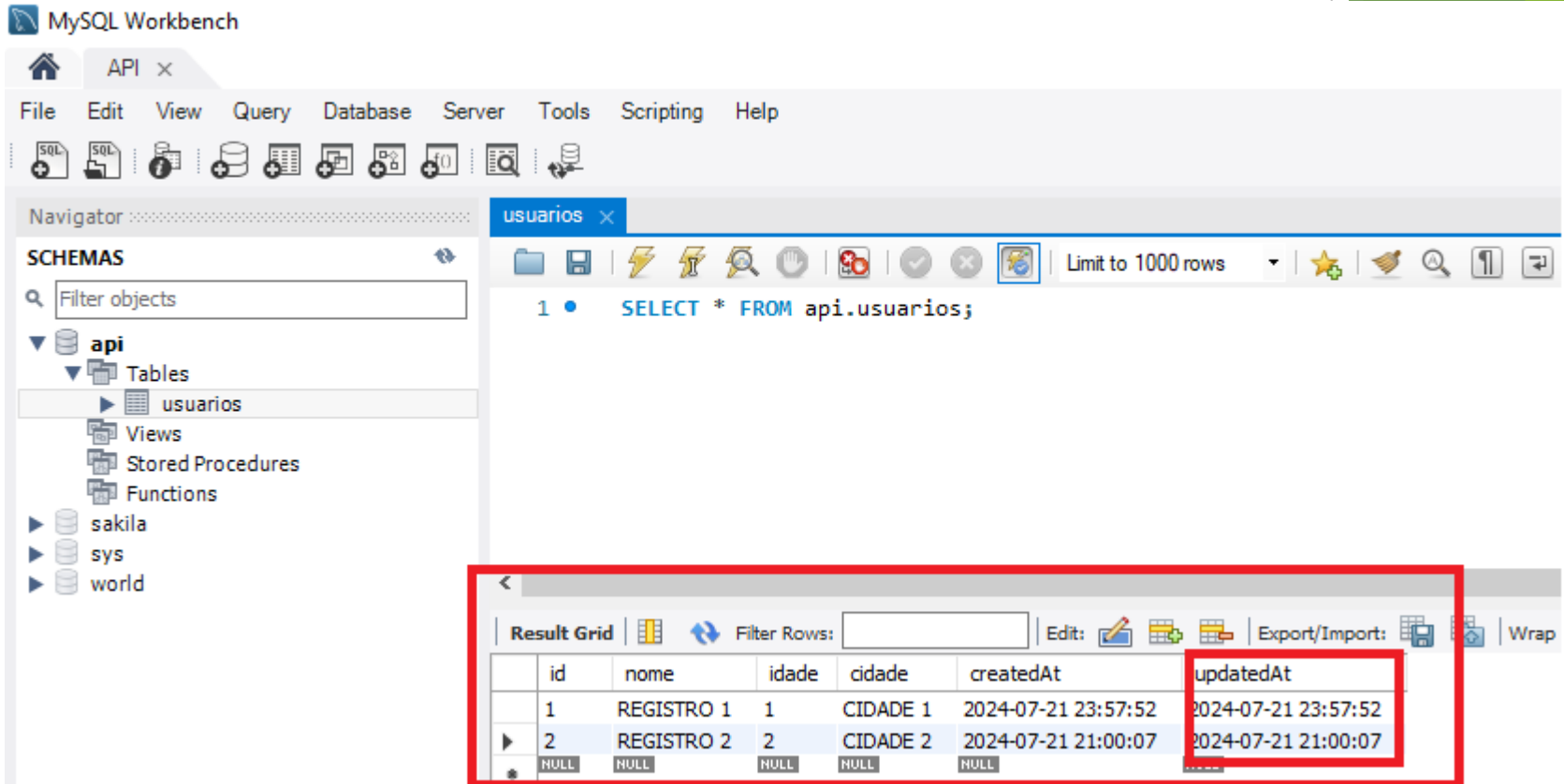
# API

- ▶ Ajustar o GMT no modelo Usuario
- ▶ Escrever o código destacado em vermelho no arquivo usuario.js
- ▶ Executar o index.js

```
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: async (user, options) => {
29       const threeHoursLater = new Date(
30         new Date().getTime() - 3 * 60 * 60 * 1000);
31       user.createdAt = threeHoursLater;
32       user.updatedAt = threeHoursLater;
33     },
34     beforeUpdate: (user, options) => {
35       const now = new Date();
36       const threeHoursLater = new Date(now.getTime() - 3 * 60 * 60 * 1000);
37       user.updatedAt = threeHoursLater;
38     }
39   }
40 }, {
41 })
42
43 module.exports = Usuario;
```

# Mysql Workbench

- Validar a inserção do registro na tabela com horário correto



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'api' selected, showing its tables, views, stored procedures, and functions. The main query editor shows the query: `SELECT * FROM api.usuarios;`. The 'Result Grid' at the bottom displays the query results. The 'updatedAt' column is highlighted with a red box.

	id	nome	idade	cidade	createdAt	updatedAt
1	1	REGISTRO 1	1	CIDADE 1	2024-07-21 23:57:52	2024-07-21 23:57:52
2	2	REGISTRO 2	2	CIDADE 2	2024-07-21 21:00:07	2024-07-21 21:00:07
	NULL	NULL	NULL	NULL	NULL	.....

# API

- ▶ Buscar os registro inseridos
- ▶ Adicionar o código na index.js

```
JS index_v2.js > ...
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejrãp mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5     const database = require('./data_base/db');
6     const Usuario = require('./modelo/Usuario');
7
8     try {
9         const resultado = await database.sync();
10        console.log(resultado);
11    } catch (error) {
12        console.log(error);
13    }
14
15    //insere registro via Sequelize
16    const resultadoCreate = await Usuario.create({
17        nome: 'REGISTRO 3',
18        idade: 3,
19        cidade: 'CIDADE 3'
20    })
21    console.log(resultadoCreate);
22
23
24    //buscar os registros inseridos
25    const usuarios = await Usuario.findAll();
26    console.log(usuarios);
27
28 })();
29
```

# API

## ► Adicionar a busca por registros

```
JS index_v2.js > ...
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejrãp mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5   const database = require('./data_base/db');
6   const Usuario = require('./modelo/Usuario');
7
8   try {
9     const resultado = await database.sync();
10    console.log(resultado);
11  } catch (error) {
12    console.log(error);
13  }
14
15  //insere registro via Sequelize
16  const resultadoCreate = await Usuario.create({
17    nome: 'REGISTRO 3',
18    idade: 3,
19    cidade: 'CIDADE 3'
20  })
21  console.log(resultadoCreate);
22
23
24  //buscar os registros inseridos
25  const usuarios = await Usuario.findAll();
26  console.log(usuarios);
27
28  })();|
29
30
```

# API

## ► Resultado da busca por registros

```
Executing (default): SELECT `id`, `nome`, `idade`, `cidade`, `createdAt`, `updatedAt` FROM `usuarios` AS `usuario`;  
[  
  usuario {  
    dataValues: {  
      id: 1,  
      nome: 'REGISTRO 1',  
      idade: 1,  
      cidade: 'CIDADE 1',  
      createdAt: 2024-07-21T21:18:39.000Z,  
      updatedAt: 2024-07-21T21:18:39.000Z  
    },  
    previousDataValues: {  
      id: 1,  
      nome: 'REGISTRO 1',  
      idade: 1,  
      cidade: 'CIDADE 1',  
      createdAt: 2024-07-21T21:18:39.000Z,  
      updatedAt: 2024-07-21T21:18:39.000Z  
    }  
  },  
  ...  
]
```

```
usuario {  
  dataValues: {  
    id: 2,  
    nome: 'REGISTRO 2',  
    idade: 2,  
    cidade: 'CIDADE 2',  
    createdAt: 2024-07-21T21:21:36.000Z,  
    updatedAt: 2024-07-21T21:21:36.000Z  
  },  
  previousDataValues: {  
    id: 2,  
    nome: 'REGISTRO 2',  
    idade: 2,  
    cidade: 'CIDADE 2',  
    createdAt: 2024-07-21T21:21:36.000Z,  
    updatedAt: 2024-07-21T21:21:36.000Z  
  }  
},  
...
```

```
usuario {  
  dataValues: {  
    id: 3,  
    nome: 'REGISTRO 3',  
    idade: 3,  
    cidade: 'CIDADE 3',  
    createdAt: 2024-07-21T21:21:49.000Z,  
    updatedAt: 2024-07-21T21:21:49.000Z  
  },  
  previousDataValues: {  
    id: 3,  
    nome: 'REGISTRO 3',  
    idade: 3,  
    cidade: 'CIDADE 3',  
    createdAt: 2024-07-21T21:21:49.000Z,  
    updatedAt: 2024-07-21T21:21:49.000Z  
  }  
},  
...
```

# API

- Adicionar a busca por um único registro na index.js

```
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejrãp mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5     const database = require('./data_base/db');
6     const Usuario = require('./modelo/Usuario');
7
8     try {
9         const resultado = await database.sync();
10        console.log(resultado);
11    } catch (error) {
12        console.log(error);
13    }
14
15    //insere registro via Sequelize
16    const resultadoCreate = await Usuario.create({
17        nome: 'REGISTRO 3',
18        idade: 3,
19        cidade: 'CIDADE 3'
20    })
21    console.log(resultadoCreate);
22
23
24    //buscar os registros inseridos
25    const usuarios = await Usuario.findAll();
26    console.log(usuarios);
27
28
29    const usuario = await Usuario.findByPk(1);
30    console.log(usuario);
31
32 })();
33
```

# API

- ▶ Alterar um registro, adicionar na index.js o código abaixo

```
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejam mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5     const database = require('./data_base/db');
6     const Usuario = require('./modelo/Usuario');
7
8     try {
9         const resultado = await database.sync();
10        console.log(resultado);
11    } catch (error) {
12        console.log(error);
13    }
14
15    //insere registro via Sequelize
16    const resultadoCreate = await Usuario.create({
17        nome: 'REGISTRO 3',
18        idade: 3,
19        cidade: 'CIDADE 3'
20    });
21    console.log(resultadoCreate);
22
23
24    //buscar os registros inseridos
25    const usuarios = await Usuario.findAll();
26    console.log(usuarios);
27
28
29    const usuario = await Usuario.findByPk(1);
30    console.log(usuario);
31
32    //alterar registro
33    const alterarUsuario = await Usuario.findByPk(1);
34    alterarUsuario.nome = "REGISTRO ALTERADO";
35
36    const resultadoalterado = await alterarUsuario.save();
37    console.log(resultadoalterado);
38
39
40 })();
41
```

# API

- Deletar um registro, adicionar na index.js o código abaixo

```
1
2 // Sync do Sequelize com nosso banco de dados, as tabelas sejam mapeadas de forma correta
3 //IIFE (Immediately Invoked Function Expression) funcao em JavaScript que e executada assim que definida.
4 (async () => {
5     const database = require('./data_base/db');
6     const Usuario = require('./modelo/Usuario');
7
8     try {
9         const resultado = await database.sync();
10        console.log(resultado);
11    } catch (error) {
12        console.log(error);
13    }
14
15    //insere registro via Sequelize
16    const resultadoCreate = await Usuario.create({
17        nome: 'REGISTRO 3',
18        idade: 3,
19        cidade: 'CIDADE 3'
20    });
21    console.log(resultadoCreate);
22
23
24    //buscar os registros inseridos
25    const usuarios = await Usuario.findAll();
26    console.log(usuarios);
27
28
29    const usuario = await Usuario.findByPk(1);
30    console.log(usuario);
31
32    //alterar registro
33    const alterarUsuario = await Usuario.findByPk(1);
34    alterarUsuario.nome = "REGISTRO ALTERADO";
35
36    const resultadoSave = await alterarUsuario.save();
37    console.log(resultadoSave);
38
39    //deletar registro
40    const deletarRegistro = await Usuario.findByPk(2);
41    deletarRegistro.destroy();
42    console.log(deletarRegistro);
43
44 })();
```



# API

- ▶ Exemplo, requisição rota buscando dados pelo Sequelize.
- ▶ Criar nova index.js e escrever o código abaixo

```
1  // server.js
2  const express = require('express');
3  const db = require('./data_base/db');
4
5  const Usuario = require('./modelo/Usuario');
6
7  const app = express();
8  💡
9  const PORT = process.env.PORT;
10
11  db.sync().then(() => {
12    app.listen(PORT, () => {
13      console.log(`Servidor rodando na porta ${PORT}`);
14    });
15  });
16
17  // Exemplo de uso
18  app.get('/usuarios', async (req, res) => {
19    //buscar os registros inseridos
20    const usuarios = await Usuario.findAll();
21    console.log(usuarios);
22    res.json(usuarios);
23  });
```

- ▶ Executar a rota users

# API

GET ⌵ http://localhost:3000/usuarios Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

HTTP Headers ☐ Raw

☒ Accept

\*/\*

☒ User-Agent

Thunder Client (https://www.thunderclient.com)

☐ header

value

Status: 200 OK Size: 145 Bytes Time: 43 ms

Response Headers<sup>6</sup> Cookies Results Docs {} ≡

```
1  [  
2    {  
3      "id": 1,  
4      "nome": "REGISTRO ALTERADO",  
5      "idade": 1,  
6      "cidade": "CIDADE 1",  
7      "createdAt": "2024-07-22T21:45:27.000Z",  
8      "updatedAt": "2024-07-23T00:45:27.000Z"  
9    }  
10 ]
```