

# PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

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

16/07/2024

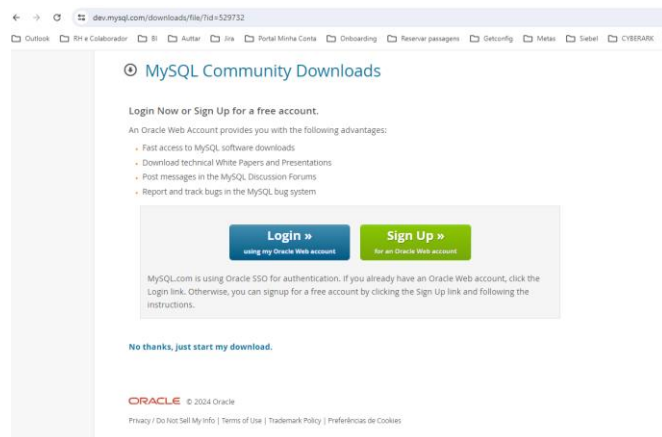
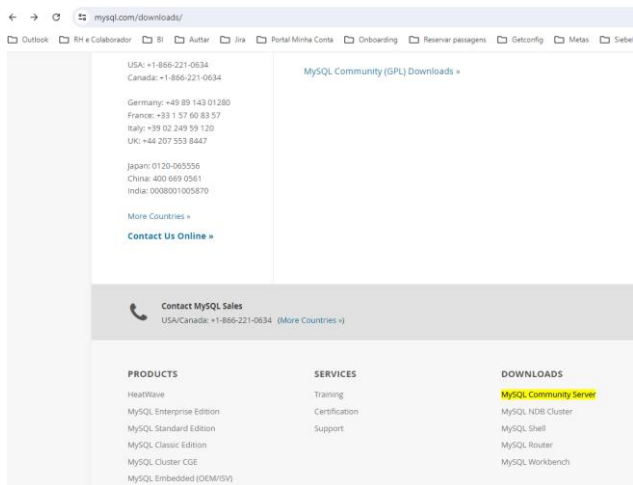
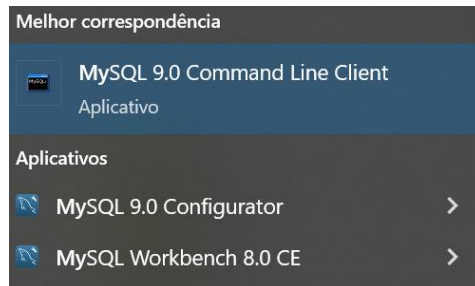
# Agenda

- ▶ Criar banco de dados relacional
- ▶ Criar API
- ▶ Criar funcionalidades
  - ▶ Cadastro de clientes
  - ▶ Listagem de clientes
  - ▶ Pesquisa de clientes
  - ▶ Edição de clientes
  - ▶ Exclusão de clientes
- ▶ ORM (Object-Relational Mapping)

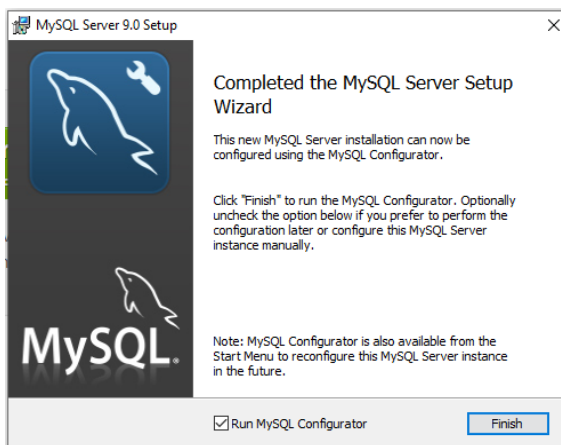
# Instalar Mysql

- Verificar se tem Mysql instalado

- Download Mysql -> <https://dev.mysql.com/downloads/mysql/>

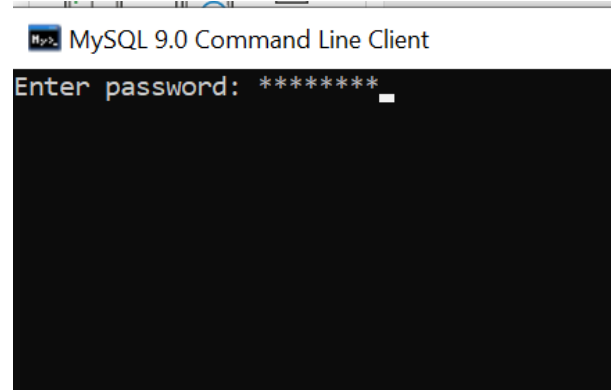


- Se não tiver Instalar

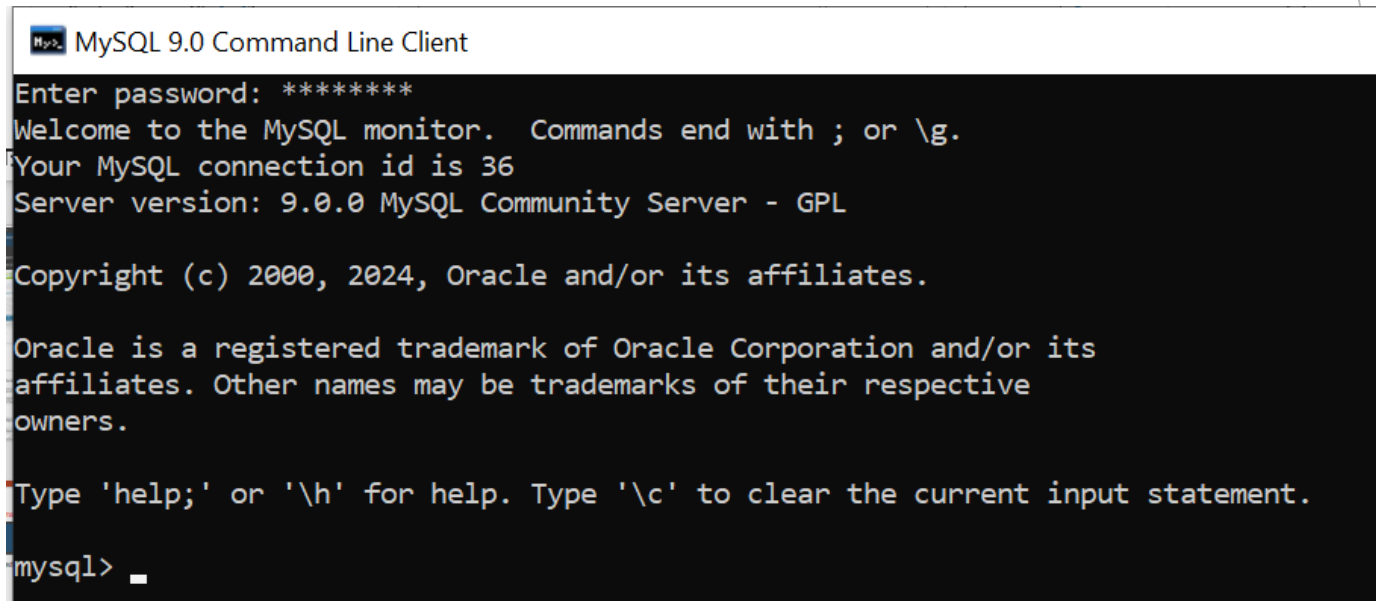


# Mysql Command Line

- ▶ Procurar no menu Windows Por Mysql Command Line Client
- ▶ Executar o Mysql Command Line. Será solicitado o password configurado na instalação do Mysql em sua máquina.



- ▶ Se a senha estiver correta, você vai ter essa visão.



# Mysql Command Line

- ▶ Executar alguns comandos no Command Line.
- ▶ Ver os databases já criados: show databases;
- ▶ Criar banco de dados:
  - ▶ CREATE DATABASE <nome do database>;
  - ▶ Criar database com nome API
- ▶ Selecionar o database criado:
  - ▶ use API;
- ▶ Criar tabela com o nome USUARIOS
  - ▶ CREATE TABLE IF NOT EXISTS USUARIOS(
    - ▶ ID INT NOT NULL AUTO\_INCREMENT,
    - ▶ NOME VARCHAR(255) NOT NULL,
    - ▶ IDADE INT NOT NULL,
    - ▶ CIDADE VARCHAR(150) NOT NULL,
    - ▶ PRIMARY KEY (ID)
  - ▶ );
- ▶ Verificar as tabelas criadas no database
- ▶ Verificar as colunas criada em uma tabela:
  - ▶ show columns from <nome da tabela>;

```
mysql> CREATE DATABASE API;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> use API
Database changed
```

```
mysql> CREATE TABLE IF NOT EXISTS USUARIOS(
-> ID INT NOT NULL AUTO_INCREMENT,
-> NOME VARCHAR(255) NOT NULL,
-> IDADE INT NOT NULL,
-> CIDADE VARCHAR(150) NOT NULL,
-> PRIMARY KEY (ID)
-> );
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> show tables;
+-----+
| Tables_in_api_teste |
+-----+
| usuarios             |
+-----+
1 row in set (0.00 sec)
```

```
mysql> show columns from usuarios;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| ID    | int           | NO   | PRI | NULL    | auto_increment |
| NOME  | varchar(255)  | NO   |     | NULL    |                |
| IDADE | int           | NO   |     | NULL    |                |
| CIDADE | varchar(150)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

# Mysql Command Line

## ► Inserir registros na tabela

- INSERT INTO USUARIOS (NOME,IDADE,CIDADE) VALUES ("DAVI",43,"CANOAS");

```
mysql> INSERT INTO USUARIOS (NOME,IDADE,CIDADE) VALUES ("DAVI",43,"CANOAS");  
Query OK, 1 row affected (0.01 sec)
```

## ► Buscar dados da tabela

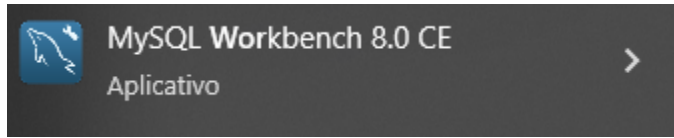
- SELECT \* FROM USUARIOS;

```
mysql> SELECT * FROM USUARIOS;  
+-----+-----+-----+-----+  
| ID | NOME | IDADE | CIDADE |  
+-----+-----+-----+-----+  
| 1 | DAVI | 43 | CANOAS |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

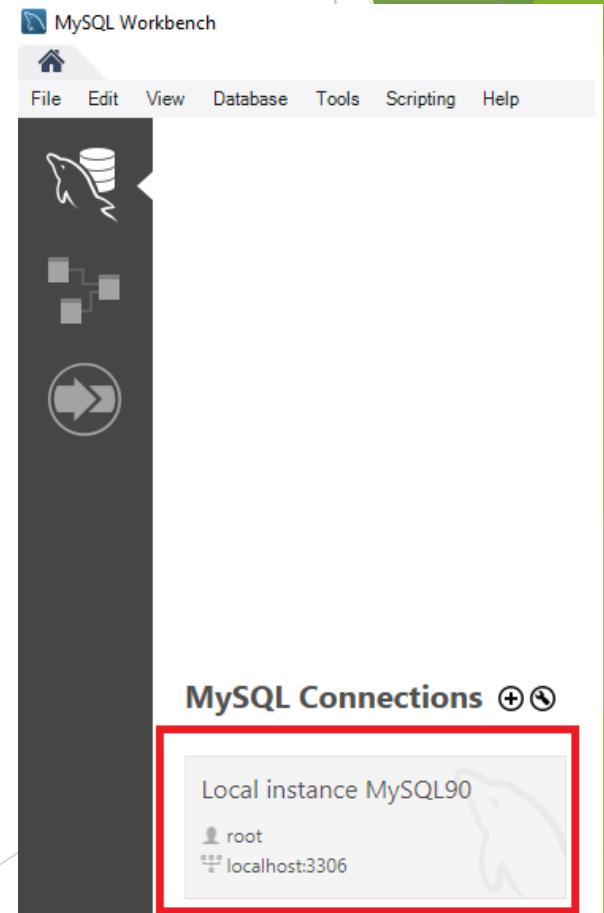
- Ver arquivo com comando no Github em [\PraticasAvancadasDesenvolvimentoWeb\Material das Aulas \Comandos MYSQL Via Command Line.txt](#)

# Mysql Workbench

- ▶ No menu Windows procurar por

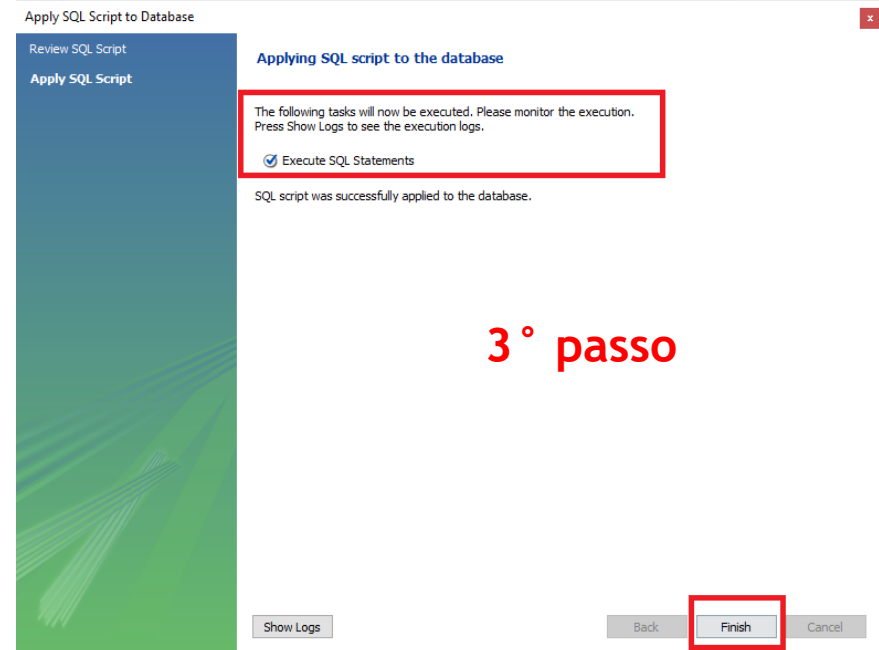
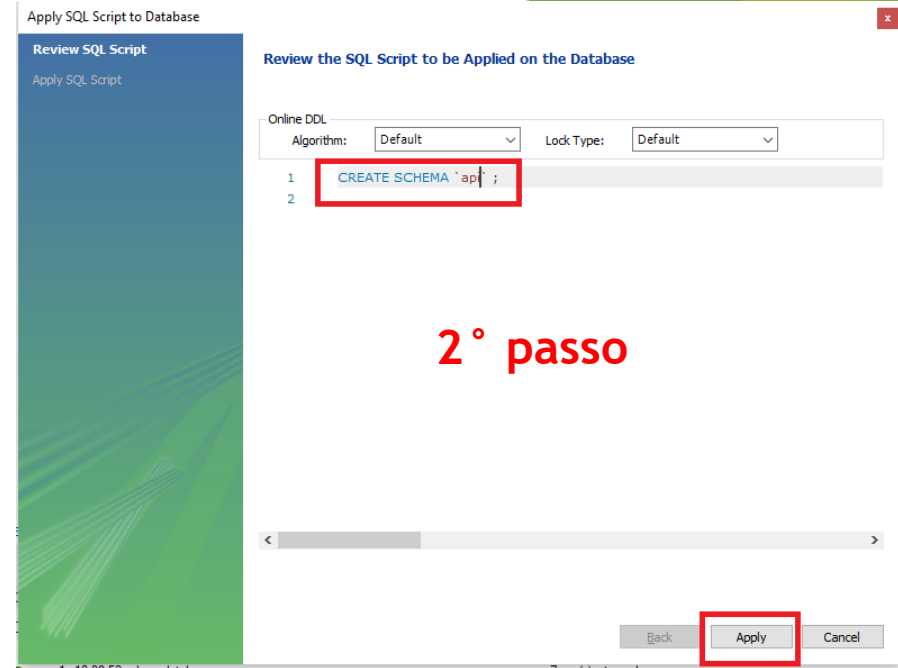
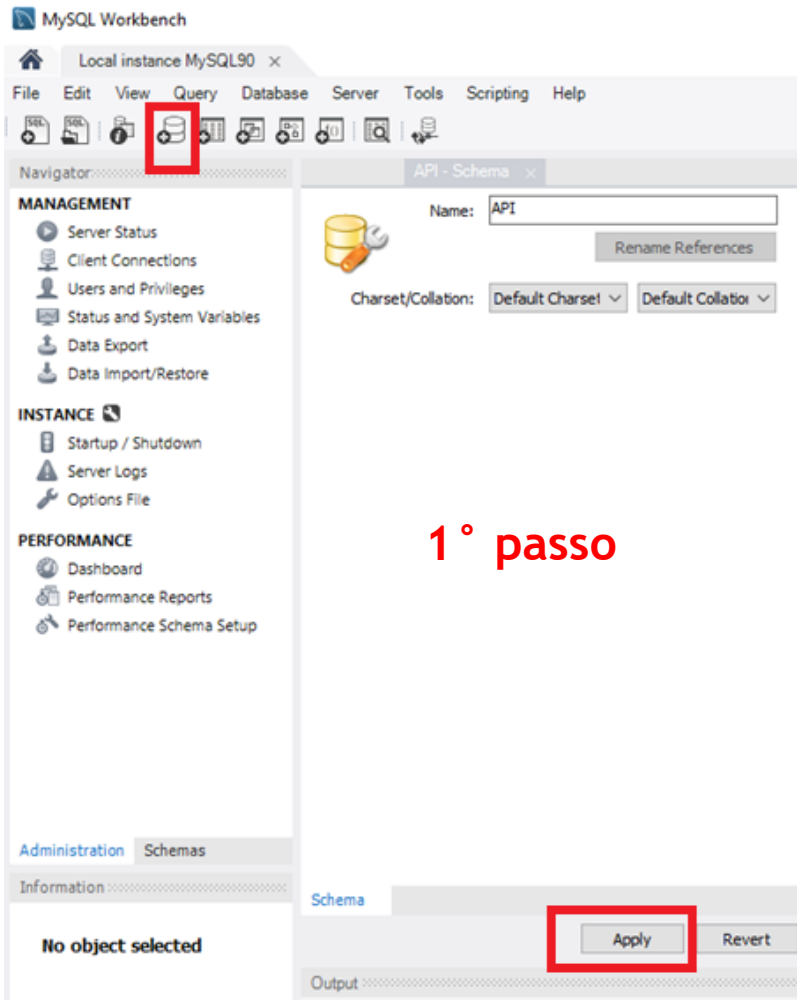


- ▶ Abrir o Mysql Workbench, clicar em Local instance



# Mysql Workbench

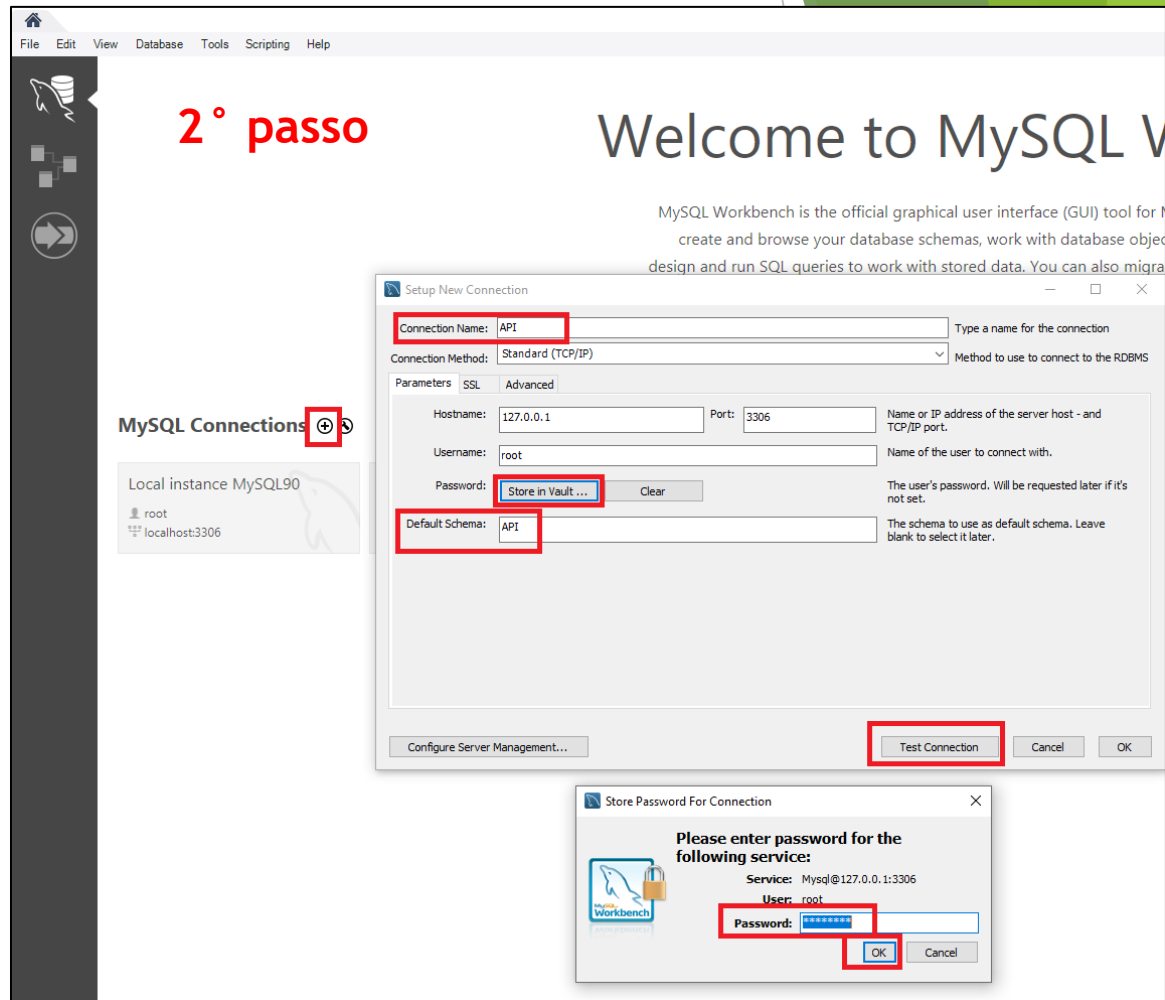
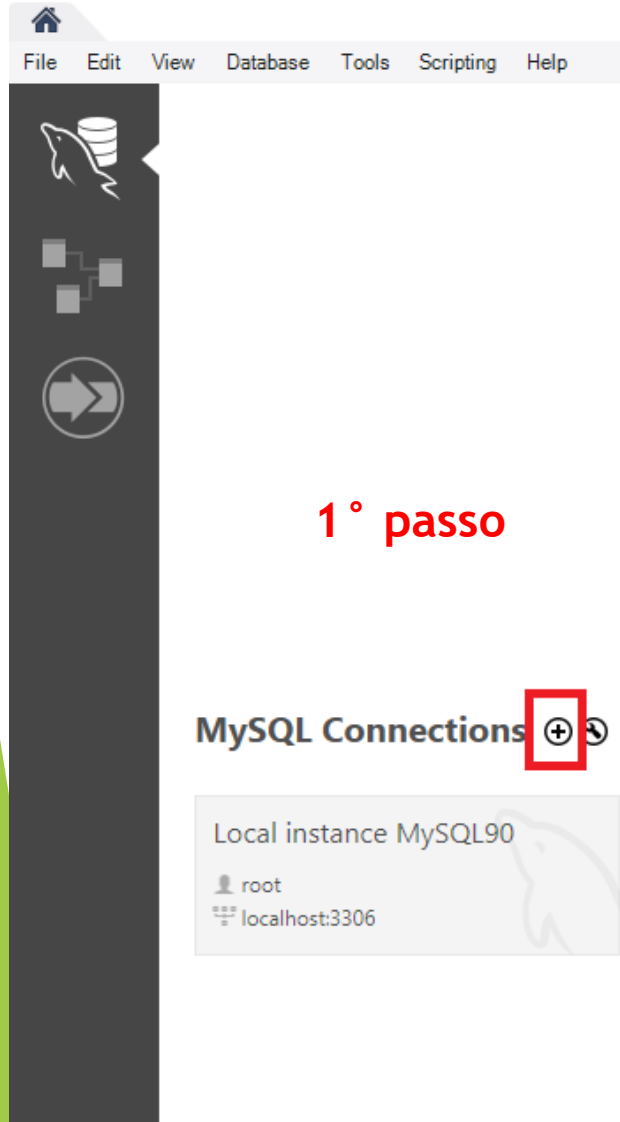
## ► Criar database pelo Workbench





# Mysql Workbench

## ► Conectar-se ao database



# Mysql Workbench

## ► Testar a conexão

3º passo

MySQL Connections ⊕ ⓘ

Local instance MySQL90

root  
localhost:3306

## Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to create and browse your database schemas, work with database objects, design and run SQL queries to work with stored data. You can also migrate data between MySQL instances.

Setup New Connection

Connection Name: API Type a name for the connection

Connection Method: Standard (TCP/IP)

Parameters SSL Advanced

Hostname: 127.0.0.1 Port: 3306

Username: root

Password:

Default Schema: API

Configure Server Management...

MySQL Workbench

**i** Successfully made the MySQL connection

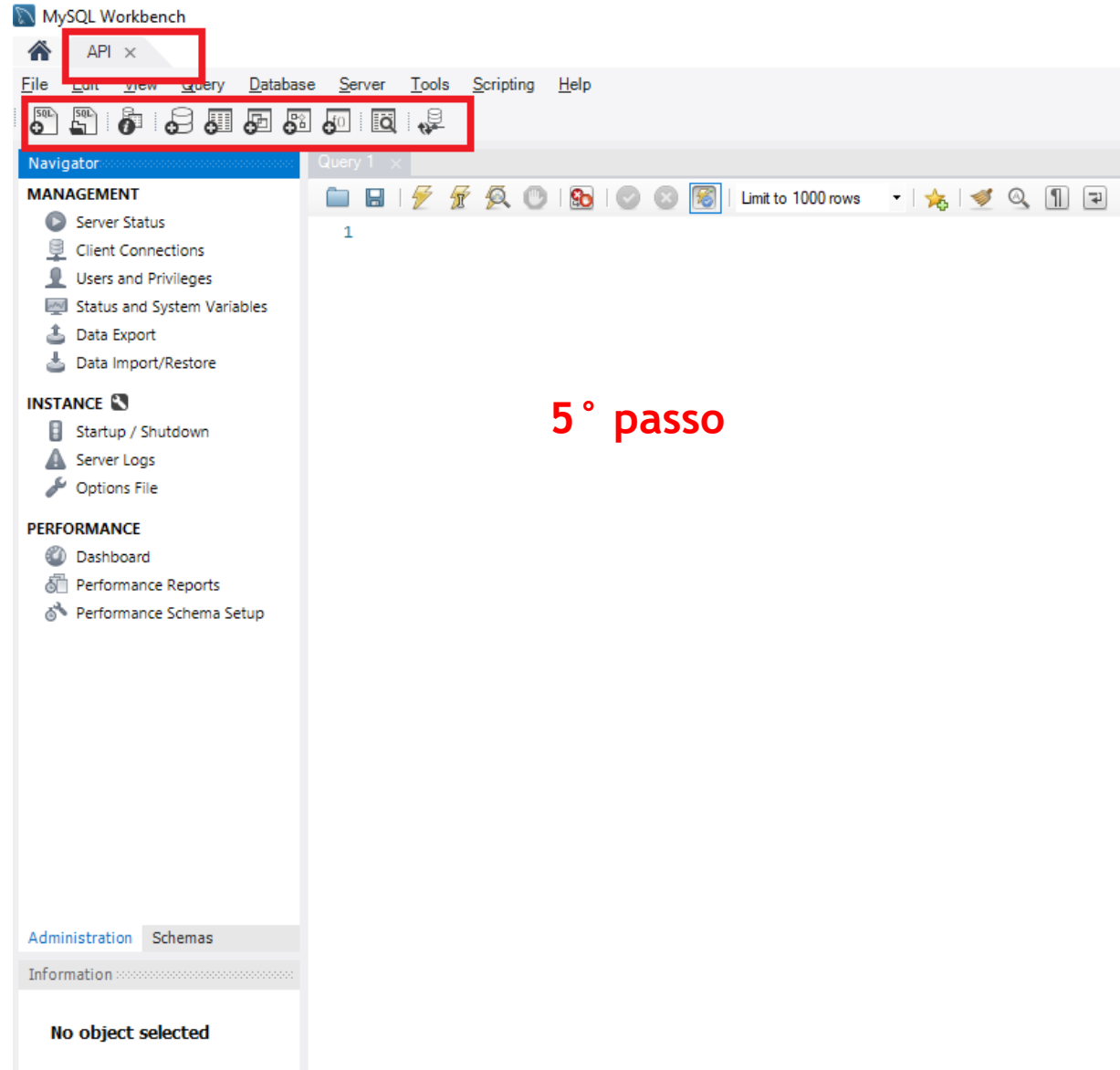
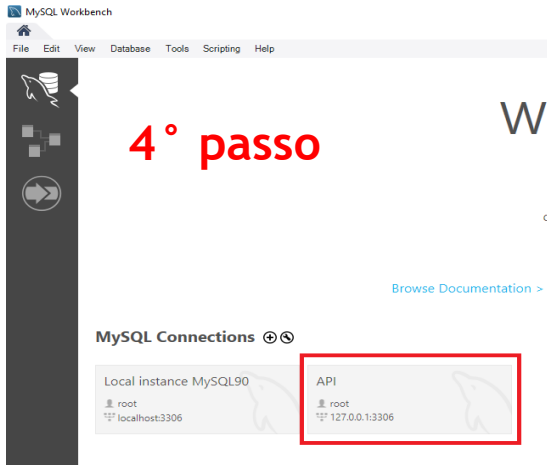
Information related to this connection:

Host: 127.0.0.1  
Port: 3306  
User: root  
SSL: enabled with TLS\_AES\_128\_GCM\_SHA256

A successful MySQL connection was made with the parameters defined for this connection.

# Mysql Workbench

## ► Utilizar conexão criada



# Mysql Workbench

## ► Criar tabela

The screenshot shows the MySQL Workbench interface. The 'Query' menu is highlighted with a red box. The 'Table Name' is 'usuarios' and the 'Schema' is 'API'. The 'Column Name' table is highlighted with a red box.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
NOME	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
IDADE	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
CIDADE	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

The 'Review SQL Script' window is open, showing the SQL script for creating the 'usuarios' table. The script is highlighted with a red box.

```
1 CREATE TABLE `API`.`usuarios` (  
2   `ID` INT NOT NULL AUTO_INCREMENT,  
3   `NOME` VARCHAR(255) NOT NULL,  
4   `IDADE` INT NOT NULL,  
5   `CIDADE` VARCHAR(150) NOT NULL,  
6   PRIMARY KEY (`ID`));  
7
```

The 'Apply' button in the 'Review SQL Script' window is highlighted with a red box. The 'Apply' button in the bottom right corner is also highlighted with a red box.

# Mysql Workbench

## ► Visualizar tabelas

The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'SCHEMAS' tree with 'api' expanded, revealing the 'usuarios' table and its columns (ID, NOME, IDADE, CIDADE). The 'Schemas' tab is selected in the bottom-left pane. The main area shows the 'usuarios' table structure for the 'API' schema. The table has four columns: ID (INT, PK, NN), NOME (VARCHAR(255), NN), IDADE (INT, NN), and CIDADE (VARCHAR(150), NN). The 'Columns' tab is active, showing the column details for 'ID'.

**Table Structure:**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
NOME	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
IDADE	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
CIDADE	VARCHAR(150)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

**Column Details (ID):**

Column Name: ID  
CharSet/Collation: Default CharSet / Default Collation  
Data Type: INT  
Default:   
Storage: ☒ Primary Key ☒ Not Null ☐ Unique ☐ Binary ☐ Unsigned ☐ Zero Fill ☒ Auto Increment ☐ Generated

Buttons: Apply, Revert

# Mysql Workbench

## ► Visualizar registros da tabela

MySQL Workbench

API x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

api

Tables

usuarios

Stored Procedures

Functions

api2

sakila

sys

world

Limit to 1000 rows

1 • `SELECT * FROM api.usuarios;`

Result Grid

ID	NOME	IDADE	CIDADE
NULL	NULL	NULL	NULL

Edit: | Export/Import: | Wrap Cell Content: |

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

# Mysql Workbench

## ► inserir registros da tabela

The screenshot illustrates the steps to insert a record into the 'usuarios' table in MySQL Workbench. The interface shows the 'api' database selected in the Navigator, with the 'usuarios' table highlighted. The Query Editor displays the SQL statement: `SELECT * FROM api.usuarios;`. The Result Grid shows the current data in the table, with a new row being added: `NULL`, `DAVI`, `43`, `CANOAS`. The 'Apply SQL Script to Database' dialog box is open, showing the SQL script: `INSERT INTO `api`.`usuarios` (`NOME`, `IDADE`, `CIDADE`) VALUES ('DAVI', '43`. The 'Apply' button is highlighted in the dialog box. The bottom status bar shows the message: 'Changes applied'.

1° passo

2° passo

3° passo

4° passo

# Criar API

- ▶ Acessar o repositório onde são salvas as aplicações
- ▶ Criar uma pasta da aplicação com o nome APIMYSQL
- ▶ Executar o comando `npm init -y`
- ▶ Executar o comando `npm install mysql2`
  - ▶ `mysql2`: conectar e mandar comandos SQL para o banco
- ▶ Executar o comando `dotenv`
  - ▶ `dotenv`: gestão das configurações do projeto
- ▶ Executar o comando `express`
  - ▶ `express`: web framework para construção da infraestrutura da API;

```
Eric@DESKTOP-8CI3MGV MINGW64 /c/Users/SenacRs/PraticasAvancadasDesenvolvimentoWeb/APIMYSQL (main)
$ npm init -y
Wrote to C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\APIMYSQL\package.json:

{
  "name": "apimysql",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

Eric@DESKTOP-8CI3MGV MINGW64 /c/Users/SenacRs/PraticasAvancadasDesenvolvimentoWeb/APIMYSQL (main)
$ ls -ltr
total 1
drwxr-xr-x 1 Eric 197609  0 Jul 15 15:24 Backup_BD/
-rw-r--r-- 1 Eric 197609 222 Jul 15 15:31 package.json

Eric@DESKTOP-8CI3MGV MINGW64 /c/Users/SenacRs/PraticasAvancadasDesenvolvimentoWeb/APIMYSQL (main)
$ npm install mysql2 dotenv express

added 76 packages, and audited 77 packages in 7s

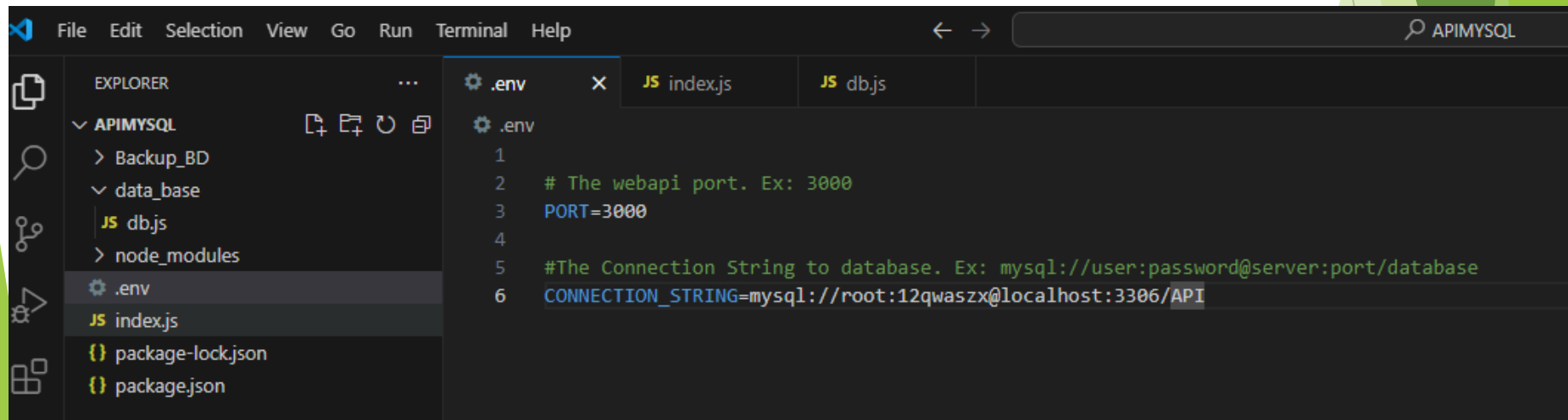
13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



# Arquivo configuração

- ▶ Abrir o Visual Studio Code
- ▶ Dentro do VSC abrir a pasta APIMYSQL
- ▶ Criar arquivo sem nome somente com a extensão .env na raiz do projeto
- ▶ Adicionar os parâmetros PORT e CONNECTION\_STRING no arquivo criado
  - ▶ PORT: valor da porta que o servidor vai estar rodando, exemplo: 3000
  - ▶ CONNECTION\_STRING: conexão utilizada para acessar o MYSQL
    - ▶ String to database. Ex: `mysql://user:password@server:port/database`

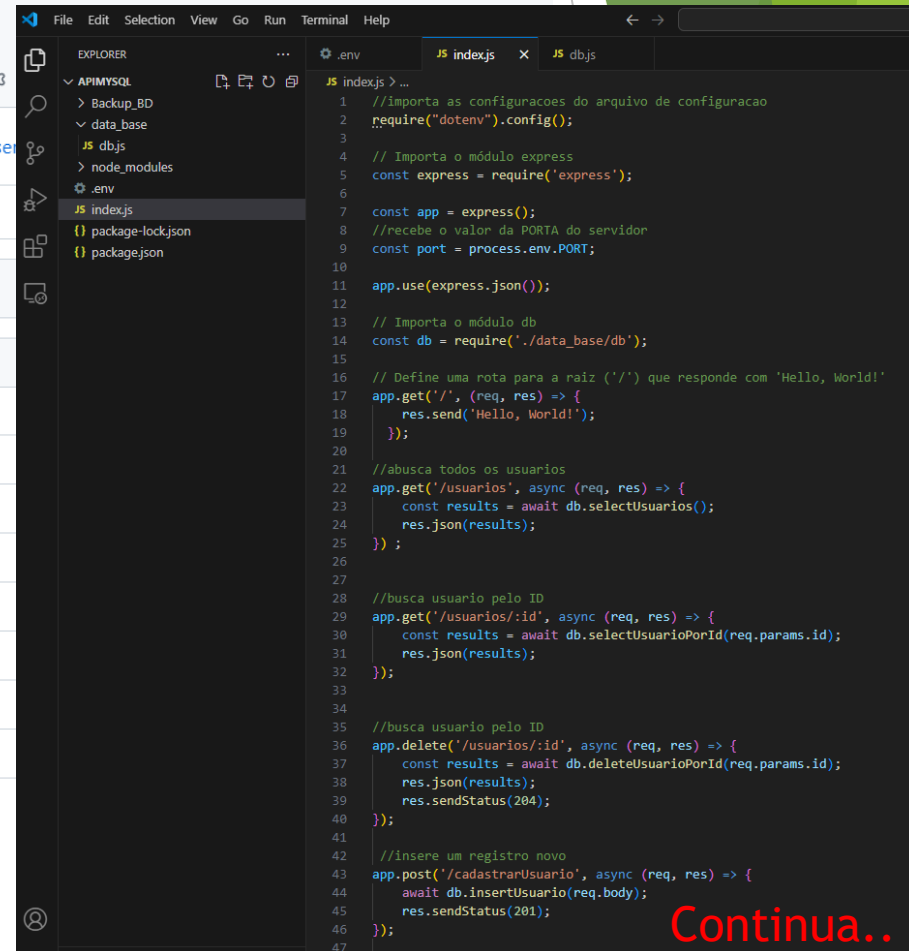
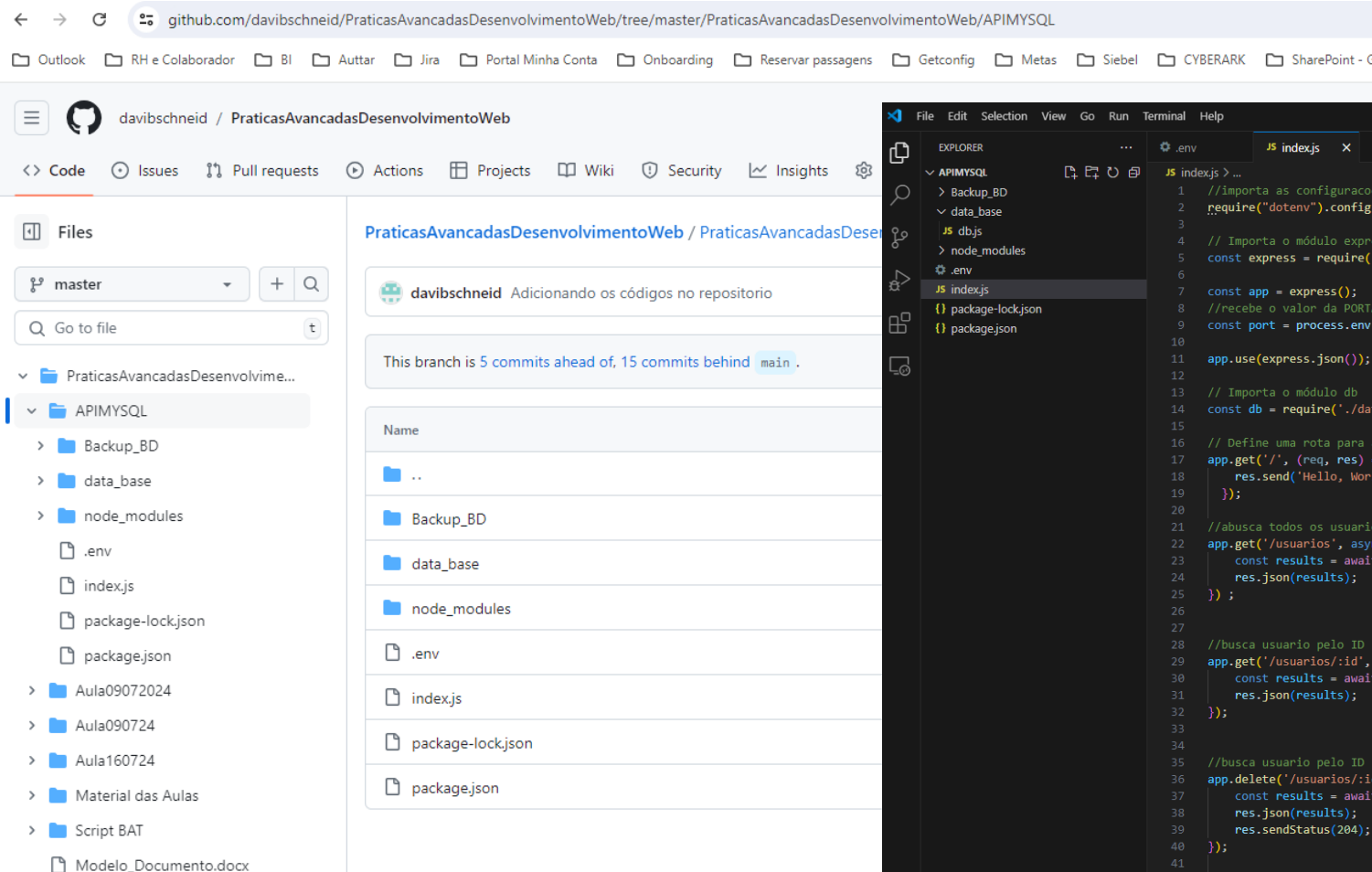


The screenshot shows the Visual Studio Code interface with the Explorer view on the left displaying the project structure. The file explorer shows a folder named 'APIMYSQL' containing subfolders 'Backup\_BD' and 'data\_base', and files 'db.js', 'node\_modules', '.env', 'index.js', 'package-lock.json', and 'package.json'. The '.env' file is selected and its contents are displayed in the editor. The editor shows the following configuration:

```
.env
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 index.js

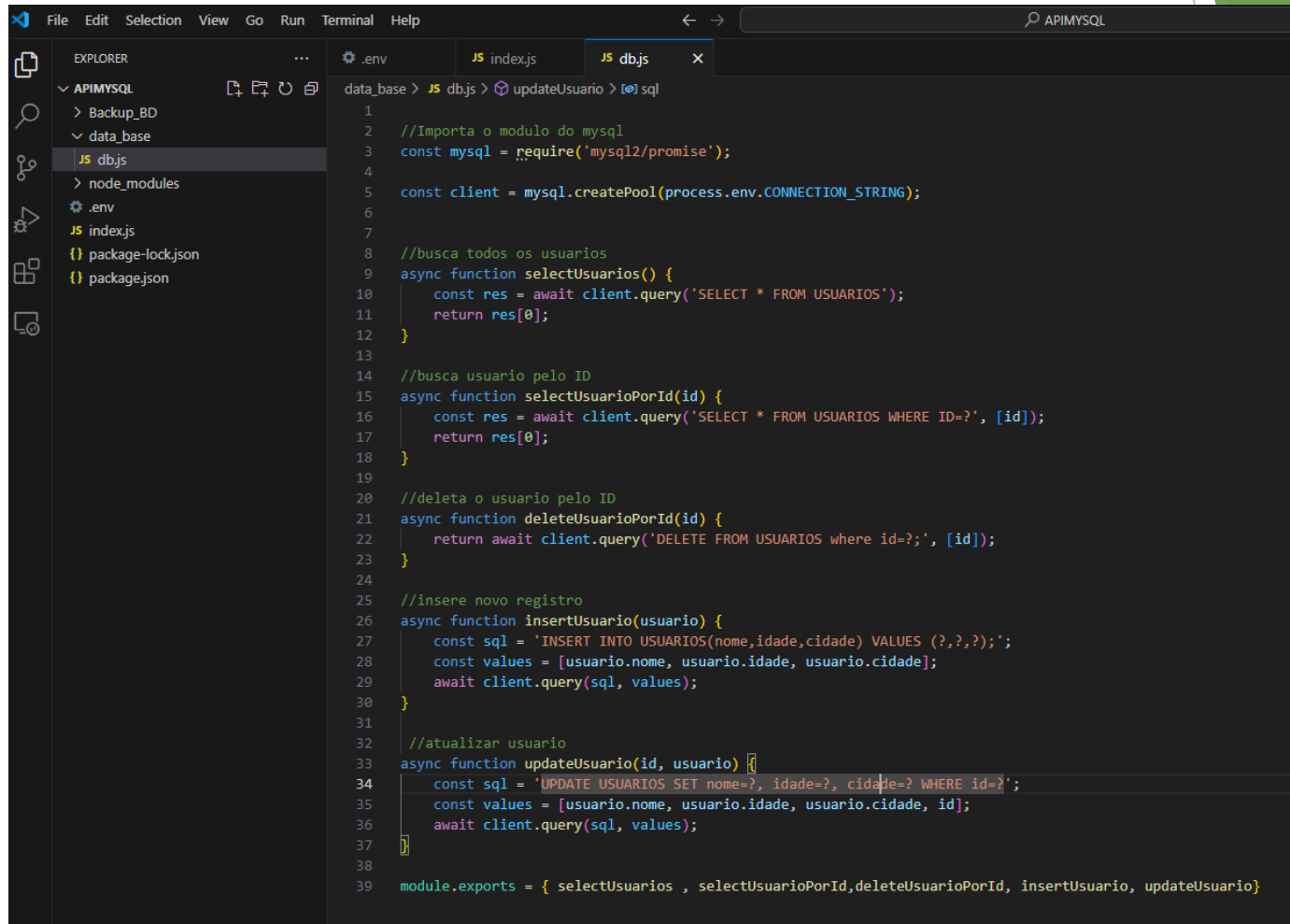
- ▶ Criar arquivo index.js na raiz do projeto
- ▶ Código disponibilizado no github



Continua..

# Criar db.js

- Criar arquivo db.js na raiz da pasta APIMYSQL

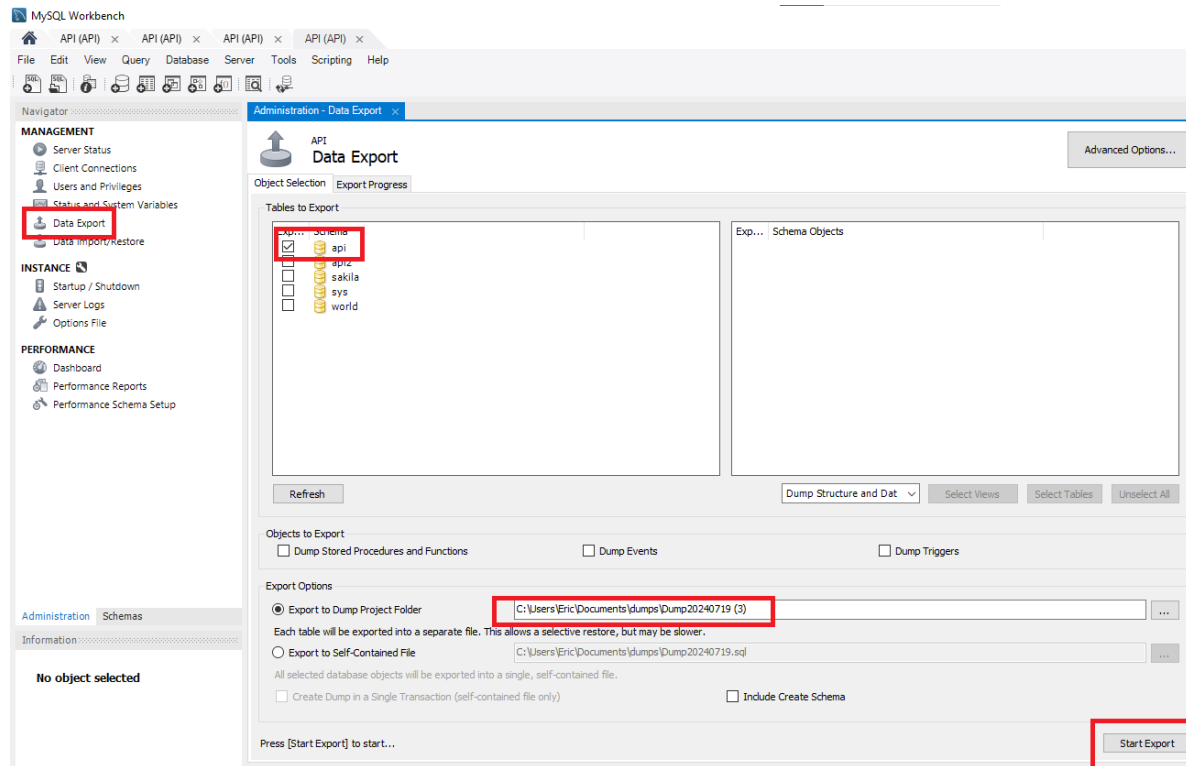


The screenshot shows the Visual Studio Code interface with the Explorer panel on the left and the Editor panel on the right. The Explorer panel shows the project structure for 'APIMYSQL', including folders like 'Backup\_BD', 'data\_base', and 'node\_modules', and files like '.env', 'index.js', 'package-lock.json', and 'package.json'. The 'db.js' file is highlighted in the Explorer panel. The Editor panel shows the content of 'db.js', which is a JavaScript file using the MySQL promise library to interact with a database. The code includes functions for selecting all users, selecting a user by ID, deleting a user by ID, inserting a new user, and updating a user. The file is named 'db.js' and is located in the 'data\_base' folder.

```
1
2 //Importa o modulo do mysql
3 const mysql = require('mysql2/promise');
4
5 const client = mysql.createPool(process.env.CONNECTION_STRING);
6
7
8 //busca todos os usuarios
9 async function selectUsuarios() {
10   const res = await client.query('SELECT * FROM USUARIOS');
11   return res[0];
12 }
13
14 //busca usuario pelo ID
15 async function selectUsuarioPorId(id) {
16   const res = await client.query('SELECT * FROM USUARIOS WHERE ID=?', [id]);
17   return res[0];
18 }
19
20 //deleta o usuario pelo ID
21 async function deleteUsuarioPorId(id) {
22   return await client.query('DELETE FROM USUARIOS where id=?', [id]);
23 }
24
25 //insere novo registro
26 async function insertUsuario(usuario) {
27   const sql = 'INSERT INTO USUARIOS(nome,idade,cidade) VALUES (?,?,:)';
28   const values = [usuario.nome, usuario.idade, usuario.cidade];
29   await client.query(sql, values);
30 }
31
32 //atualizar usuario
33 async function updateUsuario(id, usuario) {
34   const sql = 'UPDATE USUARIOS SET nome=?, idade=?, cidade=? WHERE id=?';
35   const values = [usuario.nome, usuario.idade, usuario.cidade, id];
36   await client.query(sql, values);
37 }
38
39 module.exports = { selectUsuarios , selectUsuarioPorId,deleteUsuarioPorId, insertUsuario, updateUsuario}
```

# Criar pasta backup DB

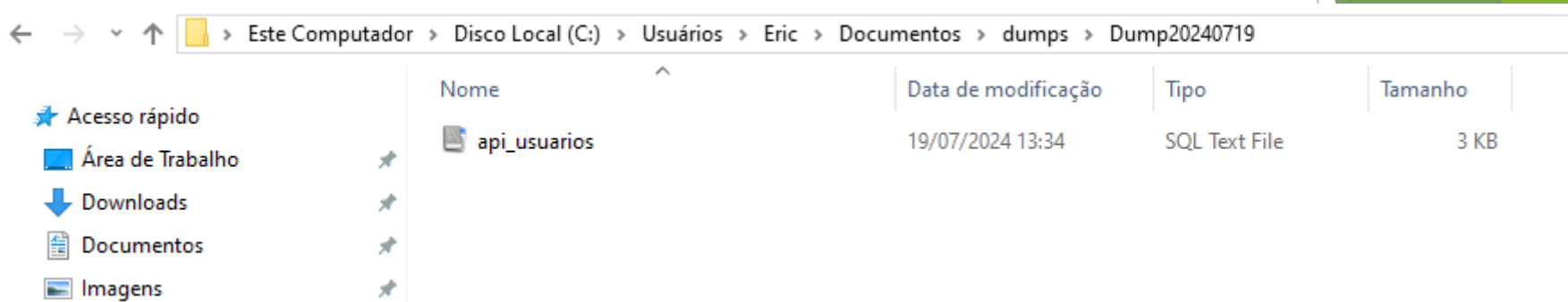
- ▶ Na raiz da pasta APIMYSQL, criar a pasta Backup\_BD
  - ▶ Dentro dela, criar o arquivo DDL.sql
  - ▶ No arquivo DDL.sql inserir os comandos executados na criação do BD.
  - ▶ Para isso, pode exportar os comandos do MysqlWorkbench



- ▶ Ou ir salvando os comandos executados via MYSQL Shell.

# Backup DB

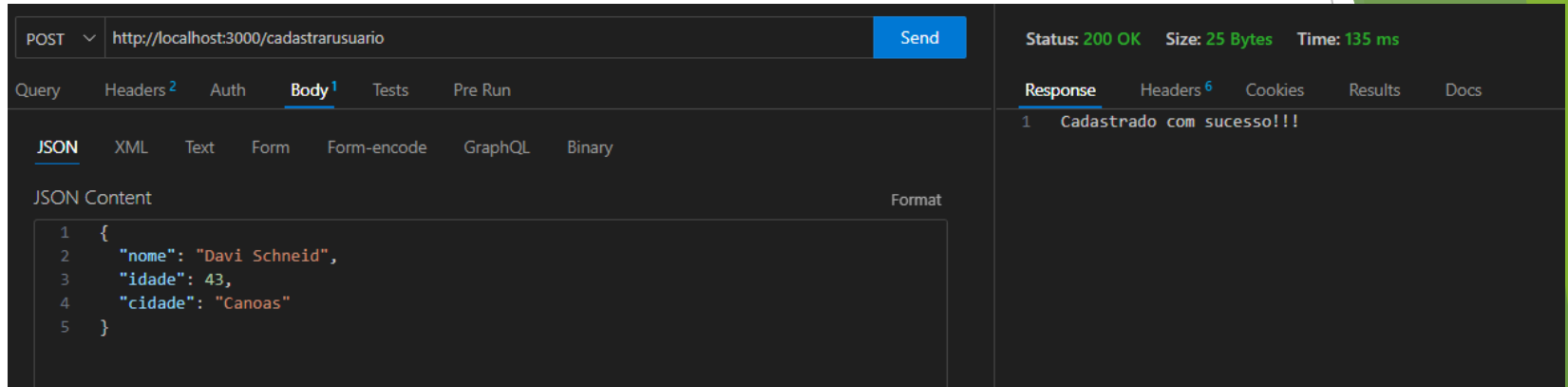
## ► Acessar a pasta DUMP



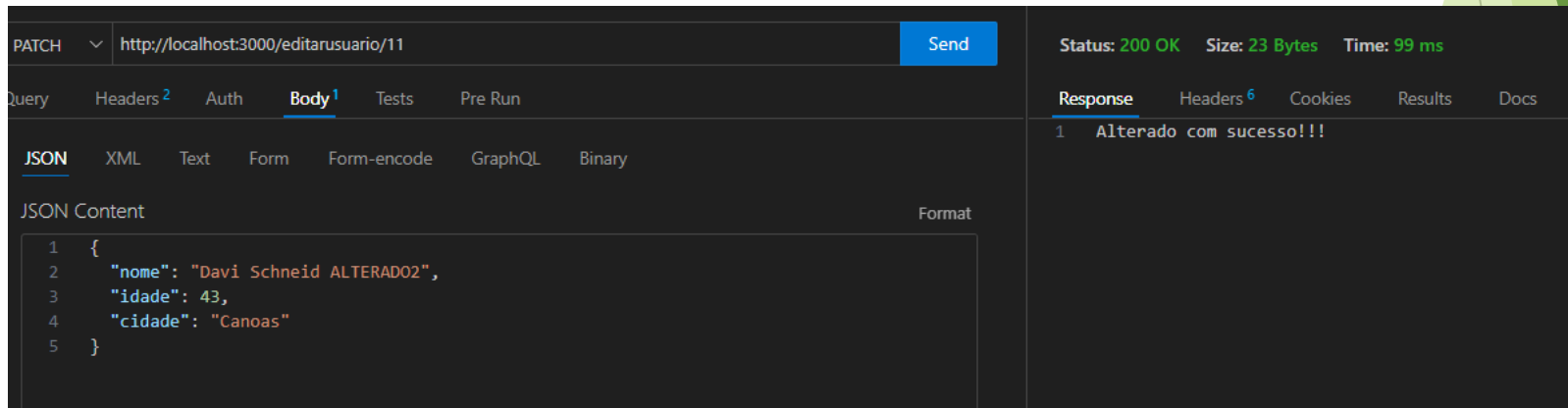
- Abrir o arquivos `api_usuarios.sql` e verificar o export da base.
- Criar o arquivo `DML.sql` dentro da pasta `Backup_BD`
- Salvar os dados expostados no arquivo `DML.sql`

# Postman ou Thunder Client

- ▶ Executa a rota cadastrarusuario

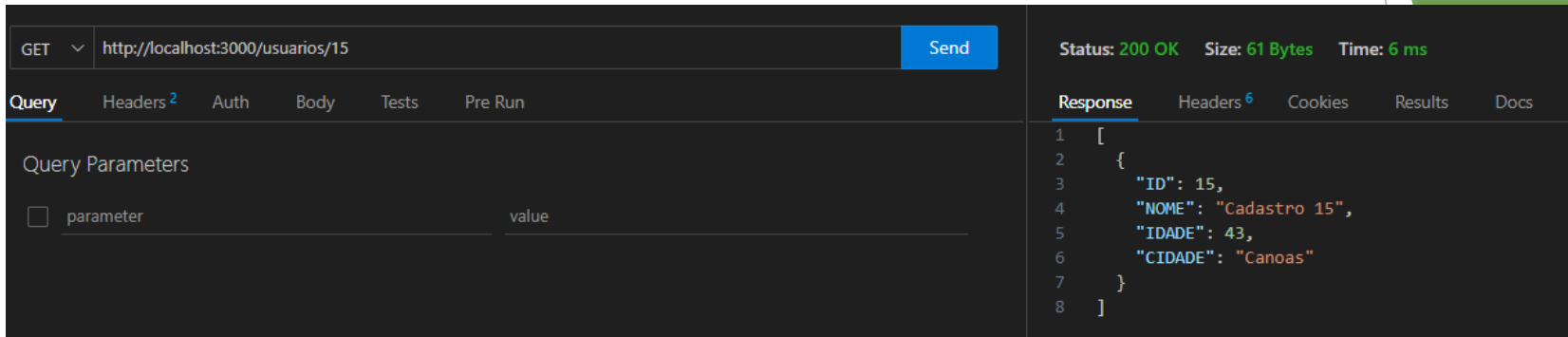


- ▶ Executa a rota editarusuario

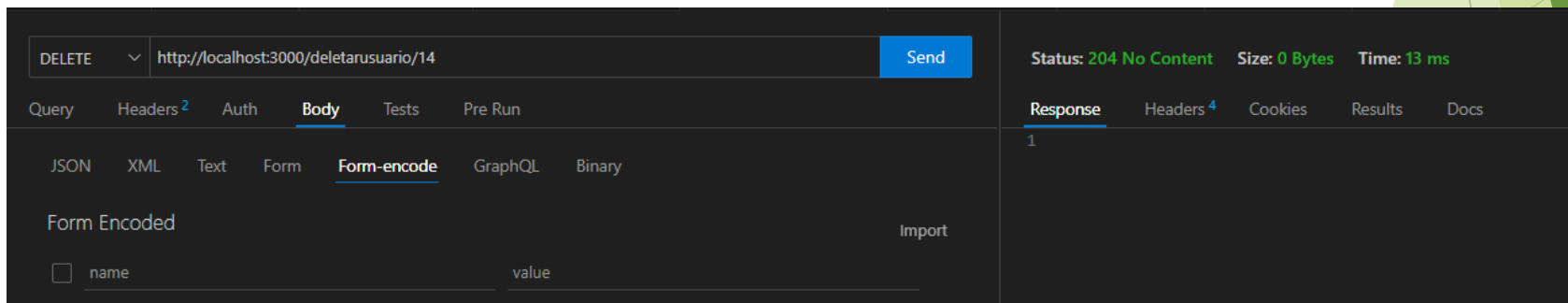


# Postman ou Thunder Client

- ▶ Executa a rota buscar usuário pelo ID



- ▶ Executa a rota deletar por id



# Postman ou Thunder Client

- ▶ Executar a rota buscar todos os usuarios

The screenshot displays the Postman interface for a REST client. The top bar shows the method 'GET' and the URL 'http://localhost:3000/usuarios'. The 'Send' button is visible. Below the URL bar, the 'Body' tab is selected, showing 'JSON' content. The right panel displays the response status '200 OK', size '351 Bytes', and time '4 ms'. The response body is a JSON array of three user objects.

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

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

**JSON** XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Status: 200 OK Size: 351 Bytes Time: 4 ms

**Response** Headers<sup>6</sup> Cookies Results Docs

```
1 [
2   {
3     "ID": 11,
4     "NOME": "Davi Schneid ALTERAD00000000xxxxxxxxxxxx",
5     "IDADE": 43,
6     "CIDADE": "Canoas"
7   },
8   {
9     "ID": 12,
10    "NOME": "Davi Schneid22222",
11    "IDADE": 43,
12    "CIDADE": "Canoas"
13  },
14  {
15    "ID": 13,
16    "NOME": "Davi Schneid22222",
17    "IDADE": 43,
18    "CIDADE": "Canoas"
19  },
20 ]
```