

Curso Node.JS

Fernando Silva Maransatto

https://www.youtube.com/watch?v=d_vXgK4uZJM&list=PLWgD0gfm500EMEDPyb3Orb28i7HK5_DkR
<https://github.com/Maransatto/rest-api-node-js>

Aula 01 - Criando o ambiente

- Instale o Node (última versão LTS) e o Visual Studio Code.
- Em C:\ crie a pasta **maransatto** e dentro dela a subpasta **rest**.

```
C:\maransatto\rest>node --version  
v14.15.3  
C:\maransatto\rest>npm --version  
6.14.9
```

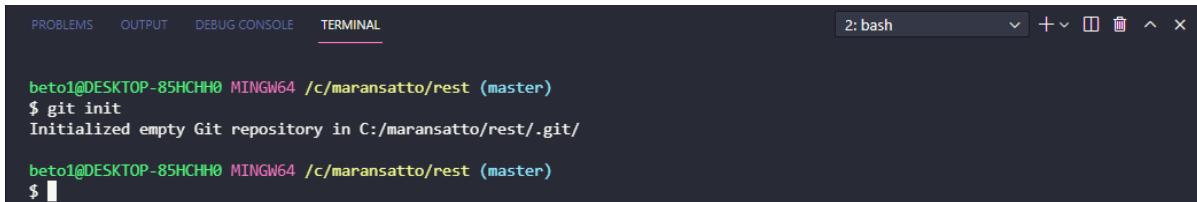
No terminal, entre com:

```
code .
```

```
C:\maransatto\rest>code .
```

- No terminal do Visual Studio Code, crie um repositório git:

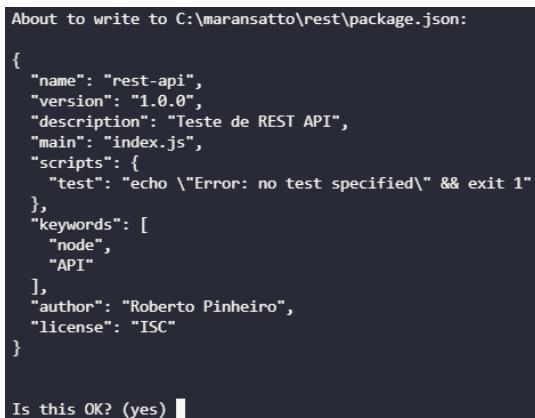
```
git init
```



```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)  
$ git init  
Initialized empty Git repository in C:/maransatto/rest/.git/  
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)  
$
```

- No terminal, entre com o comando:

```
npm init
```



```
About to write to C:\maransatto\rest\package.json:  
{  
  "name": "rest-api",  
  "version": "1.0.0",  
  "description": "Teste de REST API",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [  
    "node",  
    "API"  
  ],  
  "author": "Roberto Pinheiro",  
  "license": "ISC"  
}  
  
Is this OK? (yes)
```

Ao confirmar, digitando **yes**, é criado um arquivo chamado **package.json** (arquivo de configuração do Node).

package.json

```
{  
  "name": "rest-api",  
  "version": "1.0.0",  
  "description": "Teste de REST API",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [  
    "node",  
    "API"  
  ],  
  "author": "Roberto Pinheiro",  
  "license": "ISC"  
}
```

Instalando o Express

- O Express é uma biblioteca da npm que é utilizada para requisições HTTP do Node (base das requisições HTTP).

- No terminal, entre com o comando:

```
npm install --save express
```

--save incluirá o express no arquivo **package.json**

```
C:\maransatto\rest>npm install --save express  
npm notice created a lockfile as package-lock.json. You should commit this file.  
npm WARN rest-api@1.0.0 No repository field.  
  
+ express@4.17.1  
added 50 packages from 37 contributors and audited 50 packages in 18.706s  
found 0 vulnerabilities
```

package.json

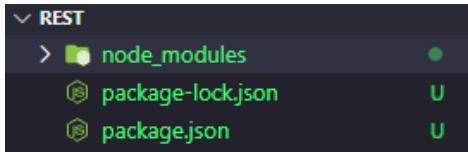
```
{  
  "name": "rest-api",  
  "version": "1.0.0",  
  "description": "Teste de REST API",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [  
    "node",  
    "express"  
  ]  
}
```

```

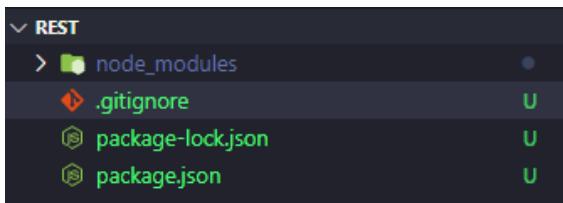
    "API"
],
"author": "Roberto Pinheiro",
"license": "ISC",
"dependencies": {
  "express": "^4.17.1"
}
}

```

- Na instalação do primeiro pacote de um projeto é criado a pasta `node_modules` que conterá todas as dependências do projeto.



- Essa pasta não deve ser subida para o repositório git.
- Para isso crie um arquivo chamado `.gitignore`, na pasta raiz do projeto, com o seguinte conteúdo:



`.gitignore`

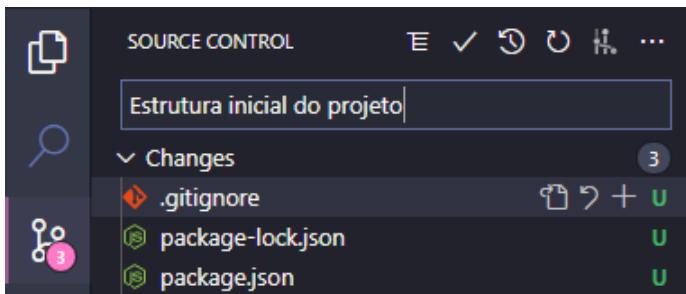
`node_modules`

- Não há necessidade de subir essa pasta, já que recuperá-la basta entrar com o comando:

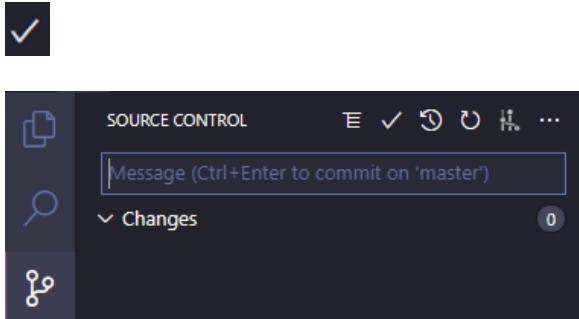
`npm install`

Isso fará com que todas as dependências especificadas no arquivo `package.json` sejam re-instaladas.

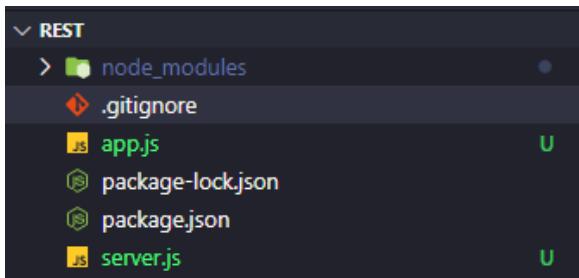
Realizando o primeiro commit



- Clique no botão Commit



- Na pasta raiz do projeto, crie um arquivo chamado **server.js** e um arquivo chamado **app.js**



app.js

```
const express = require('express');
const app = express();

app.use((req, res, next) => {
  res.status(200).send({
    mensagem: 'Primeiro teste realizado com sucesso!'
  });
});

module.exports = app;
```

server.js

```
const http = require('http');
const app = require('./app');
const server = http.createServer(app);

const port = process.env.PORT || 3000;
server.listen(port);

console.log('Servidor rodando na porta ' + port);
```

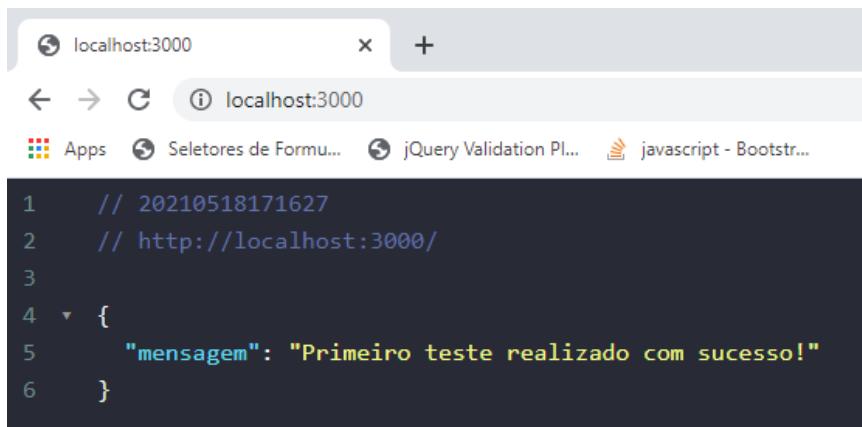
- Rode o servidor:

```
npm server.js
```

```
C:\maransatto\rest>node server.js
Servidor rodando na porta 3000
```

- No browser:

localhost:3000



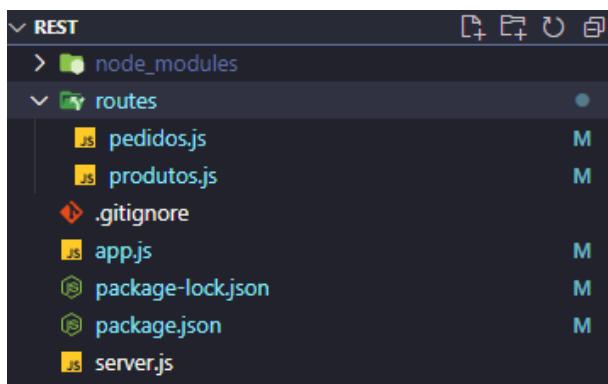
Simulação de requisições

- Para fazer simulação de requisições (testes de API REST), vamos utilizar o [Insomnia](#). Faça o download e o instale em seu computador.

The screenshot shows the Insomnia REST Client interface. At the top, it says 'Dashboard / Tutorial API - Celke ▾'. Below that, there's a search bar and some settings icons. The main area shows a request configuration: 'GET' method, 'localhost:3000' endpoint, and a 'Send' button. To the right, the status is '200 OK', '58.1 ms', and '52 B'. Below the request area, there's a preview pane showing the JSON response:
1 {
2 "mensagem": "Primeiro teste realizado com sucesso!"
3 }

Aula 02 - Criando as rotas

- Na pasta raiz do projeto crie uma pasta chamada **routes** e dentro dela dois arquivos, um chamado '**products.js**' e outro chamado '**pedidos.js**':



routes\produtos.js

```
const express = require('express');
const router = express.Router();

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  res.status(200).send({
    mensagem: "Lista todos os produtos"
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  res.status(201).send({
    mensagem: "Insere um produto!"
  });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Exibe detalhes do produto de id " + id,
  });
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Altera os dados do produto de id " + id
  });
});

// Exclui um produto
```

```
router.delete('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Exclui o produto de id " + id
  });
});

module.exports = router;
```

routes\pedidos.js

```
const express = require('express');
const router = express.Router();

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  res.status(200).send({
    mensagem: "Lista todos os pedidos"
  });
});

// Insere um pedido
router.post('/', (req, res, next) => {
  res.status(201).send({
    mensagem: "Insere um pedido!"
  });
});

// Retorna os dados de um pedido
router.get('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Exibe detalhes do pedido de id " + id,
  });
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Altera os dados do pedido de id " + id
  });
});

// Exclui um pedido
router.delete('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Exclui o pedido de id " + id
  });
});

module.exports = router;
```

app.js

```
const express = require('express');
const app = express();
const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);

module.exports = app;
```

Produtos:

GET ▾	localhost:3000/produtos	Send	200 OK	37.9 ms	38 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs	Preview ▾	Header	Cookie	Timeline	

```
1 + {
2   "mensagem": "Lista todos os produtos"
3 }
```

POST ▾	localhost:3000/produtos	Send	201 Created	6.69 ms	33 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs	Preview ▾	Header	Cookie	Timeline	

```
1 + {
2   "mensagem": "Insere um produto!"
3 }
```

GET ▾	localhost:3000/produtos/1	Send	200 OK	6.4 ms	48 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs	Preview ▾	Header	Cookie	Timeline	

```
1 + {
2   "mensagem": "Exibe detalhes do produto de id 1"
3 }
```

PATCH ▾	localhost:3000/produtos/1	Send	200 OK	9.04 ms	49 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs	Preview ▾	Header	Cookie	Timeline	

```
1 + {
2   "mensagem": "Altera os dados do produto de id 1"
3 }
```

DELETE ▾	localhost:3000/produtos/1	Send	200 OK	6.36 ms	39 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs	Preview ▾	Header	Cookie	Timeline	

```
1 + {
2   "mensagem": "Exclui o produto de id 1"
3 }
```

Pedidos:

GET ▾	localhost:3000/pedidos	Send	200 OK	4.3 ms	37 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs	Preview ▾	Header	Cookie	Timeline	

```
1 + {
2   "mensagem": "Lista todos os pedidos"
3 }
```

POST	localhost:3000/pedidos	Send	201 Created	6.5 ms	32 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header 7 Cookie Timeline

```
1 v {  
2   "mensagem": "Insere um pedido!"  
3 }
```

GET	localhost:3000/pedidos/1	Send	200 OK	4.39 ms	47 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header 7 Cookie Timeline

```
1 v {  
2   "mensagem": "Exibe detalhes do pedido de id 1"  
3 }
```

PATCH	localhost:3000/pedidos/1	Send	200 OK	3.36 ms	48 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header 7 Cookie Timeline

```
1 v {  
2   "mensagem": "Altera os dados do pedido de id 1"  
3 }
```

DELETE	localhost:3000/pedidos/1	Send	200 OK	6.43 ms	38 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header 7 Cookie Timeline

```
1 v {  
2   "mensagem": "Exclui o pedido de id 1"  
3 }
```

Aula 03 - Melhorando o Projeto e Tratando Erros

Instalando o nodemon

O **nodemon** é utilizado para não ser necessário parar e reiniciar o servidor a cada mudança realizada no projeto.

```
npm install --save-dev nodemon
```

```
C:\maransatto\rest>npm install --save-dev nodemon
> nodemon@2.0.7 postinstall C:\maransatto\rest\node_modules\nodemon
> node bin/postinstall || exit 0

npm [WARN] optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.1 (node_modules\chokidar\node_modules\fsevents):
npm [WARN] notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm [WARN] rest-api@1.0.0 No repository field.

+ nodemon@2.0.7
added 117 packages from 53 contributors and audited 168 packages in 50.457s

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

found 0 vulnerabilities
```

package.json

```
{
  "name": "rest-api",
  "version": "1.0.0",
  "description": "Teste de REST API",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon server.js"
  },
  "keywords": [
    "node",
    "API"
  ],
  "author": "Roberto Pinheiro",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

```
npm start
```

```
C:\maransatto\rest>npm start

> rest-api@1.0.0 start C:\maransatto\rest
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Servidor rodando na porta 3000
|
```

Instalando o morgan

```
npm install --save morgan
```

```
C:\maransatto\rest>npm install --save morgan
npm WARN rest-api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ morgan@1.10.0
added 4 packages from 2 contributors and audited 172 packages in 5.211s

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

found 0 vulnerabilities
```

app.js

```
const express = require('express');
const app = express();
const morgan = require('morgan');

const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');

app.use(morgan('dev'));

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);

module.exports = app;
```

- Ao realizar as requisições de produtos, por exemplo, no insomnia temos:

```
[nodemon] starting `node server.js`
Servidor rodando na porta 3000
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Servidor rodando na porta 3000
GET /produtos 200 8.685 ms - 38
GET /produtos/1 200 1.865 ms - 48
PATCH /produtos/1 200 1.148 ms - 49
DELETE /produtos/1 200 0.898 ms - 39
POST /produtos 201 1.218 ms - 33
[]
```

Tratamento para rota não encontrada

app.js

```
const express = require('express');
const app = express();
const morgan = require('morgan');

const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');

app.use(morgan('dev'));

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);

// Tratamento quando não for encontrada a rota
app.use((req, res, next) => {
  const erro = new Error('Rota não encontrada!');
  erro.status = 404;
  next(erro);
});

app.use((error, req, res, next) => {
  res.status(error.status) || 500;
  return res.send({
    erro: {
      mensagem: error.message
    }
  });
});

module.exports = app;
```

GET	localhost:3000/produto	Send	404 Not Found	47.7 ms	45 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header

```
1 + {
2 +   "erro": {
3 +     "mensagem": "Rota não encontrada!"
4 +   }
5 }
```

```
C:\maransatto\rest>npm start

> rest-api@1.0.0 start C:\maransatto\rest
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Servidor rodando na porta 3000
GET /produto 404 10.485 ms - 45
█
```

Aula 04 - Definindo o Body e tratando CORS

Instalando o body-parser

```
npm install --save body-parser
```

```
C:\maransatto\rest>npm install --save body-parser
npm WARN rest-api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ body-parser@1.19.0
updated 1 package and audited 173 packages in 5.463s

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

found 0 vulnerabilities
```

app.js

```
const express = require('express');
const app = express();
const morgan = require('morgan');
const bodyParser = require('body-parser');

const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');

app.use(morgan('dev'));
app.use(bodyParser.urlencoded({extended:false})); // apenas dados simples
app.use(bodyParser.json()); // json de entrada no body

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);

// Tratamento quando não for encontrada a rota
app.use((req, res, next) => {
  const erro = new Error('Rota não encontrada!');
  erro.status = 404;
  next(erro);
});

app.use((error, req, res, next) => {
  res.status(error.status) || 500;
  return res.send({
    erro: {
      mensagem: error.message
    }
  });
});

module.exports = app;
```

routes\produtos.js

```
const express = require('express');
const router = express.Router();

// Retorna todos os produtos
router.get('/', (req, res, next) => {
    res.status(200).send({
        mensagem: "Lista todos os produtos"
    });
});

// Insere um produto
router.post('/', (req, res, next) => {
    const produto = {
        nome: req.body.nome,
        preco: req.body.preco
    };

    res.status(201).send({
        mensagem: "Insere um produto!",
        produtoCriado: produto
    });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
    const id = req.params.id;

    res.status(200).send({
        mensagem: "Exibe detalhes do produto de id " + id,
    });
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
    const id = req.params.id;

    res.status(200).send({
        mensagem: "Altera os dados do produto de id " + id
    });
});

// Exclui um produto
router.delete('/:id', (req, res, next) => {
    const id = req.params.id;

    res.status(200).send({
        mensagem: "Exclui o produto de id " + id
    });
});

module.exports = router;
```

No Insomnia:

The screenshot shows the Insomnia REST Client interface. A POST request is made to `localhost:3000/produtos`. The request body is a JSON object with fields `nome` and `preco`. The response is a `201 Created` status with a response body containing a message and a created product object.

Send	201 Created	276 ms	106 B	2 Minutes Ago
JSON	Auth	Query	Header 1	Docs
1 v { 2 "nome": "Harry Potter e a Pedra Filosofal", 3 "preco": 59.90 4 }	Preview	Header 7	Cookie	Timeline
	1 v { 2 "mensagem": "Insere um produto!", 3 "produtoCriado": { 4 "nome": "Harry Potter e a Pedra Filosofal", 5 "preco": 59.9 6 } 7 }			

routes\pedidos.js

```
const express = require('express');
const router = express.Router();

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  res.status(200).send({
    mensagem: "Lista todos os pedidos"
  });
});

// Insere um pedido
router.post('/', (req, res, next) => {

  const pedido = {
    id_produto: req.body.id_produto,
    quantidade: req.body.quantidade
  };

  res.status(201).send({
    mensagem: "Insere um pedido!",
    pedidoCriado: pedido
  });
});

// Retorna os dados de um pedido
router.get('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Exibe detalhes do pedido de id " + id,
  });
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
  const id = req.params.id;

  res.status(200).send({
    mensagem: "Altera os dados do pedido de id " + id
  });
});

// Exclui um pedido
router.delete('/:id', (req, res, next) => {
  const id = req.params.id;
```

```

    res.status(200).send({
      mensagem: "Exclui o pedido de id " + id
    });
  });

module.exports = router;

```

POST		localhost:3000/pedidos	Send	201 Created	178 ms	79 B	Just Now
JSON	Auth	Query	Header 1	Docs			
				Preview	Header	Cookie	Timeline
<pre> 1 v { 2 "id_produto": 1, 3 "quantidade": 2 4 }</pre>							
<pre> 1 v { 2 "mensagem": "Insere um pedido!", 3 "pedidoCriado": { 4 "id_produto": 1, 5 "quantidade": 2 6 } 7 }</pre>							

Instalando o CORS

CORS = Cross Origin Resource Sharing

CORS é uma funcionalidade do HTML que permite que um site acesse um outro site dependendo de algumas restrições.

app.js

```

const express = require('express');
const app = express();
const morgan = require('morgan');
const bodyParser = require('body-parser');

const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');

app.use(morgan('dev'));
app.use(bodyParser.urlencoded({ extended: false })); // Apenas dados simples
app.use(bodyParser.json()); // json de entrada no body

app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers',
  'Origin, X-Requested-With, Content-Type, Accept, Authorization');

  if(req.method === 'OPTIONS'){
    res.header('Access-Control-Allow-Methods',
    'GET, POST, PUT, PATCH, DELETE');
    return res.status(200).send({});
  }
  next();
});

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);

// Tratamento quando não for encontrada rota
app.use((req, res, next) => {
  const erro = new Error('Rota não encontrada!');
  erro.status = 404;
  next(erro);
}

```

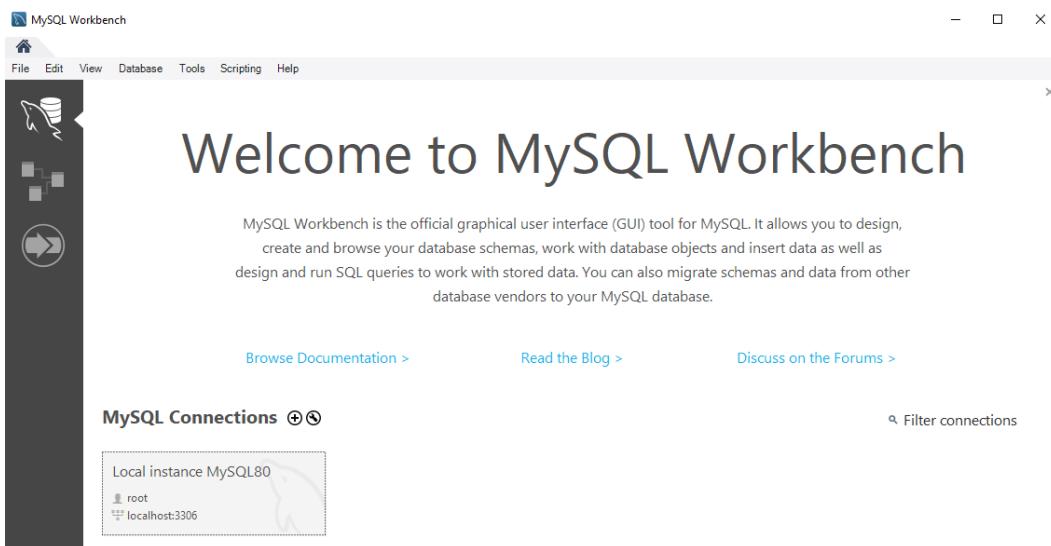
```
});

app.use((error, req, res, next) => {
  res.status(error.status) || 500;
  return res.send({
    erro: {
      mensagem: error.message
    }
  });
});

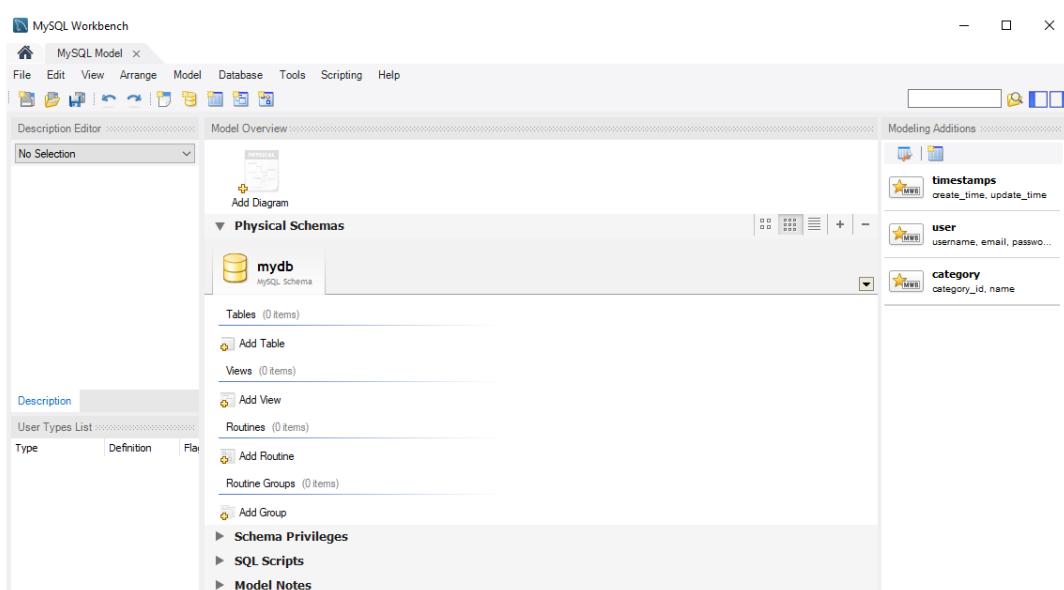
module.exports = app;
```

Aula 05 - Criando o Banco de Dados (MySQL)

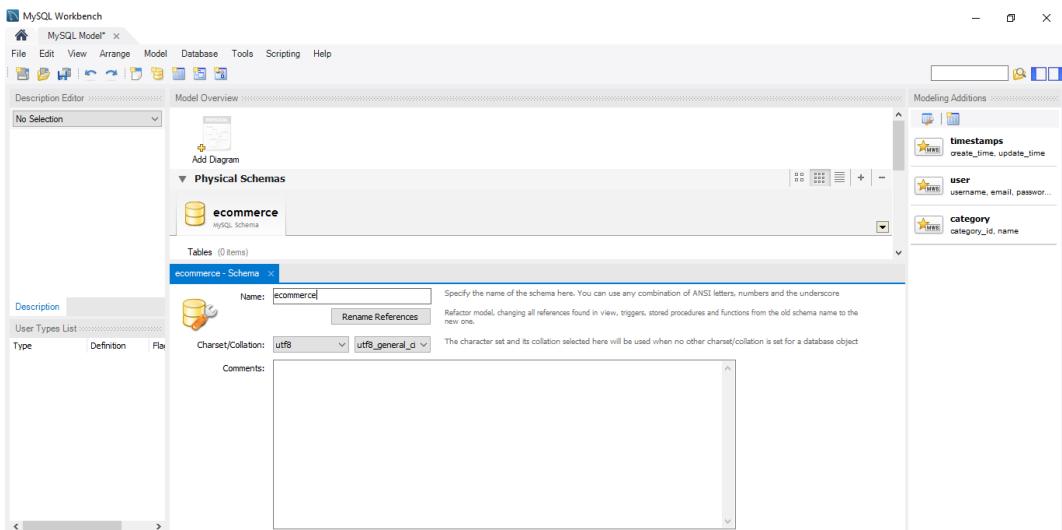
Usando o Workbench



- Adicione um novo Model

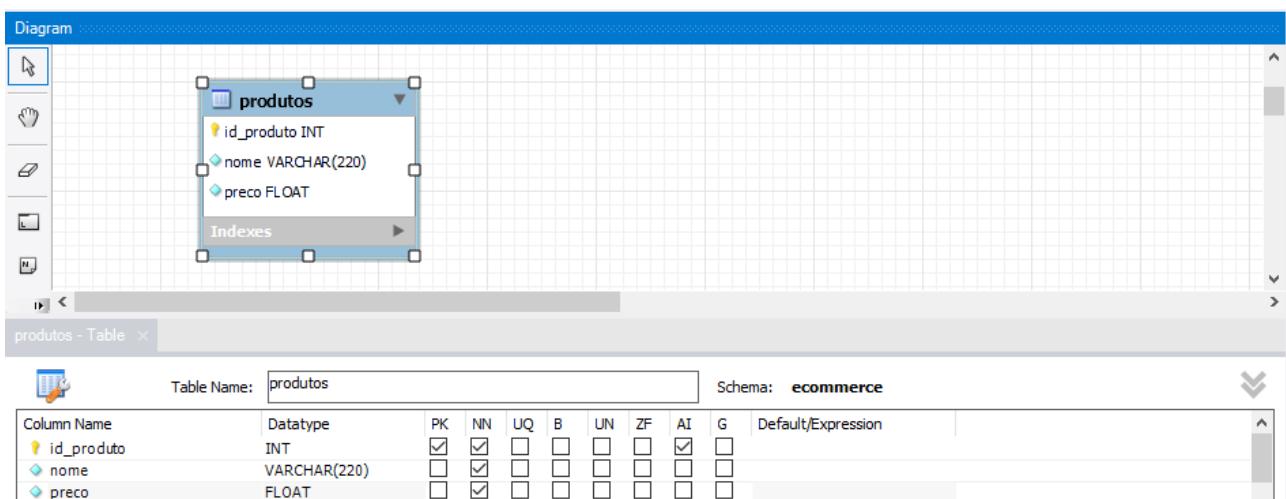


- Salve (<Ctrl><S>) o Model, na pasta do projeto, como **modelagem - ecommerce**

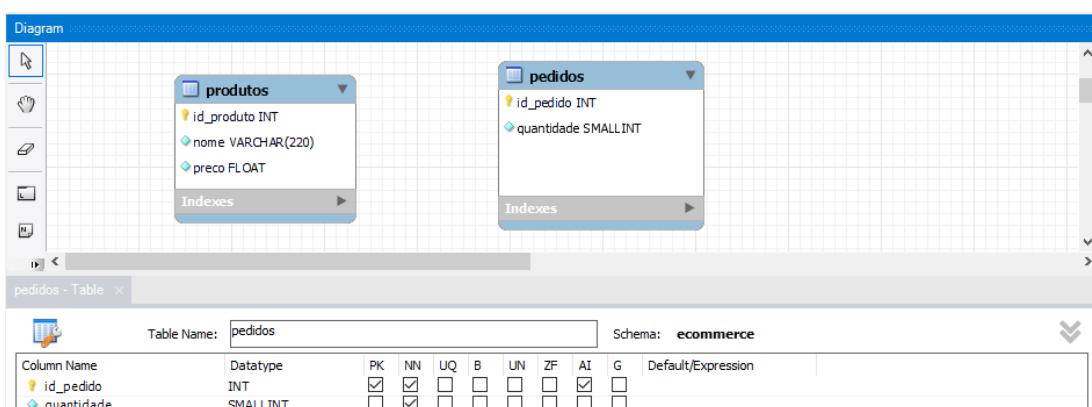


- Clique no botão **Add Diagram**

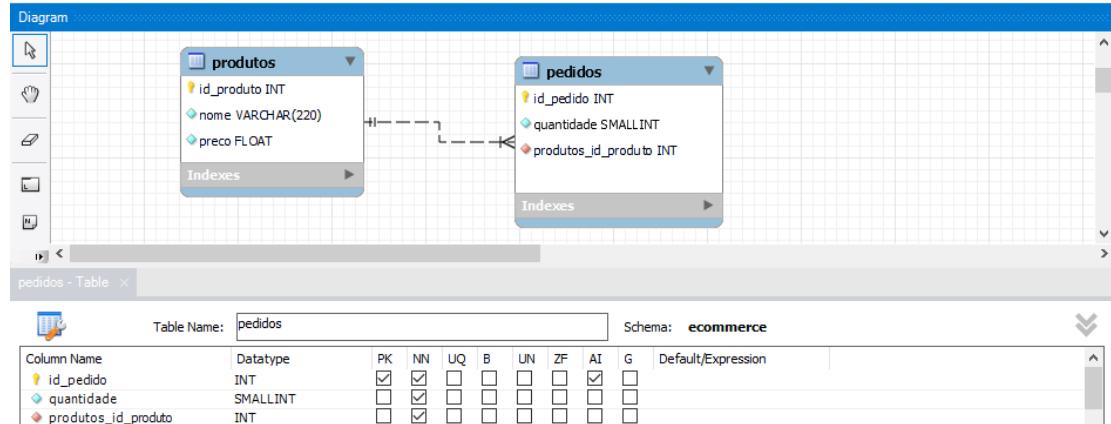
- Crie a tabela **produtos**



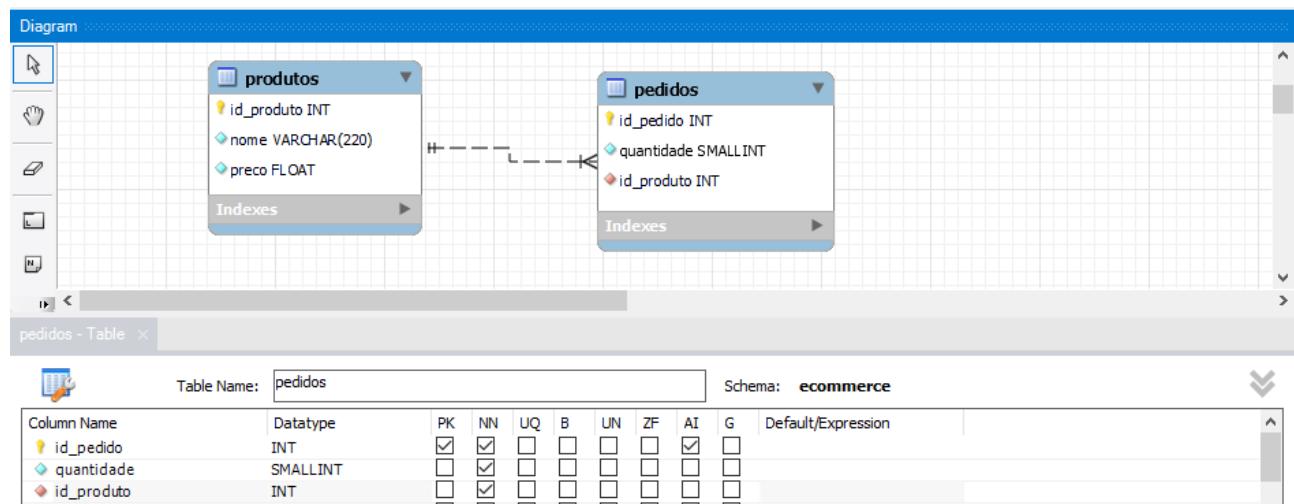
- Crie a tabela **pedidos**



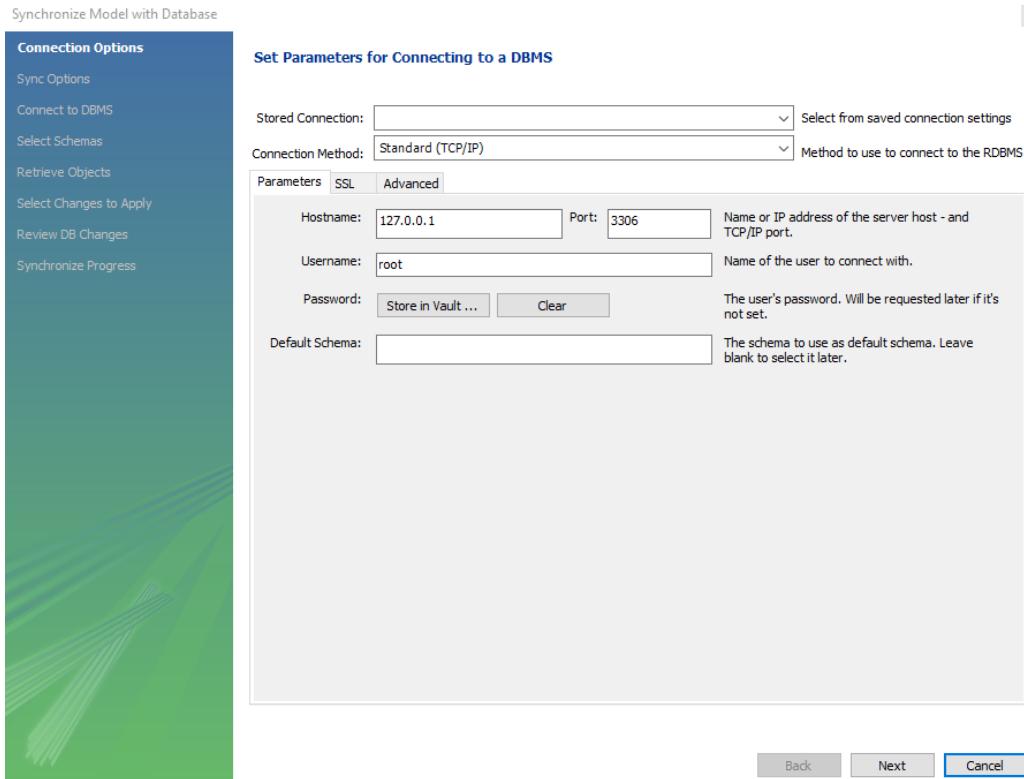
- um produto pode ter vários pedidos (relação 1 para n)



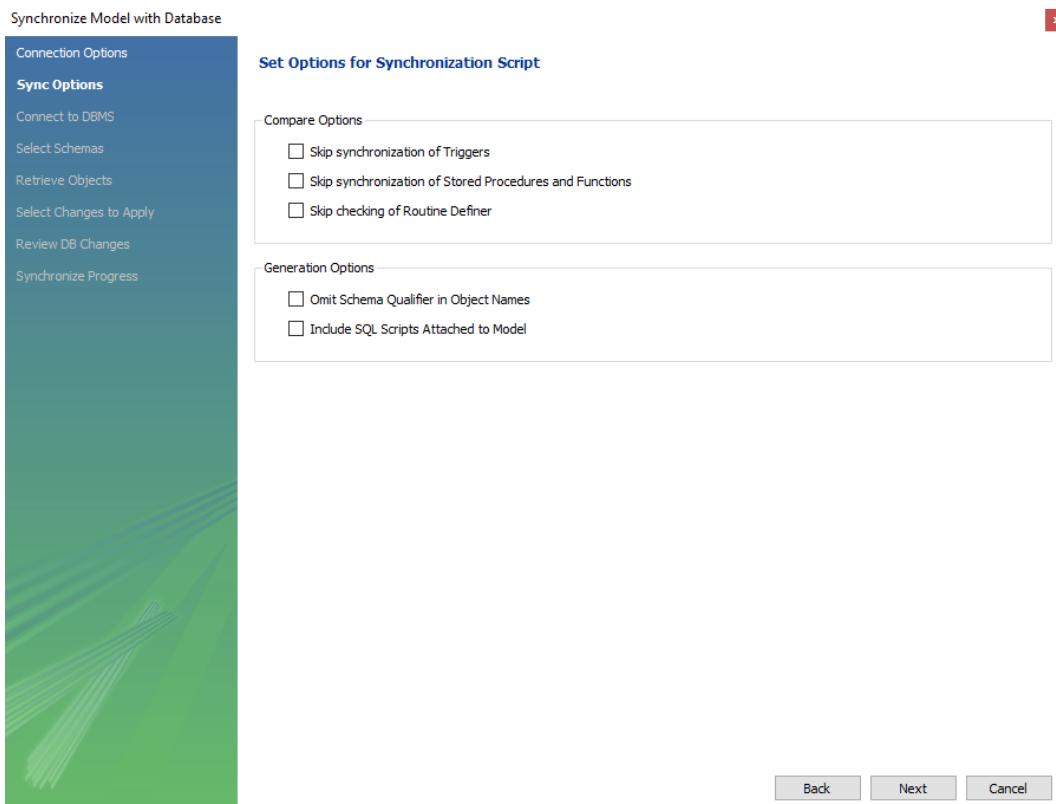
- Renomeie `produtos_id_produto` para `id_produto`:



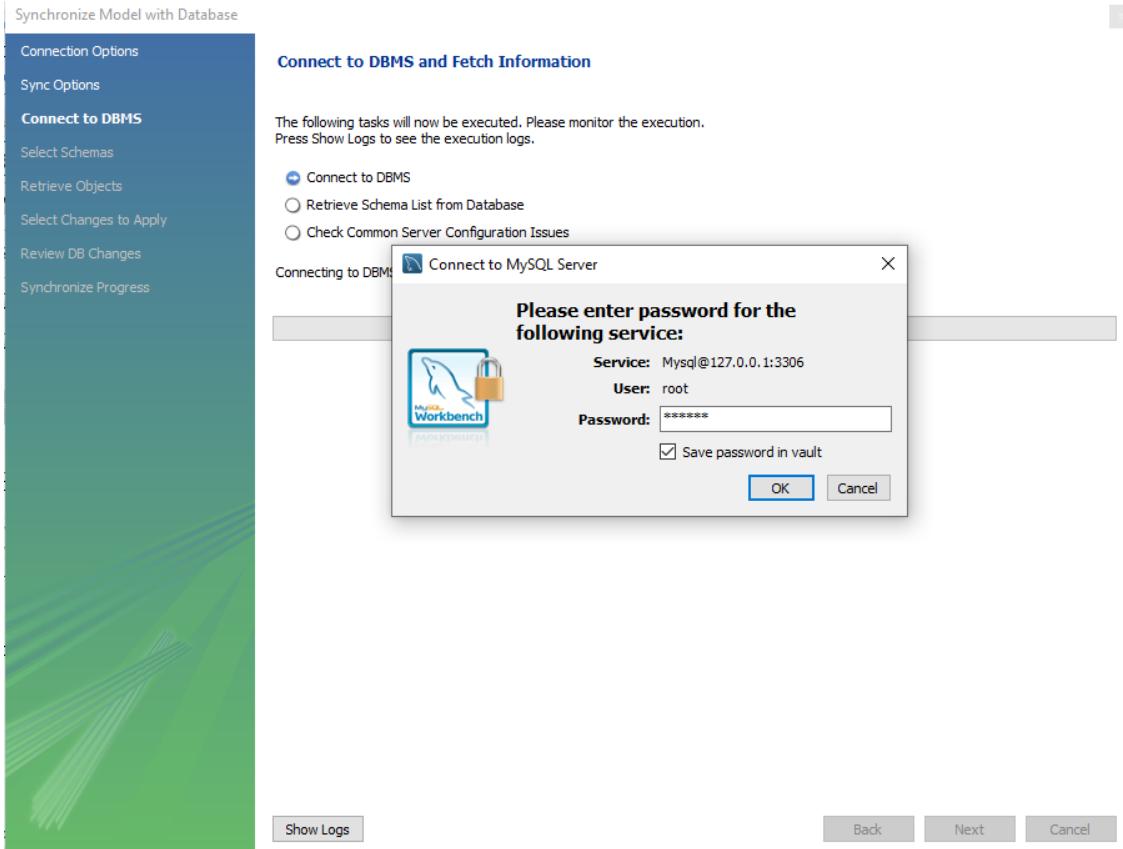
- Clique em Database -> Syncronize Model



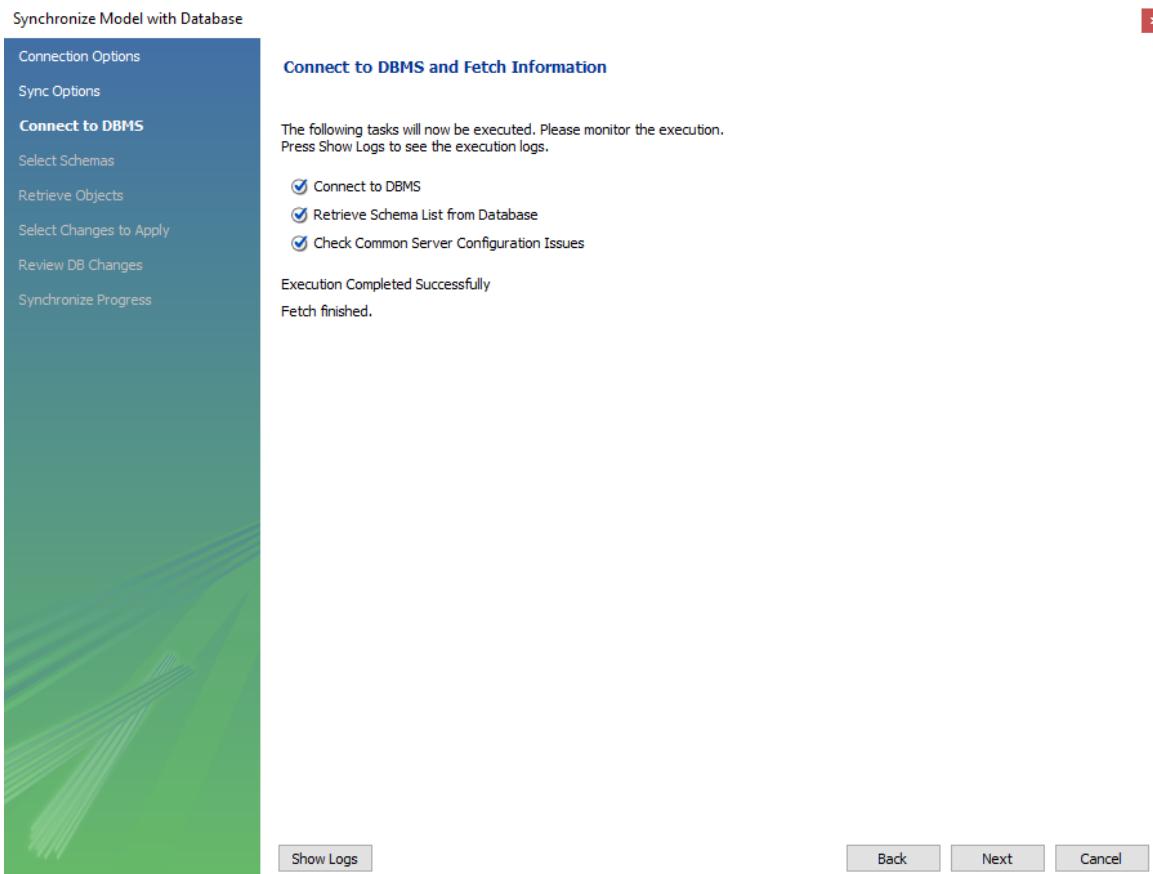
- Clique no botão Next



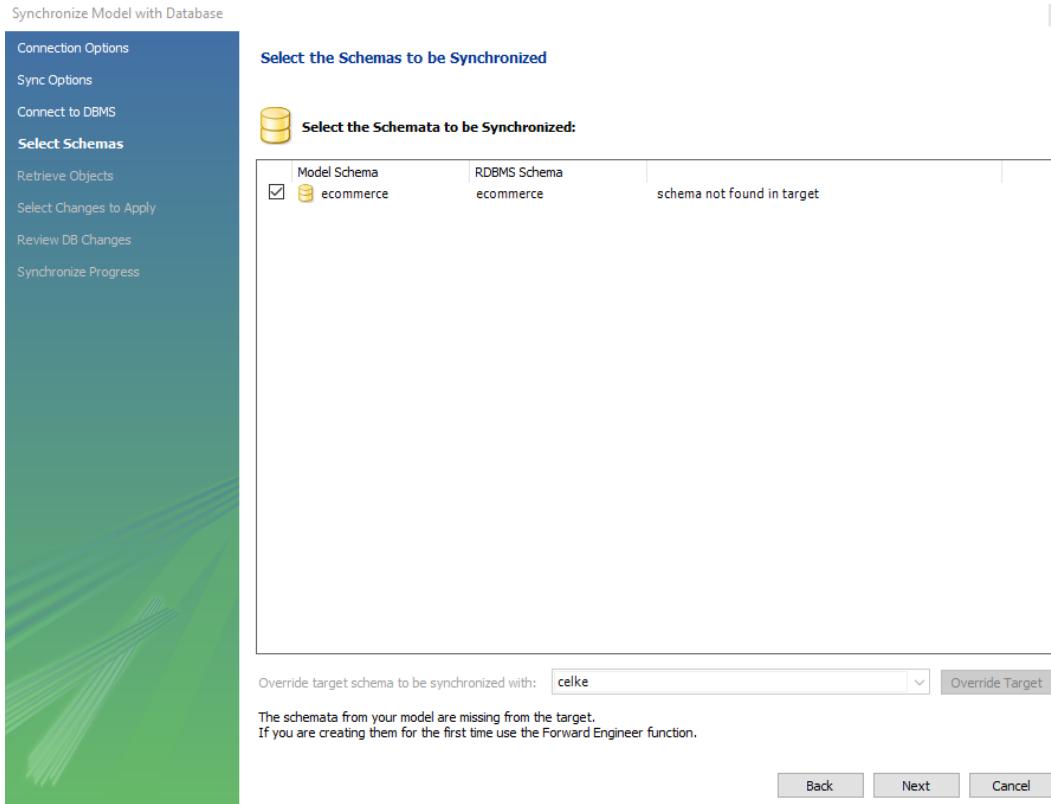
- Clique no botão Next



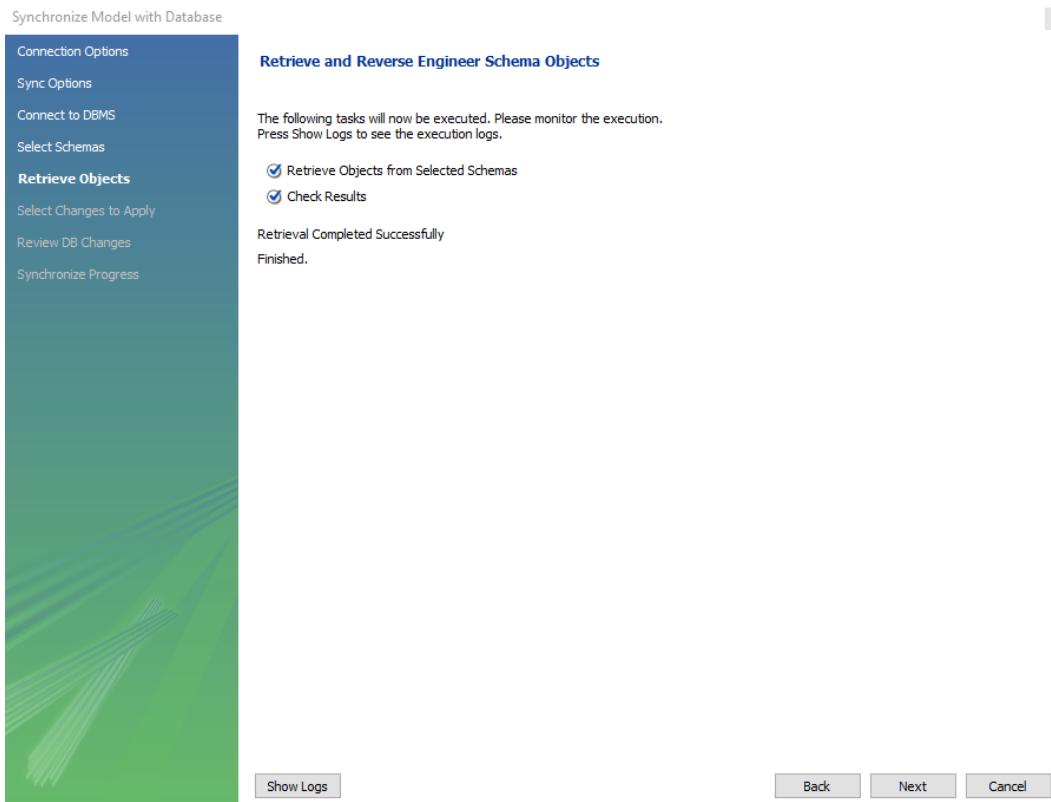
- Insira a senha (123456) e clique no botão Ok



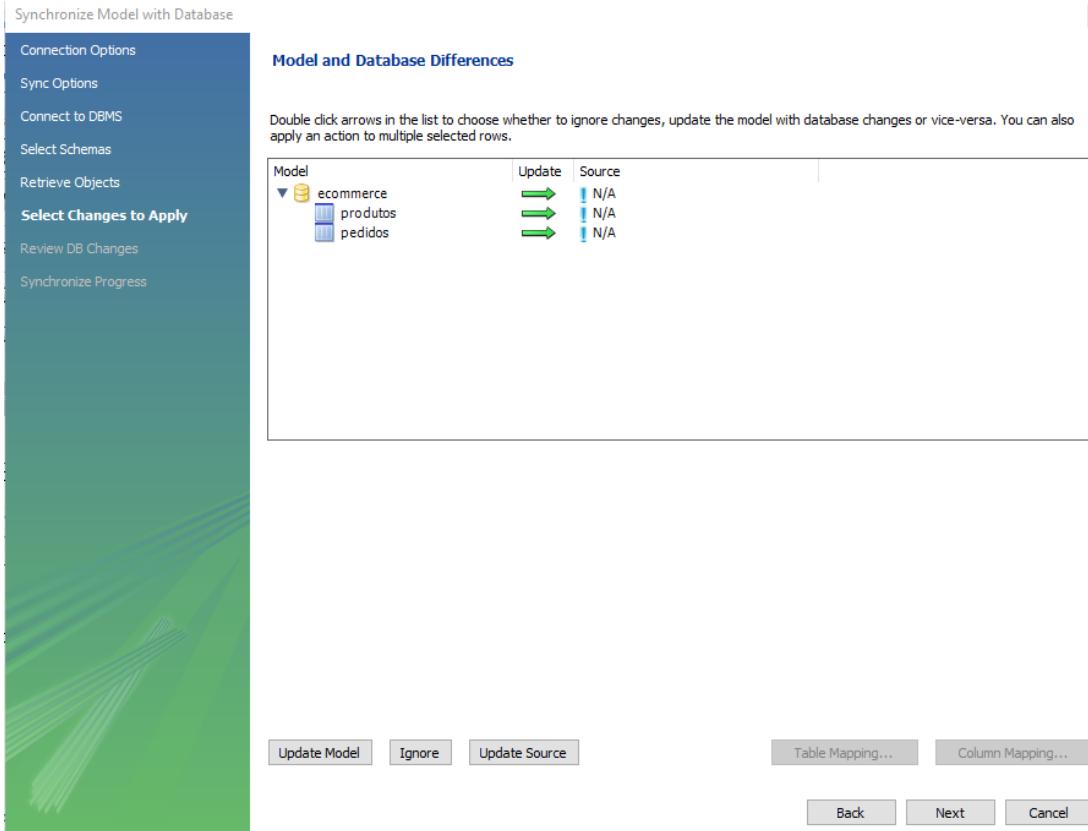
- Clique no botão **Next**



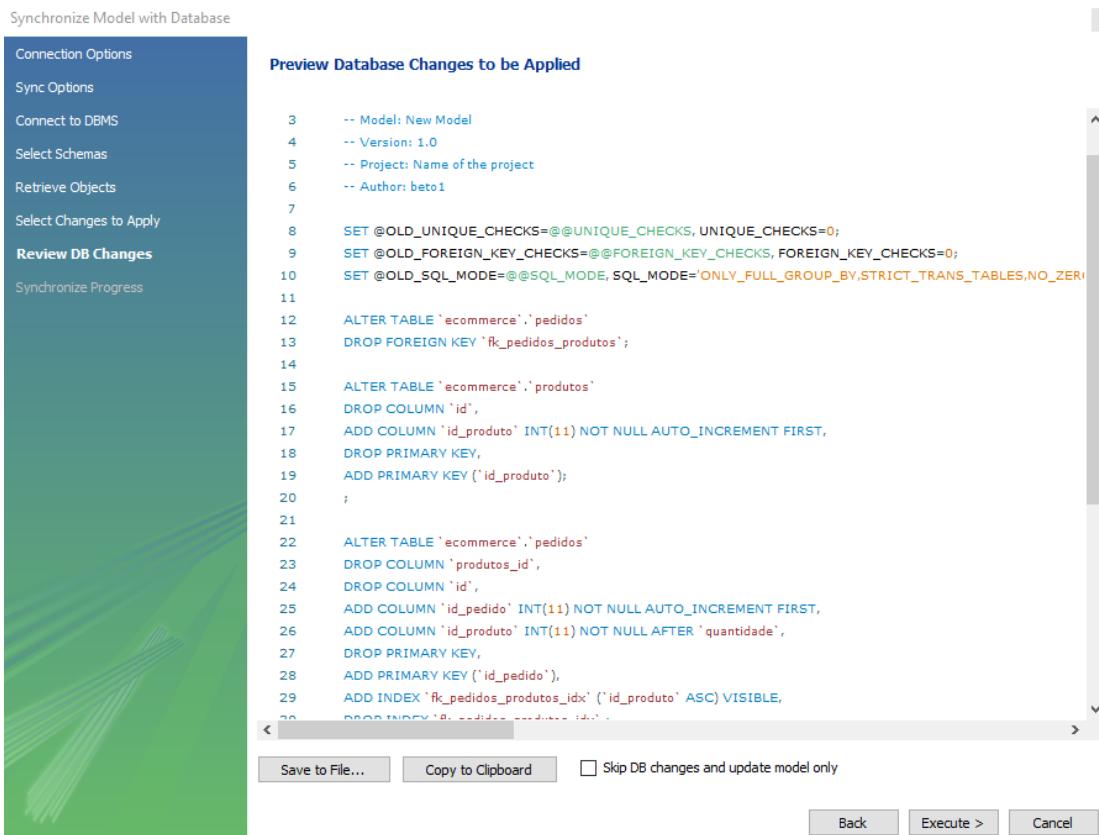
- Marque a caixa **ecommerce** e clique no botão **Next**



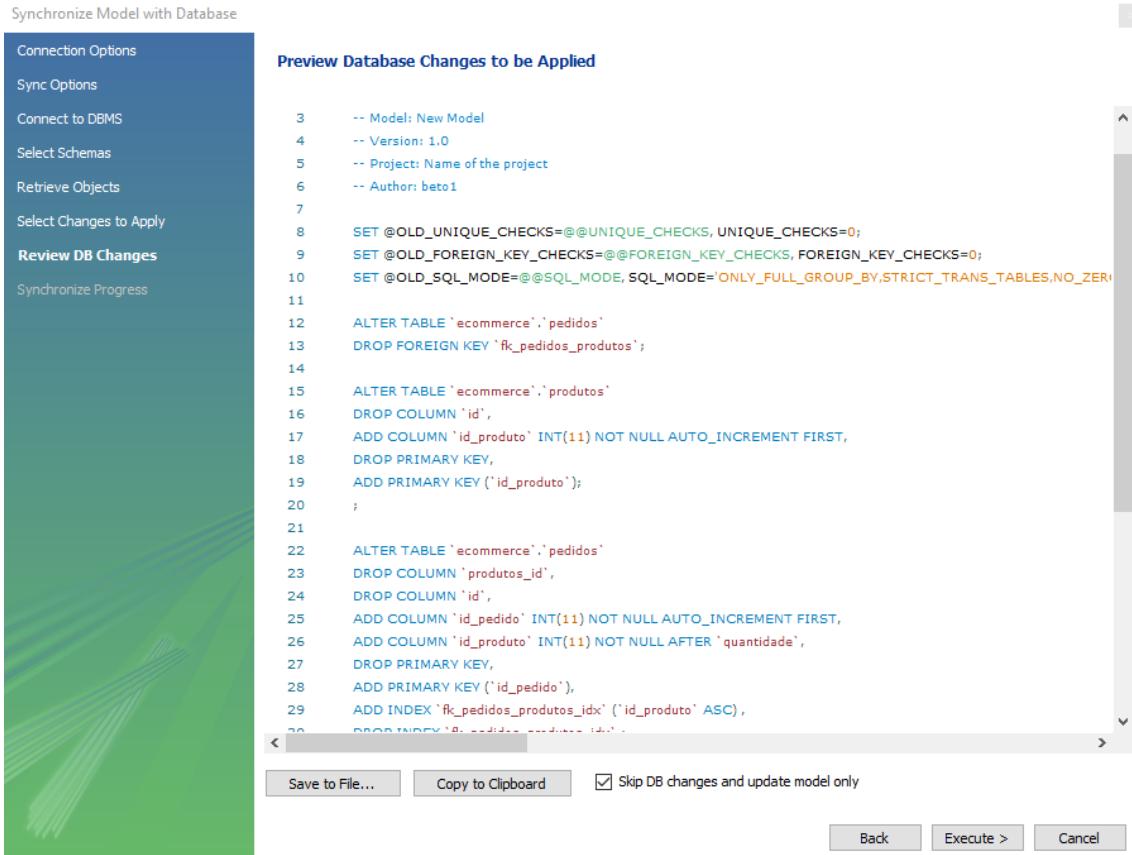
- Clique no botão **Next**



- Clique no botão **Next**



- Remova a cláusula **VISIBLE** existente na linha 29



-- MySQL Workbench Synchronization

-- Generated: 2021-05-18 16:25

-- Model: New Model

-- Version: 1.0

-- Project: Name of the project

-- Author: beto1

```

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

```

```

ALTER TABLE `ecommerce`.`pedidos`
DROP FOREIGN KEY `fk_pedidos_produtos`;

```

```

ALTER TABLE `ecommerce`.`produtos`
DROP COLUMN `id`,
ADD COLUMN `id_produto` INT(11) NOT NULL AUTO_INCREMENT FIRST,
DROP PRIMARY KEY,
ADD PRIMARY KEY (`id_produto`);
;
```

```

ALTER TABLE `ecommerce`.`pedidos`
DROP COLUMN `produtos_id`,
DROP COLUMN `id`,
ADD COLUMN `id_pedido` INT(11) NOT NULL AUTO_INCREMENT FIRST,
ADD COLUMN `id_produto` INT(11) NOT NULL AFTER `quantidade`,

```

```

DROP PRIMARY KEY,
ADD PRIMARY KEY (`id_pedido`),
ADD INDEX `fk_pedidos_produtos_idx`(`id_produto` ASC),
DROP INDEX `fk_pedidos_produtos_idx`;
;

```

```

ALTER TABLE `ecommerce`.`pedidos`
ADD CONSTRAINT `fk_pedidos_produtos`
FOREIGN KEY (`id_produto`)
REFERENCES `ecommerce`.`produtos`(`id_produto`)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

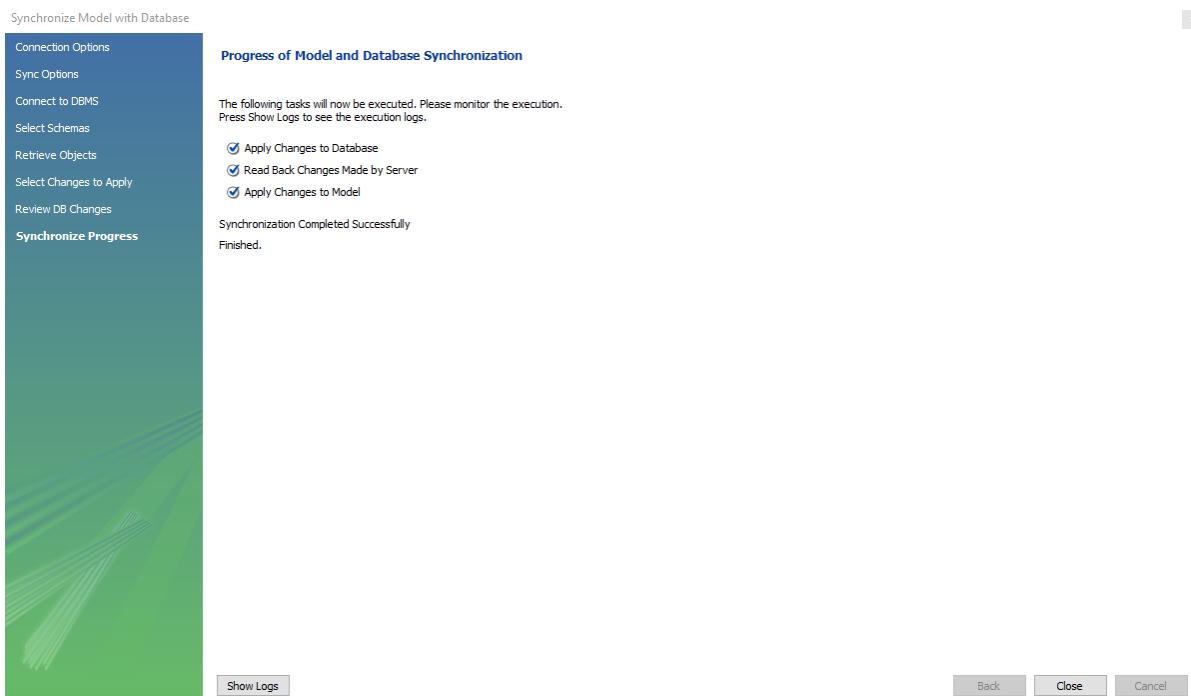
```

```

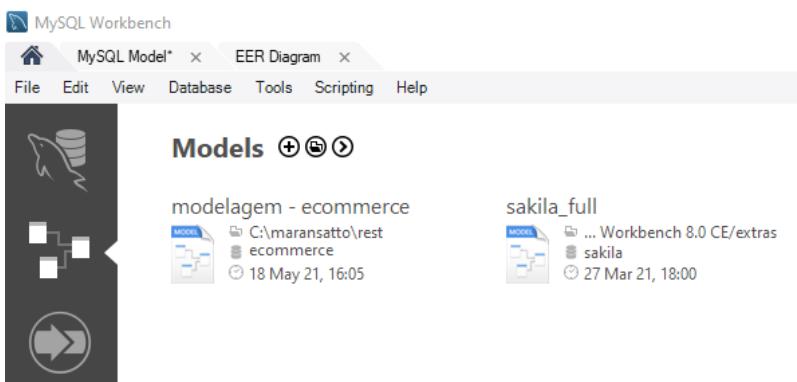
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

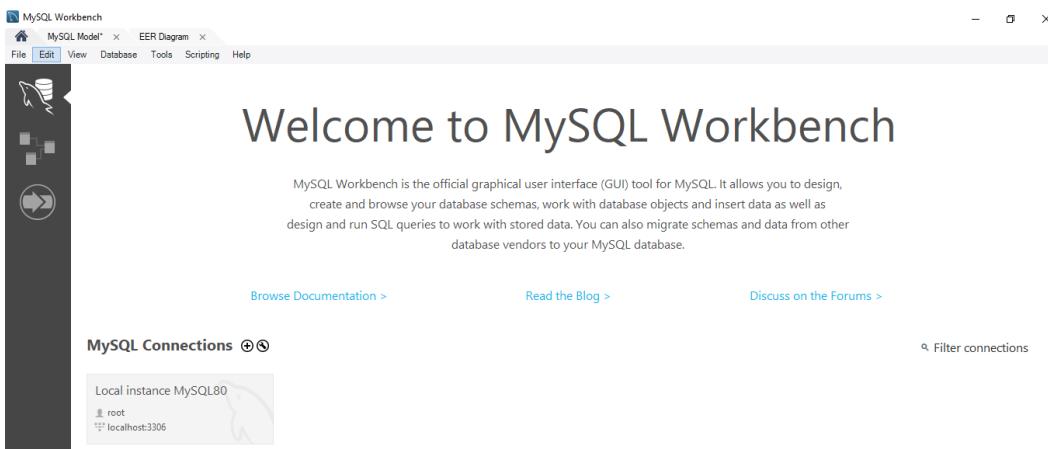
```

- Clique no botão **Execute**



- Clique no botão **Close**. O banco foi sincronizado.





- Clique no BD ecommerce

This screenshot shows the "SCHEMAS" tree view in MySQL Workbench. The tree is organized by database. Under the "ecommerce" database, there are several objects: Tables (pedidos, produtos), Views, Stored Procedures, Functions, and other databases like postapp, sistema_cadastro, sys, and test. A search bar labeled "Filter objects" is located at the top of the tree view.

Selecione o BD:

use ecommerce;

This screenshot shows the MySQL Workbench interface with a SQL editor and an output window. The SQL editor tab is titled "SQL File 22*" and contains the command "use ecommerce;". The output window is titled "Output" and shows the result of the command: "1 16:40:52 use ecommerce".

select * from produtos

1 • SELECT * FROM produtos;

This screenshot shows the "Result Grid" window in MySQL Workbench. It displays the results of the query "SELECT * FROM produtos;". The grid has three columns: id_produto, nome, and preco. There is one row of data with all values set to NULL. The top of the grid includes buttons for "Result Grid", "Edit", "Export/Import", and "Wrap Cell Content".

- Insira um produto:

```
insert into produtos (nome, preco) values ('Harry Potter', 99.60);
```

- Clique no botão Execute the statement under the keyboard cursor.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is the SQL editor pane containing two statements:
1 • `SELECT * FROM produtos;`
2 • `insert into produtos (nome, preco) values ('Harry Potter', 99.60);`

Below the SQL editor is the Result Grid pane. It has a header row with columns: id_produto, nome, and preco. The data grid shows one row with values: 1, Harry Potter, and 99.6. The 'nome' column contains 'NULL' because the value was not explicitly provided in the insert statement.

	id_produto	nome	preco
▶	1	Harry Potter	99.6
*	HULL	HULL	HULL

Aula 06 - Conectando a API ao Banco MySQL

Instalando o package do mysql

```
npm install --save mysql2
```

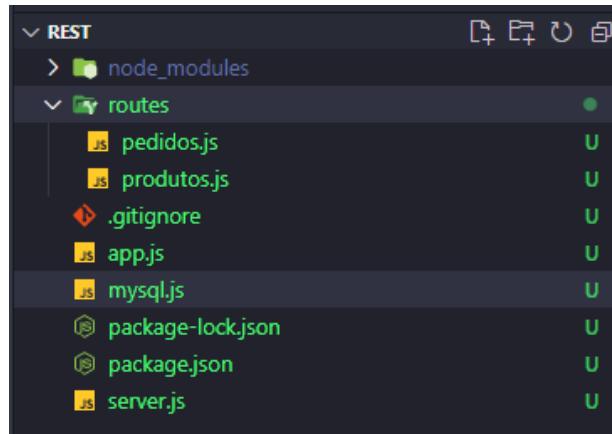
```
C:\maransatto\rest>npm install --save mysql2
npm WARN rest-api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ mysql2@2.2.5
updated 1 package and audited 195 packages in 8.343s

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

found 0 vulnerabilities
```

- Na pasta do projeto, crie um arquivo chamado **mysql.js**



mysql.js

```
const mysql = require('mysql2');

var pool = mysql.createPool({
  "user": process.env.MYSQL_USER,
  "password": process.env.MYSQL_PASSWORD,
  "database": process.env.MYSQL_DATABASE,
  "host": process.env.MYSQL_HOST,
  "port": process.env.MYSQL_PORT
});

exports.pool = pool;
```

- Na pasta raiz do projeto crie um arquivo chamado **nodemon.json**

nodemon.json

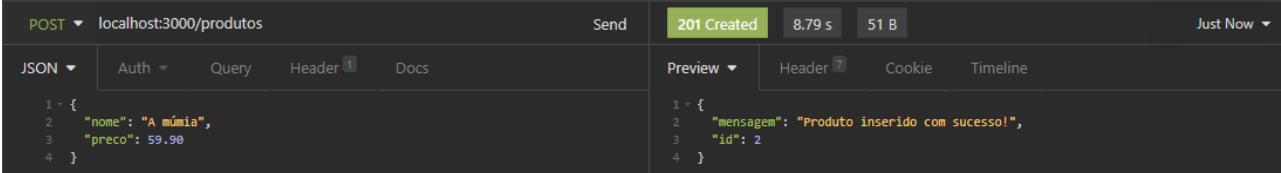
```
{  
  "env": {  
    "MYSQL_USER": "root",  
    "MYSQL_PASSWORD": "123456",  
    "MYSQL_DATABASE": "ecommerce",  
    "MYSQL_HOST": "localhost",  
    "MYSQL_PORT": 3306  
  }  
}
```

Inserindo um produto

routes\produtos.js

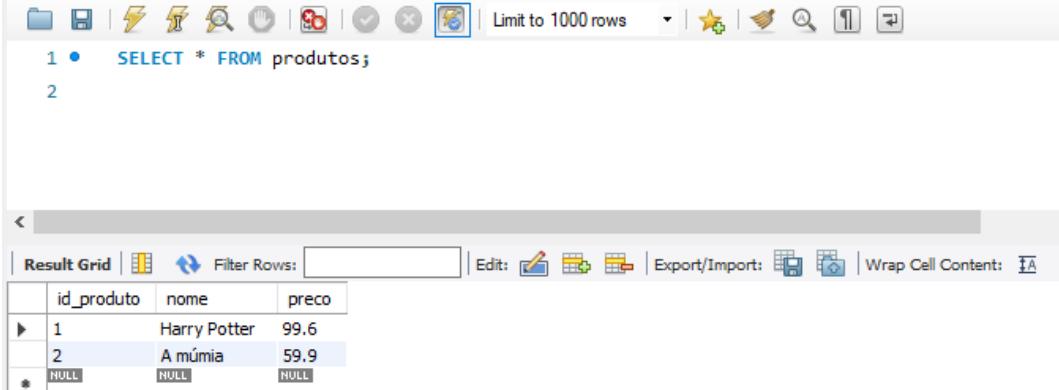
```
const express = require('express');  
const router = express.Router();  
const mysql = require('../mysql').pool;  
  
// Retorna todos os produtos  
router.get('/', (req, res, next) => {  
  res.status(200).send({  
    mensagem: "Lista todos os produtos"  
  });  
});  
  
// Insere um produto  
router.post('/', (req, res, next) => {  
  
  const produto = {  
    nome: req.body.nome,  
    preco: req.body.preco  
  };  
  
  mysql.getConnection((error, conn) =>{  
    if(error){ return res.status(500).send({error: error}) }  
    conn.query(  
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',  
      [req.body.nome, req.body.preco],  
      (error, resultado, field) => {  
        conn.release();  
        if(error){ return res.status(500).send({ error: error });}  
        res.status(201).send({  
          mensagem: "Produto inserido com sucesso!",  
          id: resultado.insertId  
        });  
      }  
    );  
  });  
  
  module.exports = router;
```

- No Insomnia:



```
POST localhost:3000/produtos
Send
201 Created 8.79 s 51 B Just Now
JSON Auth Query Header Docs
1: {
2:   "nome": "A múmia",
3:   "preco": 59.90
4: }
```

```
1: {
2:   "mensagem": "Produto inserido com sucesso!",
3:   "id": 2
4: }
```



```
1 •  SELECT * FROM produtos;
2
```

	id_produto	nome	preco
▶	1	Harry Potter	99.6
2	A múmia	59.9	
*	NULL	NULL	NULL

Listando todos os produtos cadastrados

routes\produtos.js

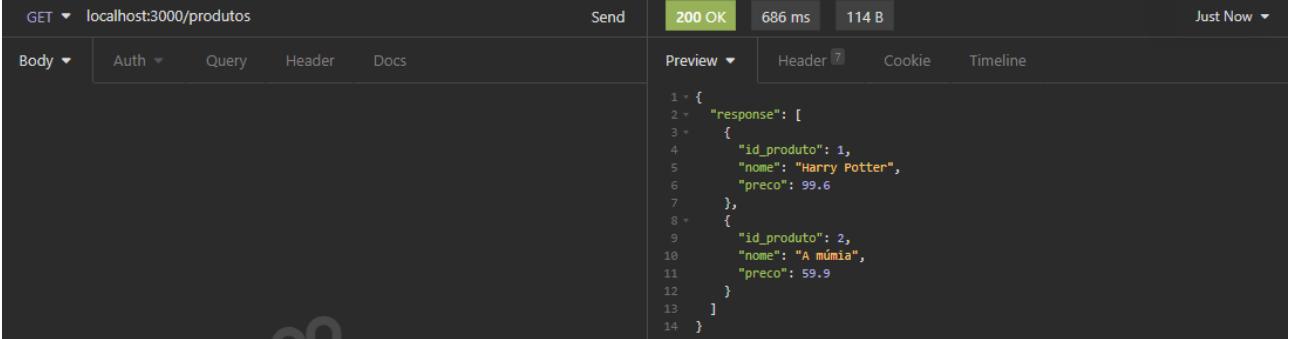
```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql2').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(201).send({
          mensagem: "Produto inserido com sucesso!",
          id: resultado.insertId
        });
      }
    );
  });
});

module.exports = router;
```

- No Insomnia:



GET ▾ localhost:3000/produtos Send 200 OK 686 ms 114 B Just Now ▾

Body ▾ Auth ▾ Query Header Docs Preview ▾ Header 7 Cookie Timeline

```
1 + {
2 +   "response": [
3 +     {
4 +       "id_produto": 1,
5 +       "nome": "Harry Potter",
6 +       "preco": 99.6
7 +     },
8 +     {
9 +       "id_produto": 2,
10 +      "nome": "A m\u00famia",
11 +      "preco": 59.9
12 +    }
13 +  ]
14 }
```

Exibindo detalhes de um produto

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

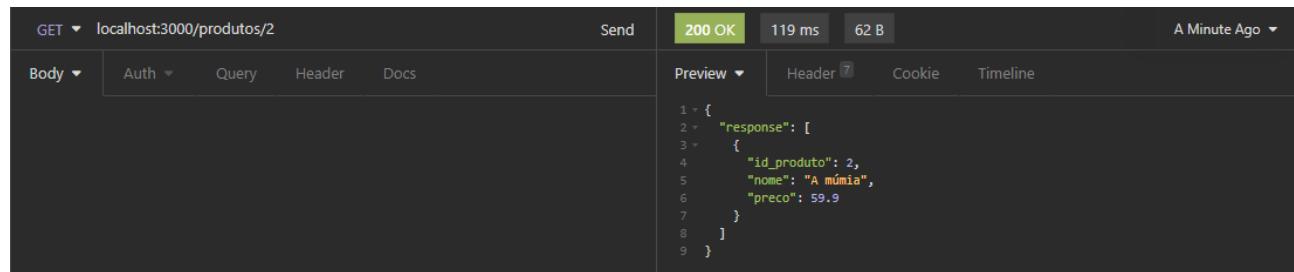
// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  })
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(201).send({
          mensagem: "Produto inserido com sucesso!",
          id: resultado.insertId
        });
      }
    )
  });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  })
});
```

```
});  
});  
  
module.exports = router;
```

- No Insomnia:



GET ▾ localhost:3000/produtos/2 Send 200 OK 119 ms 62 B A Minute Ago ▾

Body ▾ Auth ▾ Query Header Docs Preview ▾ Header 7 Cookie Timeline

```
1 ~ {  
2 ~   "response": [  
3 ~     {  
4 ~       "id_produto": 2,  
5 ~       "nome": "A m\u00famia",  
6 ~       "preco": 59.9  
7 ~     }  
8 ~   ]  
9 }
```

Editando um produto

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  })
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(201).send({
          mensagem: "Produto inserido com sucesso!",
          id: resultado.insertId
        });
      }
    )
  });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  });
});
```

```

        })
    });

// Altera um produto
router.patch('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
      [req.body.nome, req.body.preco, req.params.id],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(200).send({
          mensagem: "Produto alterado com sucesso!",
        });
      }
    );
  });
});

module.exports = router;

```

- No Insomnia:

PATCH ▾	localhost:3000/produtos/2	Send	200 OK	515 ms	44 B	Just Now ▾
JSON ▾	Auth ▾ Query Header 1 Docs		Preview ▾	Header 7	Cookie	Timeline
	1 + { 2 "nome": "A Múmia 2", 3 "preco": 69.98 4 }		1 + { 2 "mensagem": "Produto alterado com sucesso!" 3 }			

GET ▾	localhost:3000/produtos	Send	200 OK	83 ms	116 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs		Preview ▾	Header 7	Cookie	Timeline
			1 + { 2 "response": [3 { 4 "id_produto": 1, 5 "nome": "Harry Potter", 6 "preco": 99.6 7 }, 8 { 9 "id_produto": 2, 10 "nome": "A Múmia 2", 11 "preco": 69.9 12 } 13] 14 }			

Excluindo um produto

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  })
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(201).send({
          mensagem: "Produto inserido com sucesso!",
          id: resultado.insertId
        });
      }
    )
  });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, resultado, field) => {
        if(error){ return res.status(500).send({error: error}) };
        return res.status(200).send({response: resultado});
      }
    )
  });
});
```

```
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
      [req.body.nome, req.body.preco, req.params.id],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(200).send({
          mensagem: "Produto alterado com sucesso!",
        });
      }
    );
  });
});

// Exclui um produto
router.delete('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, resultado, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        res.status(200).send({
          mensagem: "Produto removido com sucesso!",
        });
      }
    );
  });
});

module.exports = router;
```

- No Insomnia:

DELETE	localhost:3000/produtos/2	Send	200 OK	544 ms	44 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header
					1 v { 2 v "mensagem": "Produto removido com sucesso!" 3 }	

GET	localhost:3000/produtos	Send	200 OK	96.3 ms	66 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header
					1 v { 2 v "response": [3 v { 4 v "id_produto": 1, 5 v "name": "Harry Potter", 6 v "preco": 99.6 7 v } 8 v] 9 }	

Aula 07 - API pública bem documentada (boas práticas)

Listando todos os produtos cadastrados

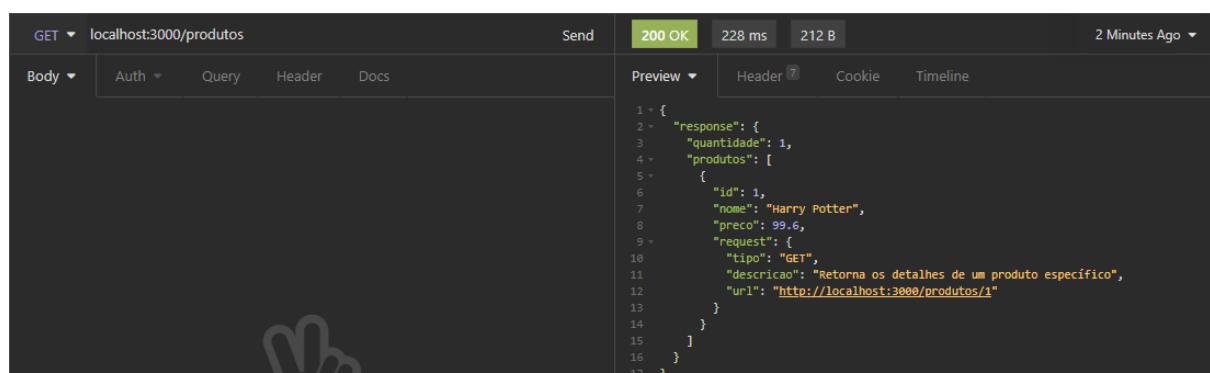
routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
});

module.exports = router;
```

- No Insomnia:



The screenshot shows the Insomnia REST Client interface. A successful GET request to `localhost:3000/produtos` is displayed. The response status is `200 OK`, with a response time of `228 ms` and a body size of `212 B`. The response was received `2 Minutes Ago`. The response body is a JSON object representing a single product:

```
1: {
2:   "response": {
3:     "quantidade": 1,
4:     "produtos": [
5:       {
6:         "id": 1,
7:         "nome": "Harry Potter",
8:         "preco": 99.6,
9:         "request": {
10:           "tipo": "GET",
11:           "descricao": "Retorna os detalhes de um produto específico",
12:           "url": "http://localhost:3000/produtos/1"
13:         }
14:       }
15:     ]
16:   }
17: }
```

Inserindo um produto

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    )
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
          }
        }
        return res.status(201).send(response);
      }
    );
  });
});
```

```

        request: {
          tipo: 'POST',
          descricao: 'Retorna todos os produtos',
          url: 'http://localhost:3000/produtos'
        }
      }
    }
  }

  return res.status(201).send(response);
}

});

module.exports = router;

```

POST ▾ localhost:3000/produtos Send **201 Created** 572 ms 198 B Just Now ▾

JSON ▾	Auth ▾	Query	Header 1	Docs	Preview ▾	Header 7	Cookie	Timeline
1 ~ { 2 "nome": "A múmia", 3 "preco": 59.90 4 }					1 ~ { 2 "mensagem": "Produto inserido com sucesso!", 3 "produtocriado": { 4 "nome": "A múmia", 5 "preco": 59.9, 6 "request": { 7 "tipo": "POST", 8 "descricao": "Retorna todos os produtos", 9 "url": "http://localhost:3000/produtos" 10 } 11 } 12 }			

← → C ⓘ localhost:3000/produtos

_apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstrap ...

```

1 // 20210518194549
2 // http://localhost:3000/produtos
3
4 ~ {
5   "response": {
6     "quantidade": 2,
7     "produtos": [
8       {
9         "id": 1,
10        "nome": "Harry Potter",
11        "preco": 99.6,
12        "request": {
13          "tipo": "GET",
14          "descricao": "Retorna os detalhes de um produto específico",
15          "url": "http://localhost:3000/produtos/1"
16        }
17      },
18      {
19        "id": 3,
20        "nome": "A múmia",
21        "preco": 59.9,
22        "request": {
23          "tipo": "GET",
24          "descricao": "Retorna os detalhes de um produto específico",
25          "url": "http://localhost:3000/produtos/3"
26        }
27      }
28    ]
29  }
30 }

```

Exibindo detalhes de um produto

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
```

```

        tipo: 'POST',
        descricao: 'Retorna todos os produtos',
        url: 'http://localhost:3000/produtos'
    }
}
}
return res.status(201).send(response);
}
)
});
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'SELECT * FROM produtos WHERE id_produto = ?',
[req.params.id],
(error, result, field) => {
if(error){ return res.status(500).send({error: error}) };
if(result.length == 0){
return res.status(404).send({
mensagem: "Não foi encontrado produto com este Id!"
})
}
const response = {
produto: {
id: result[0].id_produto,
nome: result[0].nome,
preco: result[0].preco,
request: {
tipo: 'GET',
descricao: 'Retorna todos os produtos',
url: 'http://localhost:3000/produtos'
}
}
}
return res.status(200).send(response);
}
)
});
});

module.exports = router;

```

GET ▾		localhost:3000/produtos/2	Send	404 Not Found	426 ms	55 B	3 Minutes Ago ▾
Body ▾				Preview ▾	Header 7	Cookie	Timeline
				1 ▾ { 2 "mensagem": "Não foi encontrado produto com este Id!" 3 }			

GET	localhost:3000/produtos/3	Send	200 OK	21.3 ms	155 B	Just Now
Body	Auth	Query	Header	Docs	Preview	Header

```

1 + {
2 +   "produto": {
3 +     "id": 3,
4 +     "nome": "A múmia",
5 +     "preco": 59.9,
6 +     "request": {
7 +       "tipo": "GET",
8 +       "descricao": "Retorna todos os produtos",
9 +       "url": "http://localhost:3000/produtos"
10    }
11  }
12 }

```

Editando um produto

routes\produtos.js

```

const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    )
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(

```

```

'INSERT INTO produtos (nome, preco) VALUES (?,?)',
[req.body.nome, req.body.preco],
(error, result, field) => {
  conn.release();
  if(error){ return res.status(500).send({error: error}) }
  const response = {
    mensagem: "Produto inserido com sucesso!",
    produtoCriado: {
      id: result.id_produto,
      nome: req.body.nome,
      preco: req.body.preco,
      request: {
        tipo: 'POST',
        descricao: 'Retorna todos os produtos',
        url: 'http://localhost:3000/produtos'
      }
    }
  }
  return res.status(201).send(response);
}
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Não foi encontrado produto com este Id!"
          })
        }
        const response = {
          produto: {
            id: result[0].id_produto,
            nome: result[0].nome,
            preco: result[0].preco,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os produtos',
              url: 'http://localhost:3000/produtos'
            }
          }
        }
        return res.status(200).send(response);
      }
    )
  });
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
  var id = req.params.id;

```

```

mysql.getConnection((error, conn) => {
  if(error){ return res.status(500).send({error: error}) }
  conn.query(
    'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
    [req.body.nome, req.body.preco, id],
    (error, result, field) => {
      conn.release();
      if(error){ return res.status(500).send({error: error}) }
      const response = {
        mensagem: "Produto atualizado com sucesso!",
        produtoAtualizado: {
          id: id,
          nome: req.body.nome,
          preco: req.body.preco,
          request: {
            tipo: 'PATCH',
            descricao: 'Retorna os detalhes de um produto específico',
            url: 'http://localhost:3000/produtos/' + id
          }
        }
      }
      return res.status(200).send(response);
    }
  );
});
});

module.exports = router;

```

PATCH ▾ localhost:3000/produtos/3

Send	200 OK	938 ms	238 B	Just Now ▾
JSON ▾ Auth ▾ Query Header 1 Docs	Preview ▾ Header 7 Cookie Timeline			
<pre> 1 + { 2 "nome": "A Múmia 2", 3 "preco": 69.90 4 } </pre>	<pre> 1 + { 2 "mensagem": "Produto atualizado com sucesso!", 3 "produtoAtualizado": { 4 "id": "3", 5 "nome": "A Múmia 2", 6 "preco": 69.9, 7 "request": { 8 "tipo": "PATCH", 9 "descricao": "Retorna os detalhes de um produto específico", 10 "url": "http://localhost:3000/produtos/3" 11 } 12 } 13 } </pre>			

localhost:3000/produtos/3

Apps Seletores de Formulário jQuery Validation Pl... j... ji

```

1 // 20210518200105
2 // http://localhost:3000/produtos/3
3
4 {
5   "produto": {
6     "id": 3,
7     "nome": "A Múmia 2",
8     "preco": 69.9,
9     "request": {
10       "tipo": "GET",
11       "descricao": "Retorna todos os produtos",
12       "url": "http://localhost:3000/produtos"
13     }
14   }
15 }

```

Excluindo um produto

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
              tipo: 'POST',
            }
          }
        }
        return res.status(201).send(response);
      }
    );
  });
});
```

```

        descricao: 'Retorna todos os produtos',
        url: 'http://localhost:3000/produtos'
    }
}
}
return res.status(201).send(response);
}
});
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'SELECT * FROM produtos WHERE id_produto = ?',
[req.params.id],
(error, result, field) => {
if(error){ return res.status(500).send({error: error}) };
if(result.length == 0){
return res.status(404).send({
mensagem: "Não foi encontrado produto com este Id!"
})
}
const response = {
produto: {
id: result[0].id_produto,
nome: result[0].nome,
preco: result[0].preco,
request: {
tipo: 'GET',
descricao: 'Retorna todos os produtos',
url: 'http://localhost:3000/produtos'
}
}
}
return res.status(200).send(response);
}
)
});
});

// Altera um produto
router.patch('/:id', (req, res, next) => {
var id = req.params.id;
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
[req.body.nome, req.body.preco, id],
(error, result, field) => {
conn.release();
if(error){ return res.status(500).send({error: error}) }
const response = {
mensagem: "Produto atualizado com sucesso!",
produtoAtualizado: {
id: id,
nome: req.body.nome,

```

```

        preco: req.body.preco,
        request: {
          tipo: 'PATCH',
          descricao: 'Retorna os detalhes de um produto específico',
          url: 'http://localhost:3000/produtos/' + id
        }
      }
    }
  }
  return res.status(200).send(response);
}
);
});

// Exclui um produto
router.delete('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto removido com sucesso!",
          request: {
            tipo: 'POST',
            descricao: 'Insere um produto',
            url: 'http://localhost:3000/produtos',
            body: {
              nome: 'String',
              preco: 'Number'
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
});

module.exports = router;

```

DELETE ▾ localhost:3000/produtos/3		Send	200 OK	652 ms	183 B	Just Now ▾		
Body ▾	Auth ▾	Query	Header	Docs	Preview ▾	Header 7	Cookie	Timeline
					<pre> 1 ✘ { 2 "mensagem": "Produto removido com sucesso!", 3 "request": { 4 "tipo": "POST", 5 "descricao": "Insere um produto", 6 "url": "http://localhost:3000/produtos", 7 "body": { 8 "nome": "String", 9 "preco": "Number" 10 } 11 } 12 }</pre>			

- Cadastre três novos produtos:

POST ▾ localhost:3000/produtos	Send	201 Created	444 ms	198 B	Just Now ▾
JSON ▾	Auth ▾	Query	Header 1	Docs	
<pre>1 + { 2 "nome": "A múmia", 3 "preco": 59.90 4 }</pre>		Preview ▾	Header 7	Cookie	Timeline
		<pre>1 + { 2 "mensagem": "Produto inserido com sucesso!", 3 "produtoCriado": { 4 "nome": "A múmia", 5 "preco": 59.9, 6 }, 7 "request": { 8 "tipo": "POST", 9 "descricao": "Retorna todos os produtos", 10 "url": "http://localhost:3000/produtos" 11 } 12 }</pre>			

POST ▾ localhost:3000/produtos	Send	201 Created	240 ms	197 B	Just Now ▾
JSON ▾	Auth ▾	Query	Header 1	Docs	
<pre>1 + { 2 "nome": "Avatar", 3 "preco": 109.90 4 }</pre>		Preview ▾	Header 7	Cookie	Timeline
		<pre>1 + { 2 "mensagem": "Produto inserido com sucesso!", 3 "produtoCriado": { 4 "nome": "Avatar", 5 "preco": 109.9, 6 }, 7 "request": { 8 "tipo": "POST", 9 "descricao": "Retorna todos os produtos", 10 "url": "http://localhost:3000/produtos" 11 } 12 }</pre>			

POST ▾ localhost:3000/produtos	Send	201 Created	189 ms	195 B	Just Now ▾
JSON ▾	Auth ▾	Query	Header 1	Docs	
<pre>1 + { 2 "nome": "Ghost", 3 "preco": 99.90 4 }</pre>		Preview ▾	Header 7	Cookie	Timeline
		<pre>1 + { 2 "mensagem": "Produto inserido com sucesso!", 3 "produtoCriado": { 4 "nome": "Ghost", 5 "preco": 99.9, 6 }, 7 "request": { 8 "tipo": "POST", 9 "descricao": "Retorna todos os produtos", 10 "url": "http://localhost:3000/produtos" 11 } 12 }</pre>			

Aula 08 - Aplicando as alterações nos Pedidos

- No MySQL Workbench:

The screenshot displays two separate sessions in MySQL Workbench. The top session shows the creation of a table named 'pedidos' with three columns: 'id_pedido' (int, primary key, auto-increment), 'quantidade' (smallint), and 'id_produto' (int). The bottom session shows the same table being populated with three rows, each containing NULL values for all columns.

Field	Type	Null	Key	Default	Extra
id_pedido	int	NO	PRI	NULL	auto_increment
quantidade	smallint	NO		NULL	
id_produto	int	NO	MUL	NULL	

id_pedido	quantidade	id_produto
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL

Cadastrando um pedido

routes\pedidos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Insere um produto
router.post('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
      [req.body.id_produto, req.body.quantidade],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido inserido com sucesso!",
          pedidoCriado: {
            id_pedido: result.insertId,
            id_produto: req.body.id_produto,
            quantidade: req.body.quantidade,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        res.json(response);
      }
    )
  })
})
```

```

        }
    }
    return res.status(201).send(response);
}
);
});
);

```

module.exports = router;

POST <localhost:3000/pedidos>

Send **201 Created** 757 ms 206 B Just Now

JSON Auth Query Header 1 Docs

Preview Header Cookie Timeline

```

1 + {
2   "mensagem": "Pedido inserido com sucesso!",
3   "pedidoCriado": {
4     "id_pedido": 1,
5     "id_produto": 1,
6     "quantidade": 2,
7     "request": {
8       "tipo": "GET",
9       "descricao": "Retorna todos os pedidos",
10      "url": "http://localhost:3000/pedidos"
11     }
12   }
13 }
```

1 ● `SELECT * FROM produtos;`
2 ● `describe pedidos;`
3 ● `select * from pedidos;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	<code>id_pedido</code>	<code>quantidade</code>	<code>id_produto</code>
▶	1	2	1
*	NULL	NULL	NULL

- Insira dois novos pedidos:

POST <localhost:3000/pedidos>

Send **201 Created** 328 ms 206 B Just Now

JSON Auth Query Header 1 Docs

Preview Header Cookie Timeline

```

1 + {
2   "mensagem": "Pedido inserido com sucesso!",
3   "pedidoCriado": {
4     "id_pedido": 2,
5     "id_produto": 4,
6     "quantidade": 1,
7     "request": {
8       "tipo": "GET",
9       "descricao": "Retorna todos os pedidos",
10      "url": "http://localhost:3000/pedidos"
11    }
12   }
13 }
```

POST <localhost:3000/pedidos>

Send **201 Created** 398 ms 206 B Just Now

JSON Auth Query Header 1 Docs

Preview Header Cookie Timeline

```

1 + {
2   "mensagem": "Pedido inserido com sucesso!",
3   "pedidoCriado": {
4     "id_pedido": 3,
5     "id_produto": 5,
6     "quantidade": 8,
7     "request": {
8       "tipo": "GET",
9       "descricao": "Retorna todos os pedidos",
10      "url": "http://localhost:3000/pedidos"
11    }
12   }
13 }
```

Listando todos os pedidos

routes\pedidos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM pedidos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          pedidos: result.map(pedido => {
            return{
              id_pedido: pedido.id_pedido,
              quantidade: pedido.quantidade,
              id_produto: pedido.id_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um pedido específico',
                url: 'http://localhost:3000/pedidos/' + pedido.id_pedido
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
}

// Insere um produto
router.post('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
      [req.body.id_produto, req.body.quantidade],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido inserido com sucesso!",
          pedidoCriado: {
            id_pedido: result.insertId,
            id_produto: req.body.id_produto,
            quantidade: req.body.quantidade,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(201).send(response);
      }
    );
  });
});
```

```
        }
      return res.status(201).send(response);
    }
  );
};

});
```

```
module.exports = router;
```

GET ▾ localhost:3000/pedidos Send **200 OK** 97.9 ms 551 B A Minute Ago ▾

Body ▾ Auth ▾ Query Header Docs Preview ▾ Header [7] Cookie Timeline

1 ✎ {
2 ✎ "response": {
3 ✎ "quantidade": 3,
4 ✎ "pedidos": [
5 ✎ {
6 ✎ "id_pedido": 1,
7 ✎ "quantidade": 2,
8 ✎ "id_produto": 1,
9 ✎ "request": {
10 ✎ "tipo": "GET",
11 ✎ "descricao": "Retorna os detalhes de um pedido específico",
12 ✎ "url": "http://localhost:3000/pedidos/1"
13 ✎ }
14 ✎ },
15 ✎ {
16 ✎ "id_pedido": 2,
17 ✎ "quantidade": 1,
18 ✎ "id_produto": 4,
19 ✎ "request": {
20 ✎ "tipo": "GET",
21 ✎ "descricao": "Retorna os detalhes de um pedido específico",
22 ✎ "url": "http://localhost:3000/pedidos/2"
23 ✎ }
24 ✎ },
25 ✎ {
26 ✎ "id_pedido": 3,
27 ✎ "quantidade": 8,
28 ✎ "id_produto": 5,
29 ✎ "request": {
30 ✎ "tipo": "GET",
31 ✎ "descricao": "Retorna os detalhes de um pedido específico",
32 ✎ "url": "http://localhost:3000/pedidos/3"
33 ✎ }
34 ✎ }
35 ✎]
36 ✎ }
37 }

Select a body type from above



Exibindo detalhes de um pedido

routes\pedidos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM pedidos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          pedidos: result.map(pedido => {
            return{
              id_pedido: pedido.id_pedido,
              quantidade: pedido.quantidade,
              id_produto: pedido.id_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um pedido específico',
                url: 'http://localhost:3000/pedidos/' + pedido.id_pedido
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
      [req.body.id_produto, req.body.quantidade],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido inserido com sucesso!",
          pedidoCriado: {
            id_pedido: result.insertId,
            id_produto: req.body.id_produto,
            quantidade: req.body.quantidade,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(201).send(response);
      }
    );
  });
});
```

```

        }
    }
}
return res.status(201).send(response);
}
});
});
});

// Retorna os dados de um pedido
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error) return res.status(500).send({error: error});
    conn.query(
      'SELECT * FROM pedidos WHERE id_pedido = ?',
      [req.params.id],
      (error, result, field) => {
        if(error) { return res.status(500).send({error: error}); }
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Não foi encontrado pedido com este Id!"
          })
        }
        const response = {
          pedido: {
            id_pedido: result[0].id_pedido,
            quantidade: result[0].quantidade,
            id_produto: result[0].id_produto,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(200).send(response);
      }
    )
  });
}

module.exports = router;

```

GET ▾ localhost:3000/pedidos/5 Send 404 Not Found 307 ms 54 B Just Now ▾

Body ▾	Auth ▾	Query	Header	Docs
Preview ▾ Header [7] Cookie Timeline				
<pre>1 ~ { 2 "mensagem": "Não foi encontrado pedido com este Id!" 3 }</pre>				

GET ▾ localhost:3000/pedidos/2 Send 200 OK 122 ms 158 B 3 Minutes Ago ▾

Body ▾	Auth ▾	Query	Header	Docs
Preview ▾ Header [7] Cookie Timeline				
<pre>1 ~ { 2 "pedido": { 3 "id_pedido": 2, 4 "quantidade": 1, 5 "id_produto": 4, 6 "request": { 7 "tipo": "GET", 8 "descricao": "Retorna todos os pedidos", 9 "url": "http://localhost:3000/pedidos" 10 } 11 } 12 }</pre>				

```
{
  "response": {
    "quantidade": 3,
    "pedidos": [
      {
        "id_pedido": 1,
        "quantidade": 2,
        "id_produto": 1,
        "request": {
          "tipo": "GET",
          "descricao": "Retorna os detalhes de um pedido específico",
          "url": "http://localhost:3000/pedidos/1"
        }
      },
      {
        "id_pedido": 2,
        "quantidade": 1,
        "id_produto": 4,
        "request": {
          "tipo": "GET",
          "descricao": "Retorna os detalhes de um pedido específico",
          "url": "http://localhost:3000/pedidos/2"
        }
      },
      {
        "id_pedido": 3,
        "quantidade": 8,
        "id_produto": 5,
        "request": {
          "tipo": "GET",
          "descricao": "Retorna os detalhes de um pedido específico",
          "url": "http://localhost:3000/pedidos/3"
        }
      }
    ]
  }
}
```

Excluindo um pedido

routes\pedidos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM pedidos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          pedidos: result.map(pedido => {
            return{
              id_pedido: pedido.id_pedido,
              quantidade: pedido.quantidade,
              id_produto: pedido.id_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um pedido específico',
                url: 'http://localhost:3000/pedidos/' + pedido.id_pedido
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
});

// Insere um produto
router.post('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
      [req.body.id_produto, req.body.quantidade],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido inserido com sucesso!",
          pedidoCriado: {
            id_pedido: result.insertId,
            id_produto: req.body.id_produto,
            quantidade: req.body.quantidade,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(201).send({response});
      }
    );
  });
});
```

```

        }
    }
    return res.status(201).send(response);
}
});
});

// Retorna os dados de um pedido
router.get('/:id', (req, res, next) => {
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'SELECT * FROM pedidos WHERE id_pedido = ?',
[req.params.id],
(error, result, field) => {
if(error){ return res.status(500).send({error: error}) };
if(result.length == 0){
return res.status(404).send({
mensagem: "Não foi encontrado pedido com este Id!"
})
}
const response = {
pedido: {
id_pedido: result[0].id_pedido,
quantidade: result[0].quantidade,
id_produto: result[0].id_produto,
request: {
tipo: 'GET',
descricao: 'Retorna todos os pedidos',
url: 'http://localhost:3000/pedidos'
}
}
}
return res.status(200).send(response);
}
)
});
});

// Exclui um pedido
router.delete('/:id', (req, res, next) => {
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'DELETE FROM pedidos WHERE id_pedido = ?',
[req.params.id],
(error, result, field) => {
conn.release();
if(error){ return res.status(500).send({error: error}) }
const response = {
mensagem: "Pedido removido com sucesso!",
request: {
tipo: 'POST',
descricao: 'Insere um pedido',
url: 'http://localhost:3000/pedidos',
body: {
id_produto: 'Number',
quantidade: 'Number'
}
}
}
return res.status(200).send(response);
}
)
});
});
});
});
```

```

        }
    }
    return res.status(200).send(response);
}
});
});

module.exports = router;

```

DELETE ▾ localhost:3000/pedidos/3

Send **200 OK** 605 ms 191 B Just Now ▾

Body ▾ Auth ▾ Query Header Docs

Preview ▾ Header [7] Cookie Timeline

```

1 v {
2   "mensagem": "Pedido removido com sucesso!",
3   "request": [
4     "tipo": "POST",
5     "descricao": "Insere um pedido",
6     "url": "http://localhost:3000/pedidos",
7     "body": [
8       "id_produto": "Number",
9       "quantidade": "Number"
10    ]
11  }
12 }

```

GET ▾ localhost:3000/pedidos

Send **200 OK** 6.12 ms 381 B Just Now ▾

Body ▾ Auth ▾ Query Header Docs

Preview ▾ Header [7] Cookie Timeline



Select a body type from above

```

1 v {
2   "response": [
3     {
4       "quantidade": 2,
5       "pedidos": [
6         {
7           "id_pedido": 1,
8           "quantidade": 2,
9           "id_produto": 1,
10          "request": [
11            "tipo": "GET",
12            "descricao": "Retorna os detalhes de um pedido específico",
13            "url": "http://localhost:3000/pedidos/1"
14          ],
15        },
16        {
17           "id_pedido": 2,
18           "quantidade": 1,
19           "id_produto": 4,
20           "request": [
21             "tipo": "GET",
22             "descricao": "Retorna os detalhes de um pedido específico",
23             "url": "http://localhost:3000/pedidos/2"
24           ],
25        }
26      ]
27    }

```

routes\pedidos.js

```

const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM pedidos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          pedidos: result.map(pedido => {
            return{
              id_pedido: pedido.id_pedido,
              quantidade: pedido.quantidade,

```

```

        id_produto: pedido.id_produto,
        request: {
          tipo: 'GET',
          descricao: 'Retorna os detalhes de um pedido específico',
          url: 'http://localhost:3000/pedidos/' + pedido.id_pedido
        }
      }
    }
  }
  return res.status(200).send({response});
}
)
);
}

// Insere um produto
router.post('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.body.id_produto],
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) }
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Produto não encontrado!"
          })
        }
      }
    )
    conn.query(
      'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
      [req.body.id_produto, req.body.quantidade],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido inserido com sucesso!",
          pedidoCriado: {
            id_pedido: result.insertId,
            id_produto: req.body.id_produto,
            quantidade: req.body.quantidade,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(201).send(response);
      }
    )
  });
});

// Retorna os dados de um pedido
router.get('/:id', (req, res, next) => {

```

```

mysql.getConnection((error, conn) => {
  if(error){ return res.status(500).send({error: error}) }
  conn.query(
    'SELECT * FROM pedidos WHERE id_pedido = ?',
    [req.params.id],
    (error, result, field) => {
      if(error){ return res.status(500).send({error: error}) };
      if(result.length == 0){
        return res.status(404).send({
          mensagem: "Não foi encontrado pedido com este Id!"
        })
      }
      const response = {
        pedido: {
          id_pedido: result[0].id_pedido,
          quantidade: result[0].quantidade,
          id_produto: result[0].id_produto,
          request: {
            tipo: 'GET',
            descricao: 'Retorna todos os pedidos',
            url: 'http://localhost:3000/pedidos'
          }
        }
      }
      return res.status(200).send(response);
    }
  )
});

// Exclui um pedido
router.delete('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM pedidos WHERE id_pedido = ?',
      [req.params.id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido removido com sucesso!",
          request: {
            tipo: 'POST',
            descricao: 'Insere um pedido',
            url: 'http://localhost:3000/pedidos',
            body: {
              id_produto: 'Number',
              quantidade: 'Number'
            }
          }
        }
        return res.status(200).send(response);
      }
    )
  });
});

module.exports = router;

```

POST [localhost:3000/pedidos](#)

Send **404 Not Found** 522 ms 39 B Just Now ▾

JSON ▾ Auth ▾ Query Header 1 Docs

```
1: {  
2:   "id_produto": 2,  
3:   "quantidade": 8  
4: }
```

Preview ▾ Header 7 Cookie Timeline

```
1: {  
2:   "mensagem": "Produto não encontrado!"  
3: }
```

Aula 09 - Consultando várias tabelas (INNER JOIN)

The screenshot shows the MySQL Workbench interface. In the top query editor window, three statements are listed:

```
1 •  SELECT * FROM produtos;
2 •  describe pedidos;
3 •  select * from pedidos;
```

The third statement, "select * from pedidos;", is highlighted. Below it, the result grid displays the following data:

	id_pedido	quantidade	id_produto
▶	1	2	1
	2	1	4
*	NULL	NULL	NULL

The screenshot shows the MySQL Workbench interface again. In the top query editor window, the following query is listed:

```
4 •  select pedidos.id_pedido,
5          pedidos.quantidade,
6          produtos.id_produto,
7          produtos.nome,
8          produtos.preco
9      from pedidos
10     inner join produtos
11       on produtos.id_produto = pedidos.id_produto;
```

The result grid displays the following data, combining information from both tables:

	id_pedido	quantidade	id_produto	nome	preco
▶	1	2	1	Harry Potter	99.6
	2	1	4	A m\u00famia	59.9

routes\pedidos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;

// Retorna todos os pedidos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      `select pedidos.id_pedido,
             pedidos.quantidade,
             produtos.id_produto,
             produtos.nome,
             produtos.preco
      from pedidos
      inner join produtos
      on produtos.id_produto = pedidos.id_produto;`,
```

```

(error, result, field) => {
  if(error){ return res.status(500).send({error: error}) };
  const response = {
    pedidos: result.map(pedido => {
      return{
        id_pedido: pedido.id_pedido,
        quantidade: pedido.quantidade,
        produto: {
          id_produto: pedido.id_produto,
          nome: pedido.nome,
          preco: pedido.preco
        },
        request: {
          tipo: 'GET',
          descricao: 'Retorna os detalhes de um pedido específico',
          url: 'http://localhost:3000/pedidos/' + pedido.id_pedido
        }
      }
    })
  }
  return res.status(200).send({response});
}
)
});

// Insere um produto
router.post('/', (req, res, next) => {

  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.body.id_produto],
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) }
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Produto não encontrado!"
          })
        }
      }
    )

    conn.query(
      'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
      [req.body.id_produto, req.body.quantidade],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido inserido com sucesso!",
          pedidoCriado: {
            id_pedido: result.insertId,
            id_produto: req.body.id_produto,
            quantidade: req.body.quantidade,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(201).send(response);
      }
    )
  })
})
);

```

```

        }
    }
    return res.status(201).send(response);
}
)
}

});

// Retorna os dados de um pedido
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM pedidos WHERE id_pedido = ?',
      [req.params.id],
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Não foi encontrado pedido com este Id!"
          })
        }
        const response = {
          pedido: {
            id_pedido: result[0].id_pedido,
            quantidade: result[0].quantidade,
            id_produto: result[0].id_produto,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os pedidos',
              url: 'http://localhost:3000/pedidos'
            }
          }
        }
        return res.status(200).send(response);
      }
    )
  })
});

// Exclui um pedido
router.delete('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM pedidos WHERE id_pedido = ?',
      [req.params.id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido removido com sucesso!",
          request: {
            tipo: 'POST',
            descricao: 'Insere um pedido',
            url: 'http://localhost:3000/pedidos',
          }
        }
        return res.status(200).send(response);
      }
    )
  })
});

```

```

        body: {
          id_produto: 'Number',
          quantidade: 'Number'
        }
      }
    }
  return res.status(200).send(response);
}
)
});
};

module.exports = router;

```

GET ▾ localhost:3000/pedidos

Send

Body ▾	Auth ▾	Query	Header	Docs	200 OK	259 ms	456 B	3 Minutes Ago ▾
					Preview ▾	Header	Cookie	Timeline
					<pre> 1 v { "response": { 2 v "pedidos": [3 v { 4 v "id_pedido": 1, 5 v "quantidade": 2, 6 v "produto": { 7 v "id_produto": 1, 8 v "name": "Harry Potter", 9 v "preco": 99.6 10 v }, 11 v "request": { 12 v "tipo": "GET", 13 v "descricao": "Retorna os detalhes de um pedido específico", 14 v "url": "http://localhost:3000/pedidos/1" 15 v } 16 v }, 17 v { 18 v "id_pedido": 2, 19 v "quantidade": 1, 20 v "produto": { 21 v "id_produto": 4, 22 v "name": "A múmia", 23 v "preco": 59.9 24 v }, 25 v "request": { 26 v "tipo": "GET", 27 v "descricao": "Retorna os detalhes de um pedido específico", 28 v "url": "http://localhost:3000/pedidos/2" 29 v } 30 v } 31 v] 32 v } 33 v } 34 v } </pre>			

Select a body type from above



Aula 10 - Upload de Imagens

Instalando o multer

```
npm install --save multer
```

```
C:\maransatto\rest>npm install --save multer
npm WARN rest-api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ multer@1.4.2
added 17 packages from 12 contributors and audited 212 packages in 49.62s

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

found 0 vulnerabilities
```

- Essa biblioteca é uma alternativa para tratar o formdata.

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');
const upload = multer({dest: 'uploads/'});

// Insere um produto
router.post('/', upload.single('produto_imagem'), (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
              tipo: 'POST',
              descricao: 'Retorna todos os produtos',
              url: 'http://localhost:3000/produtos'
            }
          }
        }
      }
    )
  })
})
```

```

        return res.status(201).send(response);
    }
}
});
});

module.exports = router;

```

- No Insomnia:

POST localhost:3000/produtos		Send	201 Created	2.81 s	204 B	4 Minutes Ago
Multipart	3	Auth	Query	Header	Docs	
name	Ventilador					
preco	100.55					
produto_imagem	<input type="file" value="ventilador.jpg"/>					
New name	New value					
<pre> 1 { 2 "mensagem": "Produto inserido com sucesso!", 3 "produtoCriado": { 4 "nome": "Ventilador", 5 "preco": "100.55", 6 "request": { 7 "tipo": "POST", 8 "descricao": "Retorna todos os produtos", 9 "url": "http://localhost:3000/produtos" 10 } 11 } 12 }</pre>						

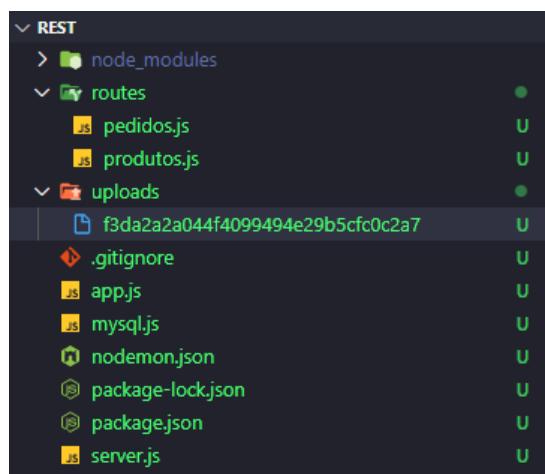
```

C:\maransatto\rest>npm start

> rest-api@1.0.0 start C:\maransatto\rest
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Servidor rodando na porta 3000
{
 fieldname: 'produto_imagem',
originalname: 'ventilador.jpg',
encoding: '7bit',
mimetype: 'image/jpeg',
destination: 'uploads/',
filename: 'f3da2a2a044f4099494e29b5fc0c2a7',
path: 'uploads\\f3da2a2a044f4099494e29b5fc0c2a7',
size: 13907
}
POST /produtos 201 2320.816 ms - 204

```



- Apague este arquivo na página **uploads**.

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

const upload = multer({ storage: storage });

// Insere um produto
router.post('/', upload.single('produto_imagem'), (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
              tipo: 'POST',
              descricao: 'Retorna todos os produtos',
              url: 'http://localhost:3000/produtos'
            }
          }
        }
        return res.status(201).send(response);
      }
    );
  });
}

module.exports = router;
```

- No Insomnia:

POST localhost:3000/produtos

Multipart 3 Send 201 Created 2.16 s 204 B 3 Minutes Ago

Auth Query Header 1 Docs Preview Header 7 Cookie Timeline

name: Ventilador
preco: 100.55
produto_imagem: ventilador.jpg
New name: New value

```
1: {
2:   "mensagem": "Produto inserido com sucesso!",
3:   "produtoCriado": {
4:     "nome": "Ventilador",
5:     "preco": "100.55",
6:     "request": {
7:       "tipo": "POST",
8:       "descricao": "Retorna todos os produtos",
9:       "url": "http://localhost:3000/produtos"
10:    }
11:  }
12: }
```

REST

node_modules routes uploads

- pedidos.js
- produtos.js
- 2021-05-19T10-59-26.539Z-ventilador.jpg
- .gitignore
- app.js
- mysql.js
- nodemon.json
- package-lock.json
- package.json
- server.js



Estipulando limites e filtrando tipos de arquivos

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-').replace(/-/g, '-');
    cb(null, data + file.originalname );
  }
});

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

// Insere um produto
router.post('/', upload.single('produto_imagem'), (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco) VALUES (?,?)',
      [req.body.nome, req.body.preco],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          nome: result[0].nome,
          preco: result[0].preco
        }
        res.json(response);
      }
    );
  });
});
```

```

        mensagem: "Produto inserido com sucesso!",
        produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
                tipo: 'POST',
                descricao: 'Retorna todos os produtos',
                url: 'http://localhost:3000/produtos'
            }
        }
    }
    return res.status(201).send(response);
}
});

module.exports = router;

```

- No Insomnia:

POST localhost:3000/produtos		Send	201 Created	1.04 s	204 B	3 Minutes Ago
Multipart	3	Auth	Query	Header	Docs	
<input type="text" value="nome"/> Ventilador <input type="text" value="preco"/> 100.55 <input type="text" value="produto_imagem"/> Curso_NodeJS-Celke.pdf <input type="text" value="New name"/> New value						Preview
<pre> 1: { 2: "mensagem": "Produto inserido com sucesso!", 3: "produtoCriado": { 4: "nome": "Ventilador", 5: "preco": "100.55", 6: "request": { 7: "tipo": "POST", 8: "descricao": "Retorna todos os produtos", 9: "url": "http://localhost:3000/produtos" 10: } 11: } 12: }</pre>						Header
						Cookie
						Timeline

```

Servidor rodando na porta 3000
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Servidor rodando na porta 3000
undefined
POST /produtos 201 965.028 ms - 204

```

Acrescentando coluna para imagem na tabela produtos

- No MySQL Workbench:

```
describe produtos;
```

	Field	Type	Null	Key	Default	Extra
▶	id_produto	int	NO	PRI	NULL	auto_increment
	nome	varchar(220)	NO		NULL	
	preco	float	NO		NULL	

```
alter table produtos add column imagem_produto varchar(500);
```

```
12 08:33:53 alter table produtos add column imagem_produto varchar(500) 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 5.516 sec
```

```
describe produtos;
```

	Field	Type	Null	Key	Default	Extra
▶	id_produto	int	NO	PRI	NULL	auto_increment
	nome	varchar(220)	NO		NULL	
	preco	float	NO		NULL	
	imagem_produto	varchar(500)	YES		NULL	

- No Insomnia:

POST	localhost:3000/produtos	Send	201 Created	852 ms	204 B	Just Now		
Multipart	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
3								
name	Ventilador							
preco	100.55							
produto_imagem	ventilador.jpg							
New name	New value							

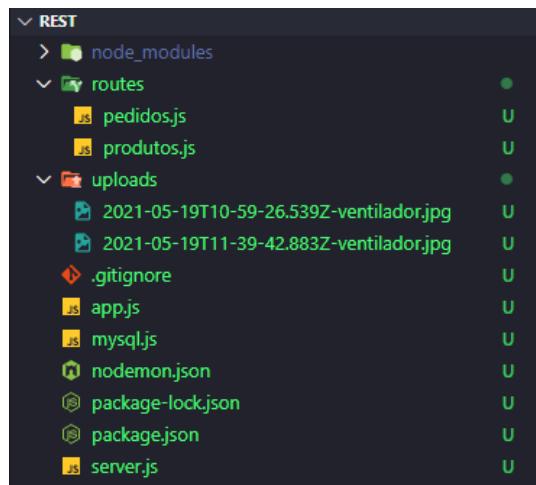
```
1 { "mensagem": "Produto inserido com sucesso!",  
2   "produtoCriado": {  
3     "nome": "Ventilador",  
4     "preco": "100.55",  
5     "request": {  
6       "tipo": "POST",  
7       "descricao": "Retorna todos os produtos",  
8       "url": "http://localhost:3000/produtos"  
9     }  
10   }  
11 }  
12 }
```

- No MySQL Workbench:

```
select * from produtos;
```

The screenshot shows the MySQL Workbench interface with a query editor containing the command "select * from produtos;". Below the editor is a result grid displaying the following data:

	id_produto	nome	preco	imagem_produto
▶	1	Harry Potter	99.6	NULL
	4	A múmia	59.9	NULL
	5	Avatar	109.9	NULL
	6	Ghost	99.9	NULL
	7	Ventilador	100.55	NULL
	8	Ventilador	100.55	NULL
	9	Ventilador	100.55	NULL
	10	Ventilador	100.55	uploads\2021-05-19T10-59-26.539Z-ventilador.jpg
*	NULL	NULL	NULL	NULL



routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});
```

```

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

// Insere um produto
router.post('/', upload.single('produto_imagem'), (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
      [req.body.nome, req.body.preco, req.file.path],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
            nome: req.body.nome,
            preco: req.body.preco,
            imagem_produto: req.file.path,
            request: {
              tipo: 'POST',
              descricao: 'Retorna todos os produtos',
              url: 'http://localhost:3000/produtos'
            }
          }
        }
        return res.status(201).send(response);
      }
    );
  });
}

module.exports = router;

```

- No Insomnia:

POST localhost:3000/produtos Send 201 Created 495 ms 272 B Just Now

Multipart 3 Auth Query Header 1 Docs

Preview Header 7 Cookie Timeline

nome: Ventilador
preco: 100.55
produto_imagem: ventilador.jpg
New name: New value

```
1 "mensagem": "Produto inserido com sucesso!",  
2 "produtoCriado": {  
3 "nome": "Ventilador",  
4 "preco": "100.55",  
5 "imagem_produto": "uploads\\2021-05-19T11-48-34.911Z-ventilador.jpg",  
6 "request": {  
7 "tipo": "POST",  
8 "descricao": "Retorna todos os produtos",  
9 "url": "http://localhost:3000/produtos"  
10 }  
11 }  
12 }  
13 }
```

Tornando a imagem pública

app.js

```
const express = require('express');
const app = express();
const morgan = require('morgan');
const bodyParser = require('body-parser');

const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');

app.use(morgan('dev'));
app.use('/uploads', express.static('uploads'));
app.use(bodyParser.urlencoded({extended:false})); // apenas dados simples
app.use(bodyParser.json()); // json de entrada no body

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);

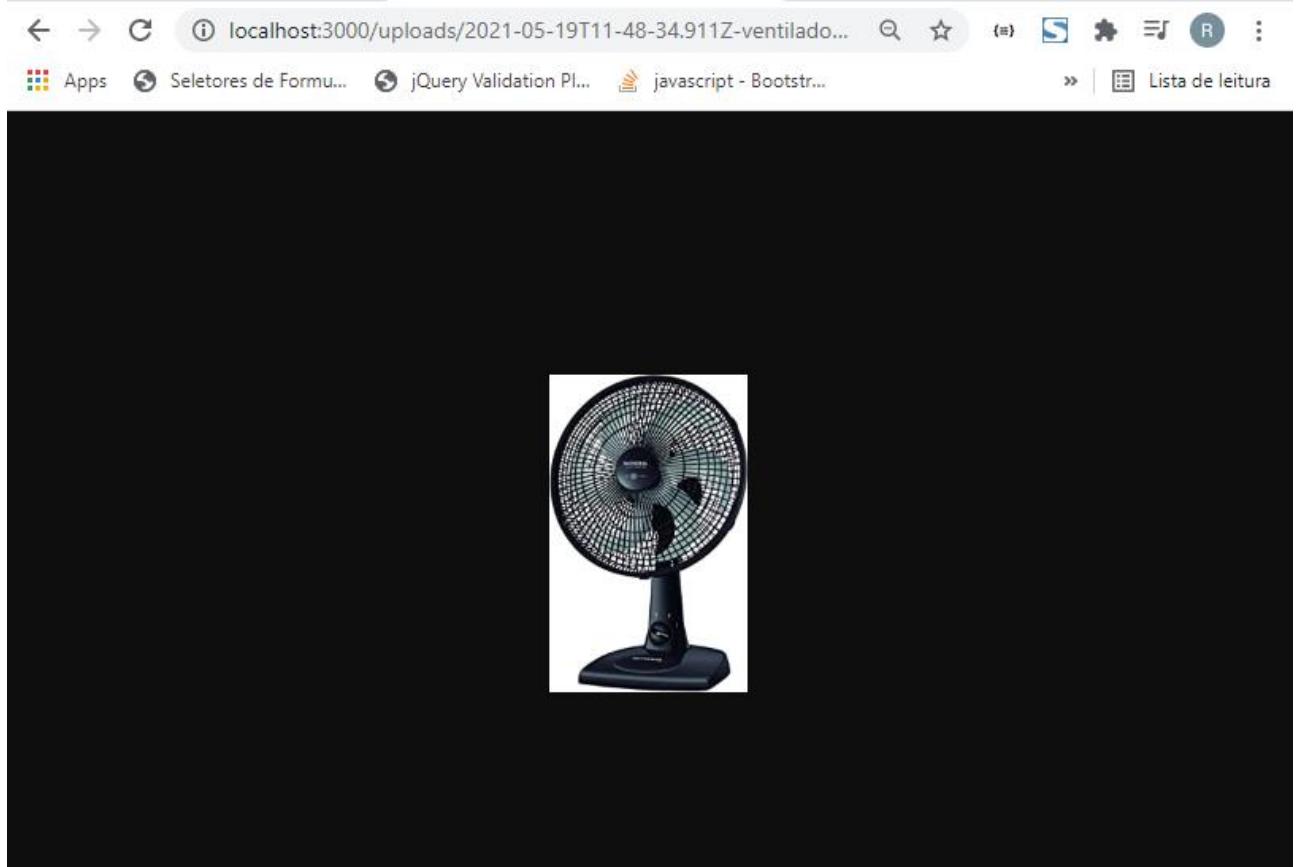
// Tratamento quando não for encontrada a rota
app.use((req, res, next) => {
  const erro = new Error('Rota não encontrada!');
  erro.status = 404;
  next(erro);
});

app.use((error, req, res, next) => {
  res.status(error.status) || 500;
  return res.send({
    erro: {
      mensagem: error.message
    }
  });
});

module.exports = app;
```

- No browser:

<http://localhost:3000/uploads/2021-05-19T11-48-34.911Z-ventilador.jpg>



routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}
```

```

        }
    }

const upload = multer({
    storage: storage,
    limits: {
        fileSize: 1024 * 1024 * 5
    },
    fileFilter: fileFilter
});

// Retorna todos os produtos
router.get('/', (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'SELECT * FROM produtos',
            (error, result, field) => {
                if(error){ return res.status(500).send({error: error}) };
                const response = {
                    quantidade: result.length,
                    produtos: result.map(prod => {
                        return{
                            id: prod.id_produto,
                            nome: prod.nome,
                            preco: prod.preco,
                            imagem_produto: prod.imagem_produto,
                            request: {
                                tipo: 'GET',
                                descricao: 'Retorna os detalhes de um produto específico',
                                url: 'http://localhost:3000/produtos/' + prod.id_produto
                            }
                        }
                    })
                }
                return res.status(200).send({response});
            }
        )
    });
});

// Insere um produto
router.post('/', upload.single('produto_imagem'), (req, res, next) => {
    console.log(req.file);
    const produto = {
        nome: req.body.nome,
        preco: req.body.preco
    };
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
            [req.body.nome, req.body.preco, req.file.path],
            (error, result, field) => {

```

```

conn.release();
if(error){ return res.status(500).send({error: error}) }
const response = {
  mensagem: "Produto inserido com sucesso!",
  produtoCriado: {
    id: result.id_produto,
    nome: req.body.nome,
    preco: req.body.preco,
    imagem_produto: req.file.path,
    request: {
      tipo: 'POST',
      descricao: 'Retorna todos os produtos',
      url: 'http://localhost:3000/produtos'
    }
  }
}
return res.status(201).send(response);
}

});

module.exports = router;

```

- No Insomnia:

GET ▾ localhost:3000/produtos		Send	200 OK	514 ms	1850 B	A Minute Ago ▾
Body ▾	Auth ▾ Query Header Docs		Preview ▾	Header 7	Cookie	Timeline
			<pre> 70 }, 71 [72 { 73 "id": 9, 74 "nome": "Ventilador", 75 "preco": 100.55, 76 "imagem_produto": null, 77 "request": { 78 "tipo": "GET", 79 "descricao": "Retorna os detalhes de um produto específico", 80 "url": "http://localhost:3000/produtos/9" 81 }, 82 [83 { 84 "id": 18, 85 "nome": "Ventilador", 86 "preco": 100.55, 87 "imagem_produto": "uploads\\2021-05-19T11-39-42.883Z-ventilador.jpg", 88 "request": { 89 "tipo": "GET", 90 "descricao": "Retorna os detalhes de um produto específico", 91 "url": "http://localhost:3000/produtos/18" 92 }, 93 [94 { 95 "id": 11, 96 "nome": "Ventilador", 97 "preco": 100.55, 98 "imagem_produto": "uploads\\2021-05-19T11-48-34.911Z-ventilador.jpg", 99 "request": { 100 "tipo": "GET", 101 "descricao": "Retorna os detalhes de um produto específico", 102 "url": "http://localhost:3000/produtos/11" 103 } 104] 105 } 106 } </pre>			
				Select a body type from above		

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              imagem_produto: prod.imagem_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
      }
    )
  });
});
```

```

        return res.status(200).send({response});
    }
}
});

// Insere um produto
router.post('/', upload.single('produto_imagem'), (req, res, next) => {
    console.log(req.file);
    const produto = {
        nome: req.body.nome,
        preco: req.body.preco
    };
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
            [req.body.nome, req.body.preco, req.file.path],
            (error, result, field) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                const response = {
                    mensagem: "Produto inserido com sucesso!",
                    produtoCriado: {
                        id: result.id_produto,
                        nome: req.body.nome,
                        preco: req.body.preco,
                        imagem_produto: req.file.path,
                        request: {
                            tipo: 'POST',
                            descricao: 'Retorna todos os produtos',
                            url: 'http://localhost:3000/produtos'
                        }
                    }
                }
                return res.status(201).send(response);
            }
        )
    });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'SELECT * FROM produtos WHERE id_produto = ?',
            [req.params.id],
            (error, result, field) => {
                if(error){ return res.status(500).send({error: error}) };
                if(result.length == 0){
                    return res.status(404).send({
                        mensagem: "Não foi encontrado produto com este Id!"
                    })
                }
                const response = {
                    produto: {
                        id: result[0].id_produto,
                        nome: result[0].nome,

```

```

        preco: result[0].preco,
        imagem_produto: result[0].imagem_produto,
        request: {
            tipo: 'GET',
            descricao: 'Retorna todos os produtos',
            url: 'http://localhost:3000/produtos'
        }
    }
}
return res.status(200).send(response);
}

module.exports = router;

```

- No Insomnia:

GET ▾ localhost:3000/produtos/11		Send	200 OK	346 ms	228 B	Just Now ▾
Body ▾	Auth ▾ Query Header Docs		Preview ▾	Header ▾	Cookie	Timeline
<pre> 1 { 2 "produto": { 3 "id": 11, 4 "nome": "Ventilador", 5 "preco": 100.55, 6 "imagem_produto": "uploads\\2021-05-19T11-48-34.911Z-ventilador.jpg", 7 "request": { 8 "tipo": "GET", 9 "descricao": "Retorna todos os produtos", 10 "url": "http://localhost:3000/produtos" 11 } 12 } 13 }</pre>						

```

1 // 20210519091203
2 // http://localhost:3000/produtos/11
3
4 +
5     "produto": {
6         "id": 11,
7         "nome": "Ventilador",
8         "preco": 100.55,
9         "imagem_produto": "uploads\\2021-05-19T11-48-34.911Z-ventilador.jpg",
10        "request": {
11            "tipo": "GET",
12            "descricao": "Retorna todos os produtos",
13            "url": "http://localhost:3000/produtos"
14        }
15    }
16 }
```

Aula 11 - Cadastro de Usuário (com senha hash)

Autenticação na API - Como funciona



- A comunicação entre cliente e servidor não existe em estado de sessão.
- O token é basicamente uma chave que autoriza tanto do lado do cliente como do lado do servidor.

O que é um token?



O token normalmente é composto por um JSON e nesse JSON incluimos algumas informações como por exemplo: o email e o id do usuário e uma assinatura. Essa assinatura pode ser validada, tanto no cliente como no servidor. Isso é um conceito de chave pública e privada. Quando se envia um token há uma chave que existe somente do lado do servidor e essa chave privada é validada junto com o token que é retornado para o cliente. Se algum usuário malicioso quiser alterar esse token ele não tem como ser verificado do lado do servidor, então não há como adulterar ele, embora ele não seja necessariamente um objeto criptografado e conseguimos inclusive ver algumas informações contidas nele.

Criando a estrutura base no BD

Criando a tabela de usuários

```
Create table usuarios (
    id_usuario integer not null primary key auto_increment,
    email varchar(220),
    senha varchar(100)
)
```

```
16 • ⚡ Create table usuarios (
17     id_usuario integer not null primary key auto_increment,
18     email varchar(220),
19     senha varchar(100)
20 );
21
```

Output

#	Time	Action	Message
13	08:35:04	describe produtos	4 row(s) returned
14	08:41:16	select * from produtos LIMIT 0, 1000	8 row(s) returned
15	15:48:49	Create table usuarios (id_usuario integer not null primary key auto_increment, email varchar(...)	0 row(s) affected

```
show tables;
```

```
22 • show tables;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	Tables_in_ecommerce
▶	pedidos
▶	produtos
▶	usuarios

- Instale o pacote bcrypt:

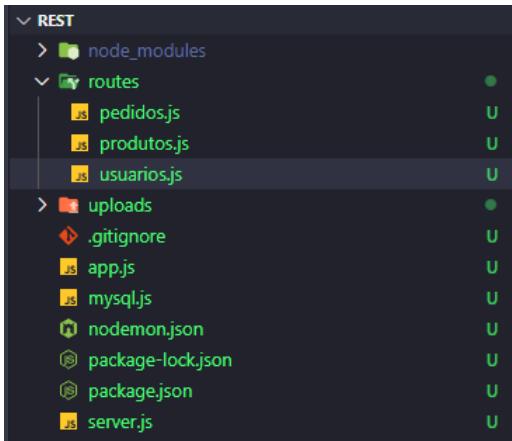
```
npm install --save bcrypt
```

```
C:\maransatto\rest>npm install --save bcrypt
> bcrypt@5.0.1 install C:\maransatto\rest\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build
[bcrypt] Success: "C:\maransatto\rest\node_modules\bcrypt\lib\binding\napi-v3\brypt_lib.node" is installed via remote
npm WARN rest-api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ bcrypt@5.0.1
added 42 packages from 109 contributors and audited 254 packages in 52.811s

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

found 0 vulnerabilities
```

- Crie a rota de usuários



Cadastrando usuários

routes\usuarios.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const bcrypt = require('bcrypt');

router.post('/cadastro', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    bcrypt.hash(req.body.senha, 10, (errBcrypt, hash) => {
      if(errBcrypt){ return res.status(500).send({error: errBcrypt}) }
      conn.query(
        'insert into usuarios(email, senha) values (?,?)',
        [req.body.email, hash],
        (error, results) => {
          conn.release();
          if(error){ return res.status(500).send({error: error}) }
          response = {
            mensagem: 'Usuário cadastrado com sucesso!',
            usuarioCriado: {
              id_usuario: results.insertId,
              email: req.body.email
            }
          }
          return res.status(201).send(response);
        }
      );
    });
  });
});

module.exports = router;
```

app.js

```
const express = require('express');
const app = express();
const morgan = require('morgan');
const bodyParser = require('body-parser');

const rotaProdutos = require('./routes/produtos');
const rotaPedidos = require('./routes/pedidos');
const rotaUsuarios = require('./routes/usuarios');

app.use(morgan('dev'));
app.use('/uploads', express.static('uploads'));
app.use(bodyParser.urlencoded({extended:false})); // apenas dados simples
app.use(bodyParser.json()); // json de entrada no body

app.use('/produtos', rotaProdutos);
app.use('/pedidos', rotaPedidos);
app.use('/usuarios', rotaUsuarios);

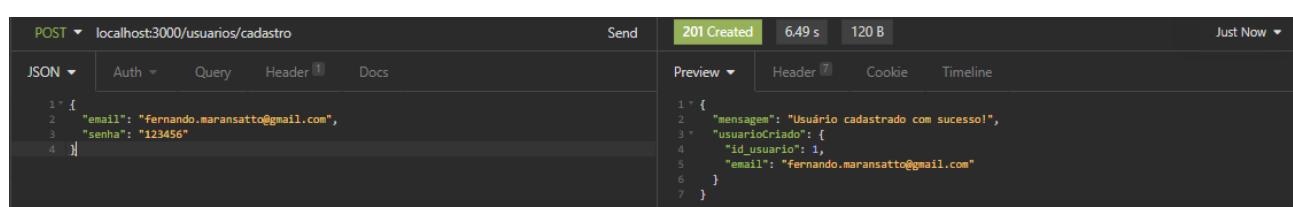
// Tratamento quando não for encontrada a rota
app.use((req, res, next) => {
  const erro = new Error('Rota não encontrada!');
  erro.status = 404;
  next(erro);
});

app.use((error, req, res, next) => {
  res.status(error.status) || 500;
  return res.send({
    erro: {
      mensagem: error.message
    }
  });
});

module.exports = app;
```

- No Insomnia:

POST localhost:3000/usuarios/cadastro



The screenshot shows the Insomnia REST client interface. The top bar indicates a POST request to 'localhost:3000/usuarios/cadastro'. The status bar shows '201 Created' with a duration of '6.49 s' and a size of '120 B', timestamped 'Just Now'. Below the status bar, there are tabs for 'Preview', 'Header', 'Cookie', and 'Timeline'. The 'Preview' tab displays a JSON response message: "Usuário cadastrado com sucesso!". The 'Header' tab shows the following headers: Content-Type: application/json, Accept: application/json, Host: localhost:3000, Connection: keep-alive, and X-Powered-By: Express. The 'Cookie' and 'Timeline' tabs are empty.

- No MySQL Workbench:

```
select * from usuarios;
```

```
24 • select * from usuarios;
```

Result Grid		
	id_usuario	email
▶	1	fernando.maransatto@gmail.com
*	NULL	NULL

senha

\$2b\$10\$6fWY0xkeXuuANQDcthGEU.lkJc1wkGBjTq4jPo/nMJIffunJDar.
NULL

- O email deve ser único:

```
alter table usuarios add unique(email);
```

```
25 • alter table usuarios add unique(email);
```

Action Output		
#	Time	Action
16	15:51:55	show tables
17	16:46:01	select * from usuarios LIMIT 0, 1000
18	16:50:56	alter table usuarios add unique(email)

Message Duration / Fetch

3 row(s) returned 1.657 sec / 0.063 sec

1 row(s) returned 0.094 sec / 0.000 sec

0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 3.625 sec

```
describe usuarios;
```

```
25 • describe usuarios;
```

	Field	Type	Null	Key	Default	Extra
▶	id_usuario	int	NO	PRI	NULL	auto_increment
	email	varchar(220)	YES	UNI	NULL	
	senha	varchar(100)	YES		NULL	

- E não pode ser nulo:

```
alter table usuarios modify column email varchar(220) not null;
```

```
describe usuarios;
```

```
27 • describe usuarios;
```

	Field	Type	Null	Key	Default	Extra
▶	id_usuario	int	NO	PRI	NULL	auto_increment
	email	varchar(220)	NO	UNI	NULL	
	senha	varchar(100)	YES		NULL	

- A senha também não pode ser nula:

```
alter table usuarios modify column senha varchar(100) not null;
```

```
describe usuarios;
```

```
29 • describe usuarios;
```

The screenshot shows the 'Result Grid' tab in MySQL Workbench. The table structure for 'usuarios' is displayed with the following columns and data:

	Field	Type	Null	Key	Default	Extra
▶	id_usuario	int	NO	PRI	NULL	auto_increment
	email	varchar(220)	NO	UNI	NULL	
	senha	varchar(100)	NO		NULL	

- No Insomnia, tentando cadastrar usuario com o mesmo email:

The screenshot shows a POST request to `localhost:3000/usuarios/cadastro`. The JSON payload is:

```
1 * {
2 *   "email": "fernando.maransatto@gmail.com",
3 *   "senha": "123456"
4 }
```

The response status is **500 Internal Server Error** with a response time of **595 ms** and a body size of **153 B**. The response header includes `Content-Type: application/json` and `Content-Length: 153`. The response body is:

```
1 * {
2 *   "error": {
3 *     "code": "ER_DUP_ENTRY",
4 *     "errno": 1062,
5 *     "sqlState": "23000",
6 *     "sqlMessage": "Duplicate entry 'fernando.maransatto@gmail.com' for key 'usuarios.email'"
7 *   }
8 }
```

routes\usuarios.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const bcrypt = require('bcrypt');

router.post('/cadastro', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }

    conn.query('select * from usuarios where email = ?', [req.body.email], (error, results) => {
      if(error){ return res.status(500).send({error: error}) }
      if(results.length > 0){
        return res.status(409).send({
          mensagem: "Usuário já cadastrado!"
        });
      } else {
        bcrypt.hash(req.body.senha, 10, (errBcrypt, hash) => {
          if(errBcrypt){ return res.status(500).send({error: errBcrypt}) }
          conn.query(
            'insert into usuarios(email, senha) values (?,?)',
            [req.body.email, hash],
            (error, results) => {
              conn.release();
              if(error){ return res.status(500).send({error: error}) }
              response = {
                mensagem: 'Usuário cadastrado com sucesso!',
                usuarioCriado: {
                  id_usuario: results.insertId,
                  email: req.body.email
                }
              }
              return res.status(201).send(response);
            }
          );
        });
      }
    });
  });
});

module.exports = router;
```

- No Insomnia, ao tentar cadastrar usuário com o mesmo email:

The screenshot shows the Insomnia REST Client interface. A POST request is being made to the endpoint `localhost:3000/usuarios/cadastro`. The request body is a JSON object with two fields: `email` and `senha`. The response is a `409 Conflict` status with a single key-value pair: `mensagem: "Usuário já cadastrado!"`.

POST <code>localhost:3000/usuarios/cadastro</code>		Send	409 Conflict	826 ms	39 B	Just Now		
JSON	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
{ "email": "fernando.maransatto@gmail.com", "senha": "123456" }						1 " { 2 "mensagem": "Usuário já cadastrado!" 3 }		

POST localhost:3000/usuarios/cadastro Send 201 Created 488 ms 118 B Just Now

JSON Auth Query Header Docs Preview Header 7 Cookie Timeline

```
1+ {  
2   "email": "miguel.maransatto@gmail.com",  
3   "senha": "123456"  
4 }
```

```
1+ {  
2   "mensagem": "Usuário cadastrado com sucesso!",  
3   "usuarioCriado": {  
4     "id_usuario": 3,  
5     "email": "miguel.maransatto@gmail.com"  
6   }  
7 }
```

- No MySQL Workbench:

```
select * from usuarios;
```

30 • select * from usuarios;

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	id_usuario	email	senha
▶	1	fernando.maransatto@gmail.com	\$2b\$10\$6fWY0xkeXuuANQDcthGEU.lkJc1wkGBjTq4jPo/nM]lfjfunJDar.
*	3	miguel.maransatto@gmail.com	\$2b\$10\$OIjtC/t3175Ad4SGa4WoBurJAi37bCVvucE6m5cad8vl4imyW.sny
*	NULL	NULL	NULL

Aula 12 - Login + JWT Token

routes\usuarios.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const bcrypt = require('bcrypt');

router.post('/cadastro', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }

    conn.query('select * from usuarios where email = ?',
    [req.body.email],
    (error, results) => {
      if(error){ return res.status(500).send({error: error}) }
      if(results.length > 0){
        return res.status(409).send({
          mensagem: "Usuário já cadastrado!"
        });
      } else {
        bcrypt.hash(req.body.senha, 10, (errBcrypt, hash) => {
          if(errBcrypt){ return res.status(500).send({error: errBcrypt}) }
          conn.query(
            'insert into usuarios(email, senha) values (?,?)',
            [req.body.email, hash],
            (error, results) => {
              conn.release();
              if(error){ return res.status(500).send({error: error}) }
              response = {
                mensagem: 'Usuário cadastrado com sucesso!',
                usuarioCriado: {
                  id_usuario: results.insertId,
                  email: req.body.email
                }
              }
              return res.status(201).send(response);
            }
          );
      }
    });
  });
});

router.post('/login', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    const query = 'select * from usuarios where email = ?';
    conn.query(query, [req.body.email], (error, results, fields) => {
      conn.release();
      if(error){ return res.status(500).send({error: error}) }
      if(results.length < 1){
        return res.status(401).send({ mensagem: 'Falha na autenticação!'});
      }
    });
  });
});
```

```

}
bcrypt.compare(req.body.senha, results[0].senha, (err, result) => {
  if(err){
    return res.status(401).send({ mensagem: 'Falha na autenticação!'});
  }
  if(result){
    return res.status(200).send({ mensagem: 'Autenticado com sucesso!'});
  }
  return res.status(401).send({ mensagem: 'Falha na autenticação!'});
});
});
});
);
});

module.exports = router;

```

- No Insomnia:

POST localhost:3000/usuarios/login		Send	200 OK	17.3 s	39 B	Just Now ▾
JSON ▾	Auth ▾ Query Header 1 Docs		Preview ▾	Header 7	Cookie	Timeline
		<pre>1 " { 2 "email": "fernando.maransatto@gmail.com", 3 "senha": "123456" 4 }</pre>				
		<pre>1 " { 2 "mensagem": "Autenticado com sucesso!" 3 }</pre>				

- Informação com erro no email:

POST localhost:3000/usuarios/login		Send	401 Unauthorized	1.64 s	39 B	Just Now ▾
JSON ▾	Auth ▾ Query Header 1 Docs		Preview ▾	Header 7	Cookie	Timeline
		<pre>1 " { 2 "email": "fernando1.maransatto@gmail.com", 3 "senha": "123456" 4 }</pre>				
		<pre>1 " { 2 "mensagem": "Falha na autenticação!" 3 }</pre>				

- Informação com erro na senha:

POST localhost:3000/usuarios/login		Send	401 Unauthorized	192 ms	39 B	Just Now ▾
JSON ▾	Auth ▾ Query Header 1 Docs		Preview ▾	Header 7	Cookie	Timeline
		<pre>1 " { 2 "email": "fernando.maransatto@gmail.com", 3 "senha": "1234567" 4 }</pre>				
		<pre>1 " { 2 "mensagem": "Falha na autenticação!" 3 }</pre>				

Retornando o token para o usuário

- Instale o jsonwebtoken:

```
npm install --save jsonwebtoken
```

```
C:\maransatto\rest>npm install --save jsonwebtoken
npm WARN rest-api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ jsonwebtoken@8.5.1
added 13 packages from 9 contributors and audited 267 packages in 33.145s

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

found 0 vulnerabilities
```

routes\usuarios.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');

router.post('/cadastro', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }

    conn.query('select * from usuarios where email = ?',
      [req.body.email],
      (error, results) => {
        if(error){ return res.status(500).send({error: error}) }
        if(results.length > 0){
          return res.status(409).send({
            mensagem: "Usuário já cadastrado!"
          });
        } else {
          bcrypt.hash(req.body.senha, 10, (errBcrypt, hash) => {
            if(errBcrypt){ return res.status(500).send({error: errBcrypt}) }
            conn.query(
              'insert into usuarios(email, senha) values (?,?)',
              [req.body.email, hash],
              (error, results) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                response = {
                  mensagem: 'Usuário cadastrado com sucesso!',
                  usuarioCriado: {
                    id_usuario: results.insertId,
                    email: req.body.email
                  }
                }
                return res.status(201).send(response);
              }
            );
          });
        }
      });
  });
});
```

```

    });
}

});

};

router.post('/login', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    const query = 'select * from usuarios where email = ?';
    conn.query(query, [req.body.email], (error, results, fields) => {
      conn.release();
      if(error){ return res.status(500).send({error: error}) }
      if(results.length < 1){
        return res.status(401).send({ mensagem: 'Falha na autenticação!'});
      }
      bcrypt.compare(req.body.senha, results[0].senha, (err, result) => {
        if(err){
          return res.status(401).send({ mensagem: 'Falha na autenticação!'});
        }
        if(result){
          const token = jwt.sign({
            id_usuario: results[0].id_usuario,
            email: results[0].email
          },
          process.env.JWT_KEY,
          {
            expiresIn: "1h"
          });
          return res.status(200).send({
            mensagem: 'Autenticado com sucesso!',
            token: token
          });
        }
        return res.status(401).send({ mensagem: 'Falha na autenticação!'});
      });
    });
  });
});

module.exports = router;

```

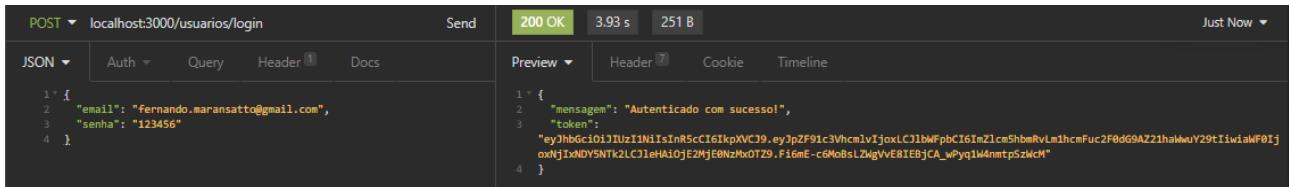
nodemon.json

```

{
  "env": {
    "MYSQL_USER": "root",
    "MYSQL_PASSWORD": "123456",
    "MYSQL_DATABASE": "ecommerce",
    "MYSQL_HOST": "localhost",
    "MYSQL_PORT": 3306,
    "JWT_KEY": "segredo"
  }
}

```

- No Insomnia:

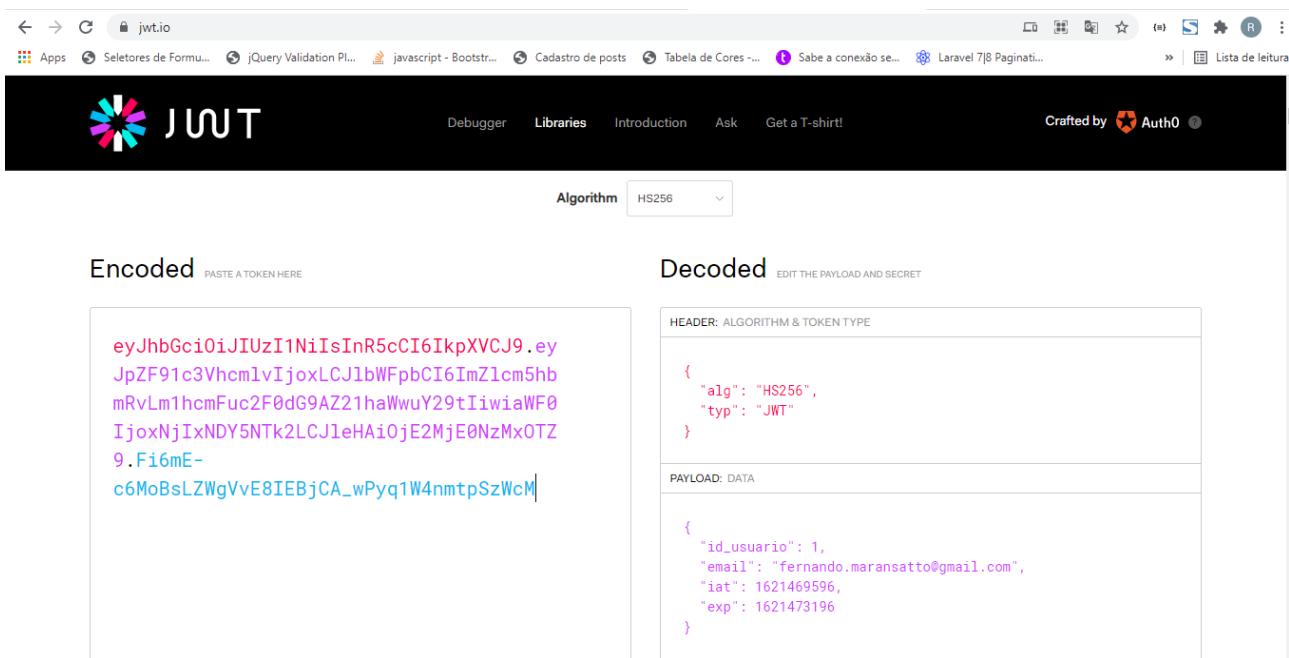


```
POST localhost:3000/usuarios/login
Send 200 OK 3.93 s 251 B Just Now
JSON Auth Query Header Docs Preview Header Cookie Timeline
1 + {
2   "mensagem": "Autenticado com sucesso!",
3   "token":
4     "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZFR91c3Vhcm1vIjoxLCJlbWFpbCI6ImZlc...  
oxNjIxNDY5NTk2LCJleHAiOjE2MjE0NzMxOTZ9.Fi6mE-c6MoBsLZWgVvE8IEBjCA_wPyq1W4nntpSzWcM"
```

token

"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZFR91c3Vhcm1vIjoxLCJlbWFpbCI6ImZlc...
mRvLm1hcmFuc2F0dG9AZ21haWwuY29tIiwiWF0IjoxNjIxNDY5NTk2LCJleHAiOjE2MjE0NzMxOTZ9.Fi6mE-c6MoBsLZWgVvE8IEBjCA_wPyq1W4nntpSzWcM"

<https://jwt.io/>



Encoded PASTE A TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

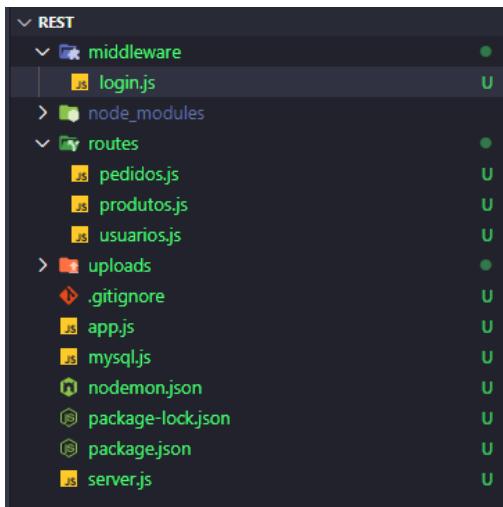
PAYOUT: DATA

```
{  
  "id_usuario": 1,  
  "email": "fernando.maransatto@gmail.com",  
  "iat": 1621469596,  
  "exp": 1621473196  
}
```

- O token fica armazenado no client e não na API. A API só fornece o token que será utilizado nas outras rotas.

Aula 13 - Protegendo as Rotas com JWT

Na pasta do projeto, crie uma subpasta chamada "middleware" e dentro dela adicione um arquivo chamado "login.js"



middleware\login.js

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
  try {
    const decode = jwt.verify(req.body.token, process.env.JWT_KEY);
    req.usuario = decode;
    next();
  } catch (error) {
    return res.status(401).send({mensagem: 'Falha na autenticação'});
  }
}
```

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');
const login = require('../middleware/login');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

});
```

```

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              imagem_produto: prod.imagem_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    )
  });
});

// Insere um produto
router.post('/', upload.single('produto_imagem'), login, (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
      [req.body.nome, req.body.preco, req.file.path],
    );
  });
});

```

```

(error, result, field) => {
  conn.release();
  if(error){ return res.status(500).send({error: error}) }
  const response = {
    mensagem: "Produto inserido com sucesso!",
    produtoCriado: {
      id: result.id_produto,
      nome: req.body.nome,
      preco: req.body.preco,
      imagem_produto: req.file.path,
      request: {
        tipo: 'POST',
        descricao: 'Retorna todos os produtos',
        url: 'http://localhost:3000/produtos'
      }
    }
  }
  return res.status(201).send(response);
}

});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Não foi encontrado produto com este Id!"
          })
        }
        const response = {
          produto: {
            id: result[0].id_produto,
            nome: result[0].nome,
            preco: result[0].preco,
            imagem_produto: result[0].imagem_produto,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os produtos',
              url: 'http://localhost:3000/produtos'
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
});

module.exports = router;

```

- No Insomnia:

POST ▾ localhost:3000/produtos Send 401 Unauthorized 306 ms 39 B

Multipart 3 ▾ Auth ▾ Query Header 1 Docs

Preview ▾ Header 7 Cookie Timeline

1 ↴ {
2 "mensagem": "Falha na autenticação!"
3 }

☰ nome	Ventilador	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
☰ preco	100.55	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
☰ produto_imagem	ventilador.jpg	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
New name	New value	

- Inserindo o token no form-data (via body):

POST ▾ localhost:3000/produtos Send 201 Created 2.12 s 272 B

Multipart 4 ▾ Auth ▾ Query Header 1 Docs

Preview ▾ Header 7 Cookie Timeline

1 ↴ {
2 "mensagem": "Produto inserido com sucesso!",
3 "produtoCriado": {
4 "nome": "Ventilador",
5 "preco": "100.55",
6 "imagem_produto": "uploads\\2021-05-20T00-52-49.551Z-ventilador.jpg",
7 "request": {
8 "tipo": "POST",
9 "descricao": "Retorna todos os produtos",
10 "url": "http://localhost:3000/produtos"
11 }
12 }
13 }

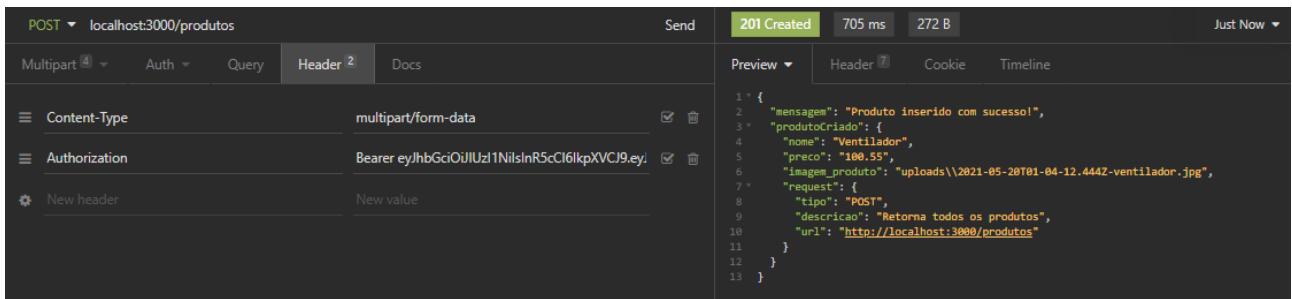
☰ nome	Ventilador	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
☰ preco	100.55	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
☰ produto_imagem	ventilador.jpg	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
☰ token	eyJhbGciOiJIUzI1NiIsInR5cCI6...	▼ <input checked="" type="checkbox"/> <input type="checkbox"/>
New name	New value	

middleware\login.js

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
  try {
    const token = req.headers.authorization.split(' ')[1];
    const decode = jwt.verify(token, process.env.JWT_KEY);
    req.usuario = decode;
    next();
  } catch (error) {
    return res.status(401).send({mensagem: 'Falha na autenticação'});
  }
}
```

- No insomnia:



POST localhost:3000/produtos Send **201 Created** 705 ms 272 B Just Now ▾

Multipart 4 Auth Query Header 2 Docs

Content-Type: multipart/form-data

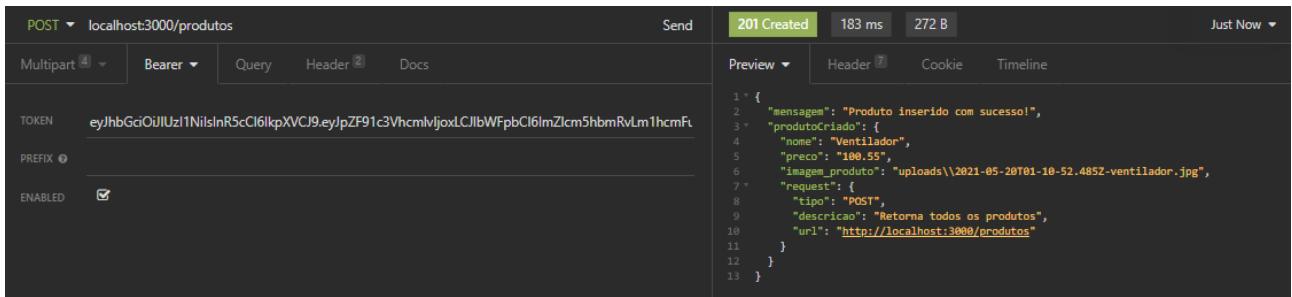
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey.

New header: New value

Preview ▾ Header 7 Cookie Timeline

```
1 " {
2   "mensagem": "Produto inserido com sucesso!",
3   "produtoCriado": {
4     "nome": "Ventilador",
5     "preco": "100.55",
6     "imagem_produto": "uploads\\2021-05-20T01-04-12.444Z-ventilador.jpg",
7     "request": {
8       "tipo": "POST",
9       "descricao": "Retorna todos os produtos",
10      "url": "http://localhost:3000/produtos"
11    }
12  }
13 }
```

Ou:



POST localhost:3000/produtos Send **201 Created** 183 ms 272 B Just Now ▾

Multipart 4 Bearer Query Header 2 Docs

TOKEN: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey/pZF91c3Vhcm1vjoxLCJlbWFpbCl6lmZlcm5hbmrLm1hcmFu

PREFIX:

ENABLED:

Preview ▾ Header 7 Cookie Timeline

```
1 " {
2   "mensagem": "Produto inserido com sucesso!",
3   "produtoCriado": {
4     "nome": "Ventilador",
5     "preco": "100.55",
6     "imagem_produto": "uploads\\2021-05-20T01-10-52.485Z-ventilador.jpg",
7     "request": {
8       "tipo": "POST",
9       "descricao": "Retorna todos os produtos",
10      "url": "http://localhost:3000/produtos"
11    }
12  }
13 }
```

Autorização para as rotas "Inserir produto", "Editar produto" e "Excluir produto"

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const mysql = require('../mysql').pool;
const multer = require('multer');
const login = require('../middleware/login');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

// Retorna todos os produtos
router.get('/', (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              imagem_produto: prod.imagem_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        res.json(response);
      }
    )
  })
})
```

```

        }
    })
}
return res.status(200).send({response});
}
)
});
});

// Insere um produto
router.post('/', upload.single('produto_imagem'), login, (req, res, next) => {
    console.log(req.file);
    const produto = {
        nome: req.body.nome,
        preco: req.body.preco
    };
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
            [req.body.nome, req.body.preco, req.file.path],
            (error, result, field) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                const response = {
                    mensagem: "Produto inserido com sucesso!",
                    produtoCriado: {
                        id: result.id_produto,
                        nome: req.body.nome,
                        preco: req.body.preco,
                        imagem_produto: req.file.path,
                        request: {
                            tipo: 'POST',
                            descricao: 'Retorna todos os produtos',
                            url: 'http://localhost:3000/produtos'
                        }
                    }
                }
                return res.status(201).send(response);
            }
        )
    });
});

// Retorna os dados de um produto
router.get('/:id', (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'SELECT * FROM produtos WHERE id_produto = ?',
            [req.params.id],
            (error, result, field) => {
                if(error){ return res.status(500).send({error: error}) };
                if(result.length == 0){
                    return res.status(404).send({
                        mensagem: "Não foi encontrado produto com este Id!"
                    })
                }
            }
        )
    });
});

```

```

const response = {
  produto: {
    id: result[0].id_produto,
    nome: result[0].nome,
    preco: result[0].preco,
    imagem_produto: result[0].imagem_produto,
    request: {
      tipo: 'GET',
      descricao: 'Retorna todos os produtos',
      url: 'http://localhost:3000/produtos'
    }
  }
}
return res.status(200).send(response);
}

// Altera um produto
router.patch('/:id', login, (req, res, next) => {
  var id = req.params.id;
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
      [req.body.nome, req.body.preco, id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto atualizado com sucesso!",
          produtoAtualizado: {
            id: id,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
              tipo: 'PATCH',
              descricao: 'Retorna os detalhes de um produto específico',
              url: 'http://localhost:3000/produtos/' + id
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
});

// Exclui um produto
router.delete('/:id', login, (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
      }
    );
  });
});

```

```
const response = {
  mensagem: "Produto removido com sucesso!",
  request: {
    tipo: 'POST',
    descricao: 'Insere um produto',
    url: 'http://localhost:3000/produtos',
    body: {
      nome: 'String',
      preco: 'Number'
    }
  }
}
return res.status(200).send(response);
}

});

module.exports = router;
```

- Gere um novo token:

POST	localhost:3000/usuarios/login	Send	200 OK	548 ms	251 B	A Minute Ago		
JSON	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
1: { 2: "email": "fernando.maransatto@gmail.com", 3: "senha": "123456" 4: }				1: { 2: "mensagem": "Autenticado com sucesso!", 3: "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZF91c3Vhcm1vJoxLCJlbWFpbCI6ImZlcm5hbmRvLm1hcmFuc2F0dG9AZ21haWwuY29tliwiaWF0ljoxNjlxNDczOTk5LCJleHAiOjE2MjE0Nzc1OTI9.AvQdvHnes1cCwyz-BVd24QgsxyFQL6XISUg7tdAqMm8" 4: }				

token:

"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZF91c3Vhcm1vJoxLCJlbWFpbCI6ImZlcm5hbmRvLm1hcmFuc2F0dG9AZ21haWwuY29tliwiaWF0ljoxNjlxNDczOTk5LCJleHAiOjE2MjE0Nzc1OTI9.AvQdvHnes1cCwyz-BVd24QgsxyFQL6XISUg7tdAqMm8"

Alteração de produto sem o token:

PATCH	localhost:3000/produtos/3	Send	401 Unauthorized	18.3 ms	39 B	Just Now		
JSON	Bearer	Query	Header	Docs	Preview	Header	Cookie	Timeline
TOKEN				1: { 2: "mensagem": "Falha na autenticação!" 3: }				
PREFIX								
ENABLED <input checked="" type="checkbox"/>								

Alteração de produto com o token:

PATCH	localhost:3000/produtos/3	Send	200 OK	245 ms	238 B	Just Now		
JSON	Bearer	Query	Header	Docs	Preview	Header	Cookie	Timeline
TOKEN				1: { 2: "mensagem": "Produto atualizado com sucesso!", 3: "produtoAtualizado": { 4: "id": "3", 5: "nome": "A Mâmia 2", 6: "preco": 69.9, 7: "request": { 8: "tipo": "PATCH", 9: "descricao": "Retorna os detalhes de um produto específico", 10: "url": "http://localhost:3000/produtos/3" 11: } 12: } 13: }				
PREFIX								
ENABLED <input checked="" type="checkbox"/>								

Exclusão de produto sem o token:

DELETE	localhost:3000/produtos/14	Send	401 Unauthorized	85.5 ms	39 B	Just Now		
Body	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
				1: { 2: "mensagem": "Falha na autenticação!" 3: }				

Exclusão de produto com o token:

DELETE	localhost:3000/produtos/14	Send	200 OK	381 ms	183 B	Just Now		
Body	Bearer	Query	Header	Docs	Preview	Header	Cookie	Timeline
TOKEN				1: { 2: "mensagem": "Produto removido com sucesso!", 3: "request": { 4: "tipo": "POST", 5: "descricao": "Insere um produto", 6: "url": "http://localhost:3000/produtos", 7: "body": { 8: "nome": "String", 9: "preco": "Number" 10: } 11: } 12: }				
PREFIX								
ENABLED <input checked="" type="checkbox"/>								

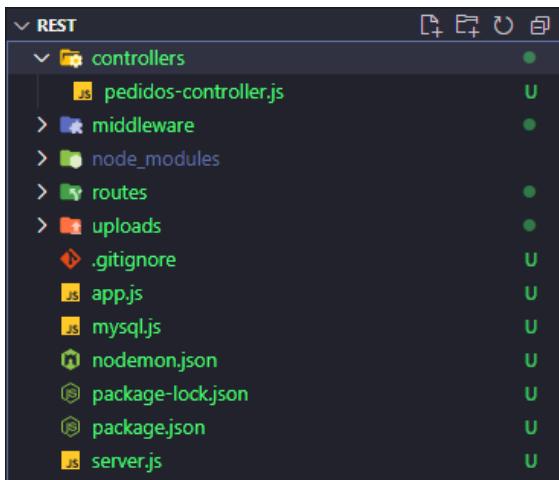
- Tente cadastrar um produto com o token vencido (depois de 1h)

The screenshot shows a POST request to the endpoint `localhost:3000/produtos`. The request method is `POST`, and the URL is `localhost:3000/produtos`. The response status is `401 Unauthorized`, with a response time of `29.2 ms` and a response size of `39 B`. The response body is a JSON object:

```
1: {  
2:   "mensagem": "Falha na autenticação!"  
3: }
```

Aula 14 - Separando Rotas e Controllers

- Na pasta do projeto crie uma subpasta chamada **controllers** e dentro adicione um arquivo chamado **pedidos-controller.js**:



controllers\pedidos-controller.js

```
const mysql = require('../mysql').pool;

exports.getPedidos = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      `select pedidos.id_pedido,
              pedidos.quantidade,
              produtos.id_produto,
              produtos.nome,
              produtos.preco
        from pedidos
       inner join produtos
      on produtos.id_produto = pedidos.id_produto;`,
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          pedidos: result.map(pedido => {
            return{
              id_pedido: pedido.id_pedido,
              quantidade: pedido.quantidade,
              produto: {
                id_produto: pedido.id_produto,
                nome: pedido.nome,
                preco: pedido.preco
              },
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um pedido específico',
                url: 'http://localhost:3000/pedidos/' + pedido.id_pedido
              }
            }
          })
        });
        res.json(response);
      })
  })
}
```

```

        }
        return res.status(200).send({response});
    }
}
};

exports.postPedidos = (req, res, next) => {

mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
        'SELECT * FROM produtos WHERE id_produto = ?',
        [req.body.id_produto],
        (error, result, field) => {
            if(error){ return res.status(500).send({error: error}) }
            if(result.length == 0){
                return res.status(404).send({
                    mensagem: "Produto não encontrado!"
                })
            }
        }
    )

    conn.query(
        'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
        [req.body.id_produto, req.body.quantidade],
        (error, result, field) => {
            conn.release();
            if(error){ return res.status(500).send({error: error}) }
            const response = {
                mensagem: "Pedido inserido com sucesso!",
                pedidoCriado: {
                    id_pedido: result.insertId,
                    id_produto: req.body.id_produto,
                    quantidade: req.body.quantidade,
                    request: {
                        tipo: 'GET',
                        descricao: 'Retorna todos os pedidos',
                        url: 'http://localhost:3000/pedidos'
                    }
                }
            }
            return res.status(201).send(response);
        }
    );
});

};

exports.getUmPedido = (req, res, next) => {
mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
        'SELECT * FROM pedidos WHERE id_pedido = ?',
        [req.params.id],
        (error, result, field) => {
            if(error){ return res.status(500).send({error: error}) };
            if(result.length == 0){
                return res.status(404).send({
                    mensagem: "Não foi encontrado pedido com este Id!"
                })
            }
        }
    );
};

```

```

        })
    }
const response = {
  pedido: {
    id_pedido: result[0].id_pedido,
    quantidade: result[0].quantidade,
    id_produto: result[0].id_produto,
    request: {
      tipo: 'GET',
      descricao: 'Retorna todos os pedidos',
      url: 'http://localhost:3000/pedidos'
    }
  }
}
return res.status(200).send(response);
}

})
};

exports.deletePedido = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM pedidos WHERE id_pedido = ?',
      [req.params.id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Pedido removido com sucesso!",
          request: {
            tipo: 'POST',
            descricao: 'Insere um pedido',
            url: 'http://localhost:3000/pedidos',
            body: {
              id_produto: 'Number',
              quantidade: 'Number'
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
};
};
```

routes\pedidos.js

```
const express = require('express');
const router = express.Router();

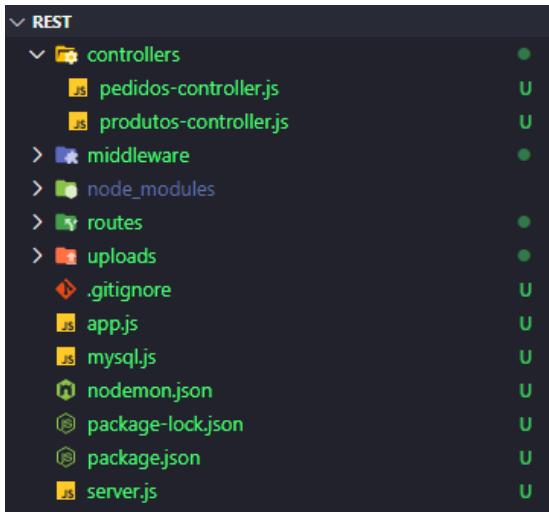
const PedidosController = require('../controllers/pedidos-controller');

router.get('/', PedidosController.getPedidos);
router.post('/', PedidosController.postPedidos);
router.get('/:id', PedidosController.getUmPedido);
router.delete('/:id', PedidosController.deletePedido);

module.exports = router;
```

- Faça todos os testes no Insomnia.

- Na pasta **controllers** adicione um arquivo chamado **produtos-controller.js**:



controllers\produtos-controller.js

```
const mysql = require('../mysql').pool;

exports.getProdutos = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              imagem_produto: prod.imagem_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: 'http://localhost:3000/produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    );
  });
};

exports.postProduto = (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
}
```

```

mysql.getConnection((error, conn) => {
  if(error){ return res.status(500).send({error: error}) }
  conn.query(
    'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
    [req.body.nome, req.body.preco, req.file.path],
    (error, result, field) => {
      conn.release();
      if(error){ return res.status(500).send({error: error}) }
      const response = {
        mensagem: "Produto inserido com sucesso!",
        produtoCriado: {
          id: result.id_produto,
          nome: req.body.nome,
          preco: req.body.preco,
          imagem_produto: req.file.path,
          request: {
            tipo: 'POST',
            descricao: 'Retorna todos os produtos',
            url: 'http://localhost:3000/produtos'
          }
        }
      }
      return res.status(201).send(response);
    }
  )
});
};

exports.getUmProduto = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, result, field) => {
        if(error){ return res.status(500).send({error: error}) };
        if(result.length == 0){
          return res.status(404).send({
            mensagem: "Não foi encontrado produto com este Id!"
          })
        }
        const response = {
          produto: {
            id: result[0].id_produto,
            nome: result[0].nome,
            preco: result[0].preco,
            imagem_produto: result[0].imagem_produto,
            request: {
              tipo: 'GET',
              descricao: 'Retorna todos os produtos',
              url: 'http://localhost:3000/produtos'
            }
          }
        }
        return res.status(200).send(response);
      }
    )
  });
};

```

```

exports.updateProduto = (req, res, next) => {
  var id = req.params.id;
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
      [req.body.nome, req.body.preco, id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto atualizado com sucesso!",
          produtoAtualizado: {
            id: id,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
              tipo: 'PATCH',
              descricao: 'Retorna os detalhes de um produto específico',
              url: 'http://localhost:3000/produtos/' + id
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
};

exports.deleteProduto = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'DELETE FROM produtos WHERE id_produto = ?',
      [req.params.id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto removido com sucesso!",
          request: {
            tipo: 'POST',
            descricao: 'Insere um produto',
            url: 'http://localhost:3000/produtos',
            body: {
              nome: 'String',
              preco: 'Number'
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
};

```

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const multer = require('multer');
const login = require('../middleware/login');

const ProdutosController = require('../controllers/produtos-controller');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

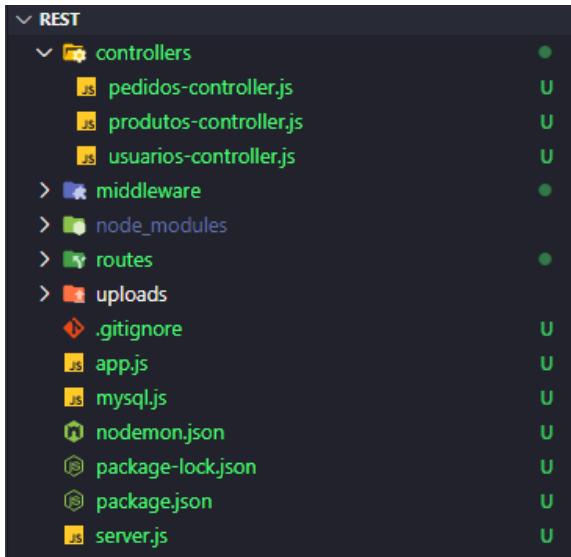
const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

router.get('/', ProdutosController.getProdutos);
router.post('/', 
  upload.single('produto_imagem'),
  login,
  ProdutosController.postProduto
);
router.get('/:id', ProdutosController.getUmProduto);
router.patch('/:id', login, ProdutosController.updateProduto);
router.delete('/:id', login, ProdutosController.deleteProduto);

module.exports = router;
```

- Faça todos os testes no Insomnia.

- Na pasta **controllers** adicione um arquivo chamado **usuarios-controller.js**:



controllers\usuarios-controller.js

```
const mysql = require('../mysql').pool;
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');

exports.cadastrarUsuario = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }

    conn.query('select * from usuarios where email = ?',
      [req.body.email],
      (error, results) => {
        if(error){ return res.status(500).send({error: error}) }
        if(results.length > 0){
          return res.status(409).send({
            mensagem: "Usuário já cadastrado!"
          });
        } else {
          bcrypt.hash(req.body.senha, 10, (errBcrypt, hash) => {
            if(errBcrypt){ return res.status(500).send({error: errBcrypt}) }
            conn.query(
              'insert into usuarios(email, senha) values (?,?)',
              [req.body.email, hash],
              (error, results) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                response = {
                  mensagem: 'Usuário cadastrado com sucesso!',
                  usuarioCriado: {
                    id_usuario: results.insertId,
                    email: req.body.email
                  }
                }
                return res.status(201).send(response);
              }
            );
          });
        }
      });
  });
};
```

```

        }
    });

exports.Login = (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        const query = 'select * from usuarios where email = ?';
        conn.query(query, [req.body.email], (error, results, fields) => {
            conn.release();
            if(error){ return res.status(500).send({error: error}) }
            if(results.length < 1){
                return res.status(401).send({ mensagem: 'Falha na autenticação!'});
            }
            bcrypt.compare(req.body.senha, results[0].senha, (err, result) => {
                if(err){
                    return res.status(401).send({ mensagem: 'Falha na autenticação!'});
                }
                if(result){
                    const token = jwt.sign({
                        id_usuario: results[0].id_usuario,
                        email: results[0].email
                    },
                    process.env.JWT_KEY,
                    {
                        expiresIn: "1h"
                    });
                    return res.status(200).send({
                        mensagem: 'Autenticado com sucesso!',
                        token: token
                    });
                }
                return res.status(401).send({ mensagem: 'Falha na autenticação!'});
            });
        });
    });
};

```

routes\usuarios.js

```

const express = require('express');
const router = express.Router();
const UsuariosController = require('../controllers/usuarios-controller');

router.post('/cadastro', UsuariosController.cadastrarUsuario);
router.post('/login', UsuariosController.Login);

module.exports = router;

```

- Faça todos os testes no Insomnia.

.gitignore

```
node_modules
.vscode
uploads/*
```

Aula 15 - Deploy no Heroku

<https://www.heroku.com/>

The screenshot shows the Heroku homepage. At the top, there's a navigation bar with links for Apps, Seletores de Formulários, jQuery Validation Pl..., javascript - Bootstrap..., Cadastro de posts, Tabela de Cores..., Sabe a conexão se..., Laravel 7|8 Paginati..., and Lista de leitura. Below the navigation is a search bar and a "Log in or Sign up" button. The main content features a large blue circular graphic with various icons representing different services and technologies like databases, servers, and code. To the right of the graphic, the text "ELEMENTS" is at the top, followed by "Powerful platform, unparalleled ecosystem". Below this, a paragraph explains Heroku's ecosystem of add-ons and buildpacks. A "SIGN UP FOR FREE" button and a "Explore Heroku Elements" link are also present.

The screenshot shows the Heroku dashboard at dashboard.heroku.com/apps. The top navigation bar is identical to the homepage. The main area displays a list of three apps under the "Personal" tab: "app-blog-nodejs" (Node.js, heroku-20, United States), "app-incidents" (PHP, heroku-20, United States), and "app-ss11" (PHP, heroku-18, United States). There is a "New" button to create a new app. A search bar at the top allows filtering of apps and pipelines.

- Crie uma nova aplicação chamada **rest-api-nodejs-maransatto**

The screenshot shows the "Create New App" form on the Heroku dashboard. The "App name" field contains "rest-api-nodejs-maransatto", which is highlighted with a green border and has a green checkmark icon. Below it, a message says "rest-api-nodejs-maransatto is available". The "Choose a region" dropdown is set to "United States". There is a "Create app" button at the bottom. A "Create New Pipeline" button is also visible above the "Create app" button.

Personal > rest-api-nodejs-maransatto

Overview Resources Deploy Metrics Activity Access Settings

Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and **promote code** between them.
[Learn more](#)

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests.
[Learn more](#)

Choose a pipeline

Deployment method
 Heroku Git Use Heroku CLI
 GitHub Connect to GitHub
 Container Registry Use Heroku CLI

Deploy using Heroku Git
Use git in the command line or a GUI tool to deploy this app.

Install the Heroku CLI
Download and install the [Heroku CLI](#).
If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository
Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a rest-api-nodejs-maransatto
```

Deploy your application
Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Existing Git repository
For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a rest-api-nodejs-maransatto
```

heroku.com Blogs Careers Documentation [Support](#)

Terms of Service Privacy Cookies © 2021 Salesforce.com

- Instale o Heroku CLI.

Profile

web: node server.js

- Entre com os comandos:

git init

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest
$ git init
Initialized empty Git repository in C:/maransatto/rest/.git/
```

```
heroku git:remote -a rest-api-nodejs-maransatto
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ heroku git:remote -a rest-api-nodejs-maransatto
»   Warning: heroku update available from 7.51.0 to 7.54.0.
set git remote heroku to https://git.heroku.com/rest-api-nodejs-maransatto.git
```

```
git remote -v
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git remote -v
heroku  https://git.heroku.com/rest-api-nodejs-maransatto.git (fetch)
heroku  https://git.heroku.com/rest-api-nodejs-maransatto.git (push)
```

```
git status
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    app.js
    controllers/
    middleware/
    mysql.js
    nodemon.json
    package-lock.json
    package.json
    routes/
    server.js

nothing added to commit but untracked files present (use "git add" to track)
```

```
git add .
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git add .
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
```

```
git status
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git status
On branch master

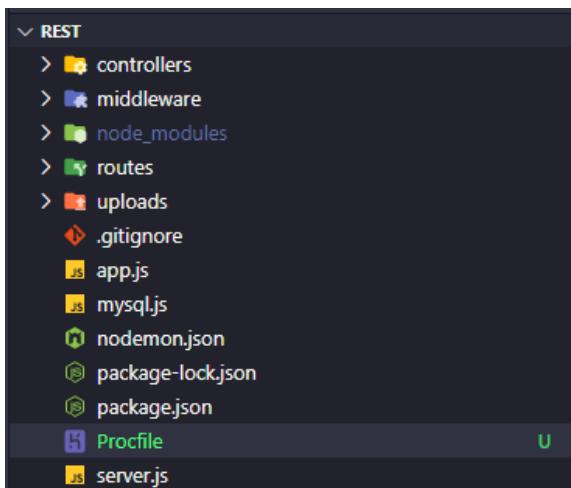
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore
    new file:   app.js
    new file:   controllers/pedidos-controller.js
    new file:   controllers/produtos-controller.js
    new file:   controllers/usuarios-controller.js
    new file:   middleware/login.js
    new file:   mysql.js
    new file:   nodemon.json
    new file:   package-lock.json
    new file:   package.json
    new file:   routes/pedidos.js
    new file:   routes/produtos.js
    new file:   routes/usuarios.js
    new file:   server.js
```

```
git commit -m "First Commit"
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git commit -m "First commit"
[master (root-commit) ec22eea] First commit
 14 files changed, 2623 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 app.js
 create mode 100644 controllers/pedidos-controller.js
 create mode 100644 controllers/produtos-controller.js
 create mode 100644 controllers/usuarios-controller.js
 create mode 100644 middleware/login.js
 create mode 100644 mysql.js
 create mode 100644 nodemon.json
 create mode 100644 package-lock.json
 create mode 100644 package.json
 create mode 100644 routes/pedidos.js
 create mode 100644 routes/produtos.js
 create mode 100644 routes/usuarios.js
 create mode 100644 server.js
```

- Na pasta do projeto adicione um arquivo chamado **Procfile** com o seguinte conteúdo:



- E faça o commit:

```
git status
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Procfile

nothing added to commit but untracked files present (use "git add" to track)
```

```
git add .
```

```
git status
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git add .

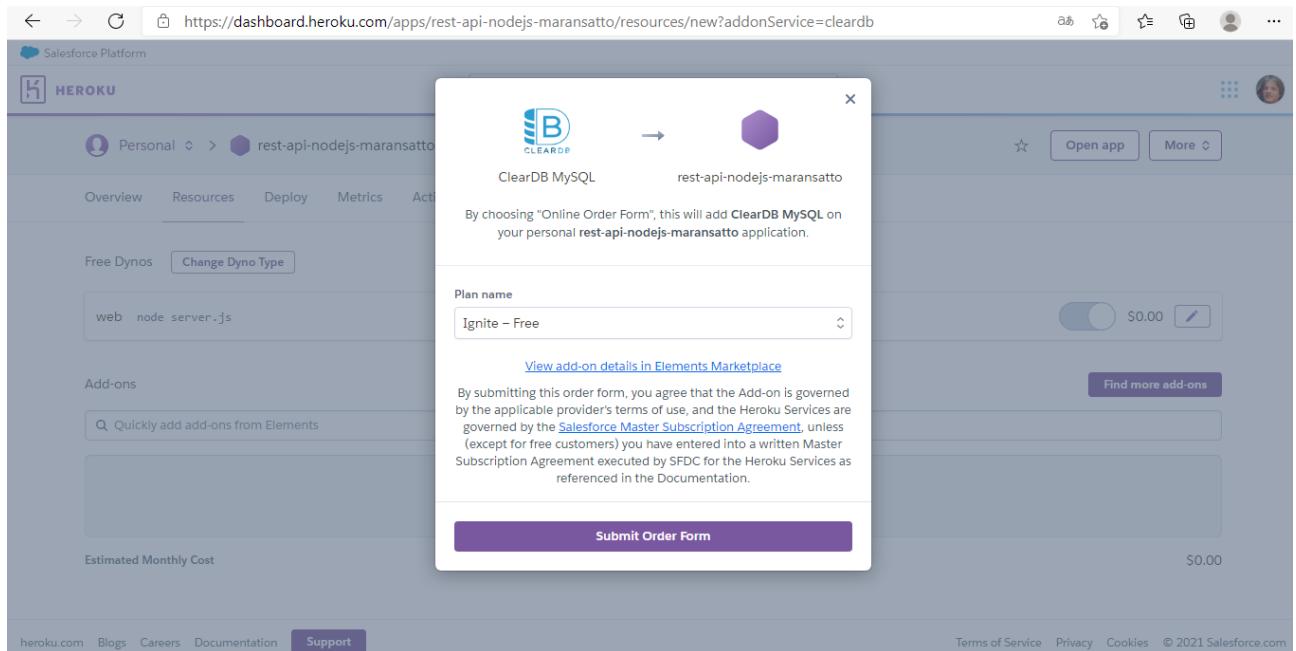
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Procfile
```

```
git commit -m "Incluido o arquivo Procfile"
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git commit -m "Incluido o arquivo Procfile"
[master cb6d64c] Incluido o arquivo Procfile
 1 file changed, 1 insertion(+)
 create mode 100644 Procfile
```

Criando o banco de dados

- No Heroku instale o addon **ClearDB MySQL**. Porém, para isso é necessário ter os dados de seu cartão de crédito registrados no Heroku.



The screenshot shows the Heroku dashboard interface. A modal window is open, titled 'ClearDB MySQL' for the application 'rest-api-nodejs-maransatto'. The modal contains instructions: 'By choosing "Online Order Form", this will add ClearDB MySQL on your personal rest-api-nodejs-maransatto application.' Below this, there's a dropdown for 'Plan name' set to 'Ignite - Free'. A note below the dropdown states: 'By submitting this order form, you agree that the Add-on is governed by the applicable provider's terms of use, and the Heroku Services are governed by the [Salesforce Master Subscription Agreement](#), unless (except for free customers) you have entered into a written Master Subscription Agreement executed by SFDC for the Heroku Services as referenced in the Documentation.' At the bottom of the modal is a purple 'Submit Order Form' button. The background of the dashboard shows the application's overview with 'Free Dynos' and 'Add-ons' sections.

- Outra opção é usar um serviço como o oferecido pela [db4free.net](#) para hospedar um banco de dados MySQL



The screenshot shows the db4free.net homepage. The top navigation bar includes links for 'Bem-vindo', 'Doações', 'Traduções', 'Log de alterações', 'Ficha técnica', 'Banco de dados', 'phpMyAdmin', 'Twitter', 'blog mpopp.net', and 'Mudar idioma [+]' (with a '+' icon). The main content area features a yellow header 'Bem-vindo ao db4free.net'. Below it, text explains the service: 'O db4free.net fornece um serviço para teste com a mais recente - as vezes versões em desenvolvimento - do MySQL Server. Você pode facilmente [criar uma conta gráta](#)s e testar suas aplicações, por exemplo, para ter certeza que seus sistemas continuarão funcionando após a atualização do MySQL. O db4free.net também serve como um bom recurso de aprendizado e para ajudá-lo a se familiarizar com as novidades introduzidas em novas versões.' It also mentions that db4free.net provides the latest MySQL version and is updated frequently. A 'Donate' button with payment method icons (VISA, MasterCard, SEPA) is visible at the bottom left.

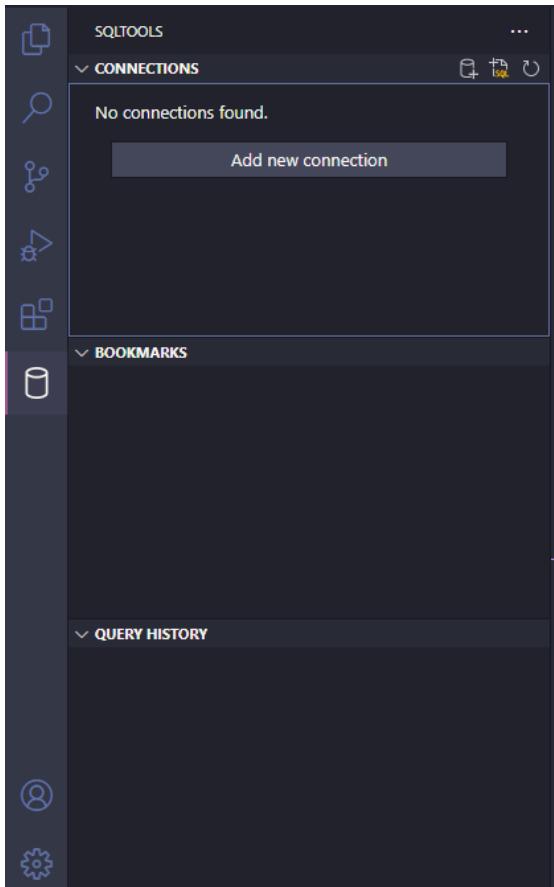
- Anote as informações da conta:

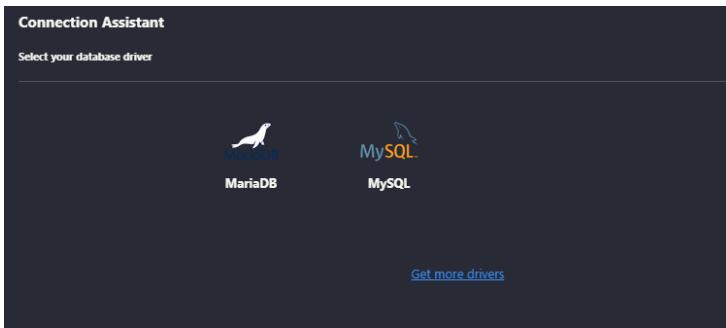
```
host = db4free.net
user = beto10561
database =
password = *****
port = 3306
```

- No Visual Studio Code, instale a extensão **SQLTools**:

The screenshot shows the SQLTools extension page in the Visual Studio Code Marketplace. At the top, there's a large blue cylinder icon representing a database. The extension name "SQLTools" is displayed in bold, along with the developer name "Matheus Teixeira" and the version "v0.23.0". It has a 4.5-star rating with 988 reviews. Below this, a brief description reads: "Database management done right. Connection explorer, query runner, intellisense, bookmarks, query history. Feel like a d...". There are buttons for "Disable" and "Uninstall", and a note: "This extension is enabled globally." A message below says: "This extension is recommended based on the files you recently opened." Navigation tabs "Details" and "Feature Contributions" are visible. The main section is titled "SQLTools for Visual Studio Code" and contains links for "DOCS", "HERE", "SUPPORTED BY", "VS CODE POWER USER COURSE →", "PATREON", "SUPPORT", "PAYPAL", and "DONATE". It also shows the "LICENSE" as "MIT". A sub-section below says: "Database management done right. Connection explorer, query runner, intellisense, bookmarks, query history. Feel like a database hero!" and provides a link to the documentation: "You can read the entire docs in <https://vscode-sqldtools.mteixeira.dev/>". A "Features" section lists some provided by SQLTools.

- Abra-o e adicione uma nova conexão com os dados anotados anteriormente (da conta do db4free.net):



This screenshot shows a detailed view of the Connection Assistant configuration for MySQL. The "Connection Settings" section contains the following fields:

- Connection name*: [empty input field]
- Connection group: [empty input field]
- Connect using*: Server and Port
- Server Address*: localhost
- Port*: 3306
- Database*: [empty input field]
- Username*: [empty input field]
- Use password: Ask on connect

Below this, under "MySQL driver specific options", there is a single field:

Authentication Protocol: default

A note below the protocol field says: "Try to switch protocols in case you have problems."

Connection name: mysql-heroku

Server Address: db4free.net

Port: 3306

Database: rest_api_nodejs

Username: beto10561

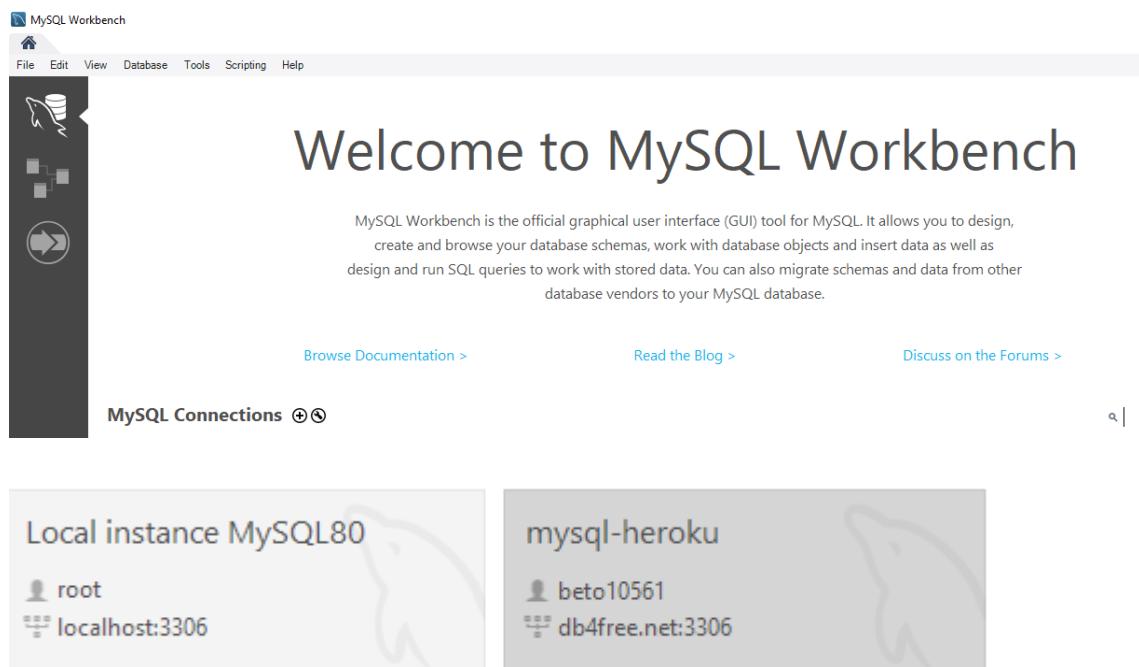
Password: *****

- Entre com o comando:

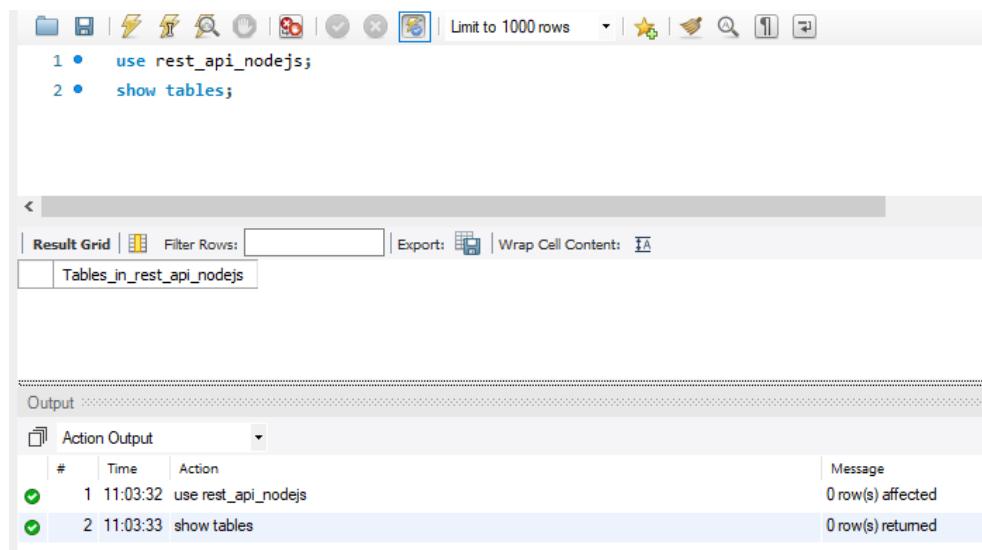
show tables;

A screenshot of a terminal window titled "api_rest_nodejs: show tables;". The window displays a single line of code: "show tables;". Below the code, a green status bar indicates "Query returned 0 rows".

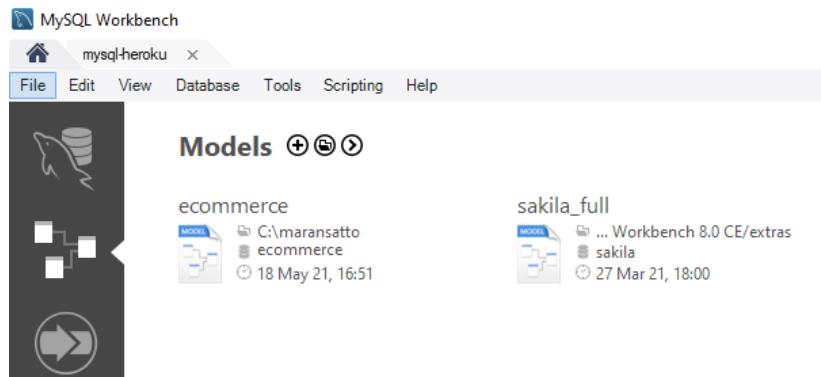
- É necessário adicionar as tabelas. Para isso entre no MySQL Workbench e adicione uma conexão com os dados anteriores:

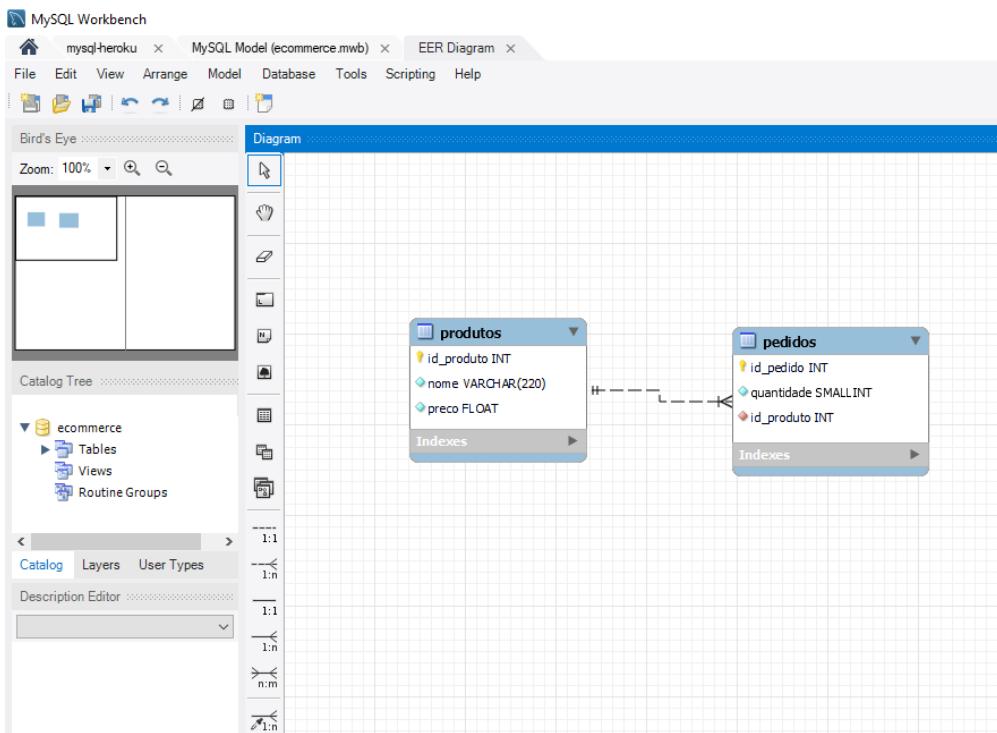


- Depois de criada a conexão, dê clique duplo para abri-la:

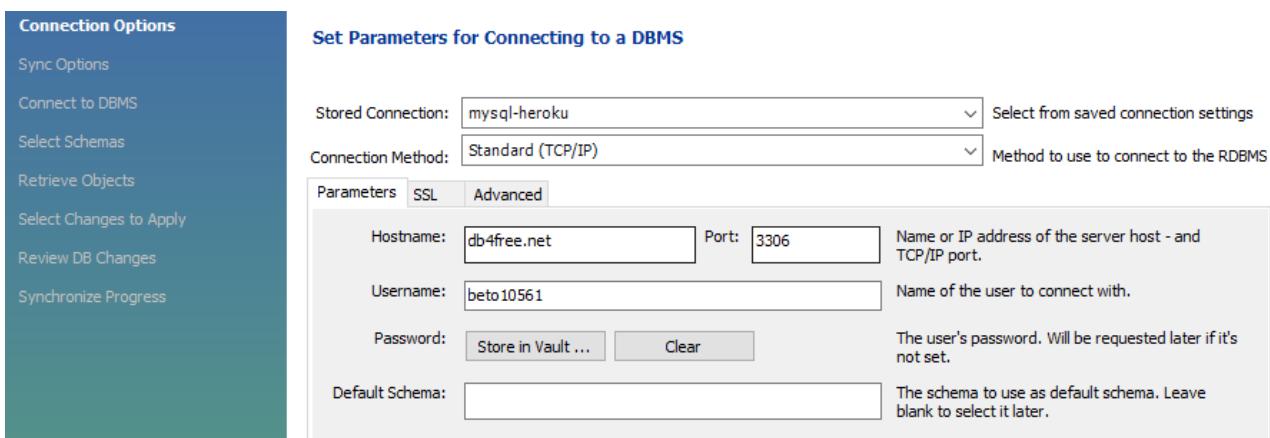


- Acesse, com clique duplo, o diagrama "ecommerce", criado em aulas anteriores:

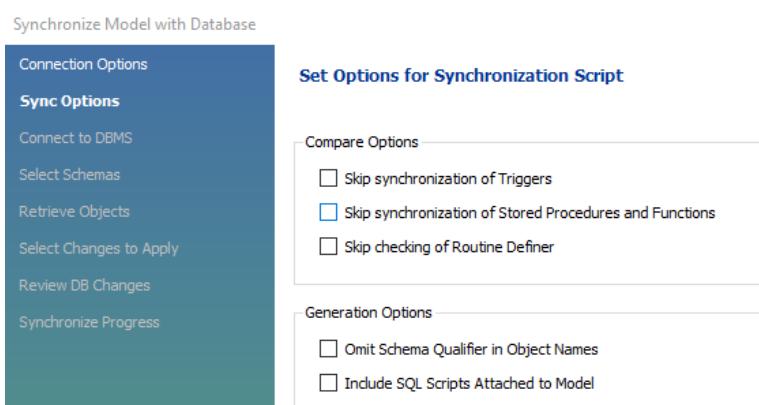




- Clique no menu **Database -> Syncronize Model**



- Clique no botão "Next":



- Clique no botão "Next" e aguarde a conexão:

Connection Options

Sync Options

Connect to DBMS

Select Schemas

Retrieve Objects

Select Changes to Apply

Review DB Changes

Synchronize Progress

Connect to DBMS and Fetch Information

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

- Connect to DBMS
- Retrieve Schema List from Database
- Check Common Server Configuration Issues

Execution Completed Successfully
Fetch finished.

Message Log

```
Fetching schema list.
OK
```

- Clique no botão "Next"

Synchronize Model with Database

Connection Options

Sync Options

Connect to DBMS

Select Schemas

Retrieve Objects

Select Changes to Apply

Review DB Changes

Synchronize Progress

Select the Schemas to be Synchronized

Select the Schemata to be Synchronized:

	Model Schema	RDBMS Schema	
<input checked="" type="checkbox"/>	ecommerce	ecommerce	schema not found in target

Override target schema to be synchronized with:

The schemata from your model are missing from the target.
If you are creating them for the first time use the Forward Engineer function.

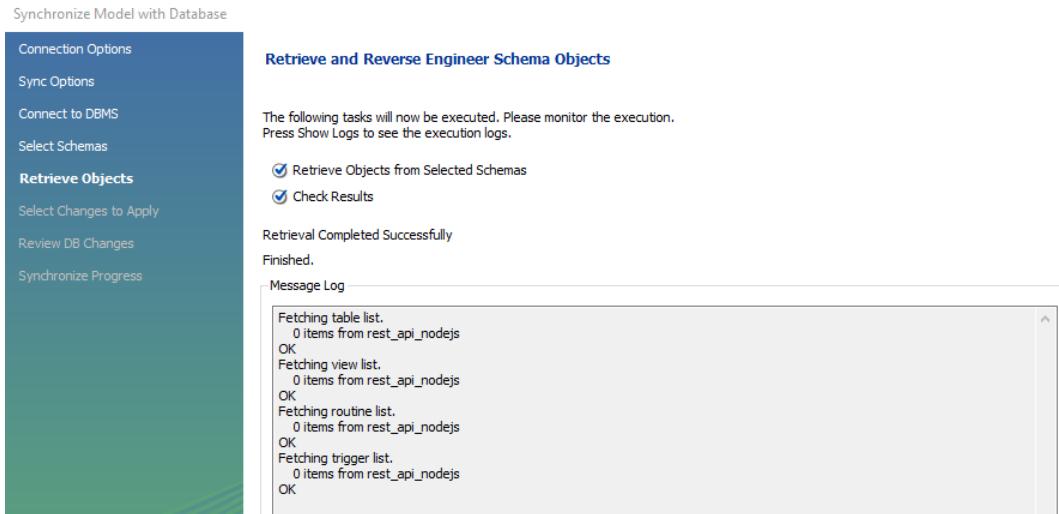
- IMPORTANTE: Marque a caixa "ecommerce" e clique no botão "Override Target". Atenção! Esse cuidado é fundamental pois se não for feito será criado uma nova BD "ecommerce"

Select the Schemata to be Synchronized:

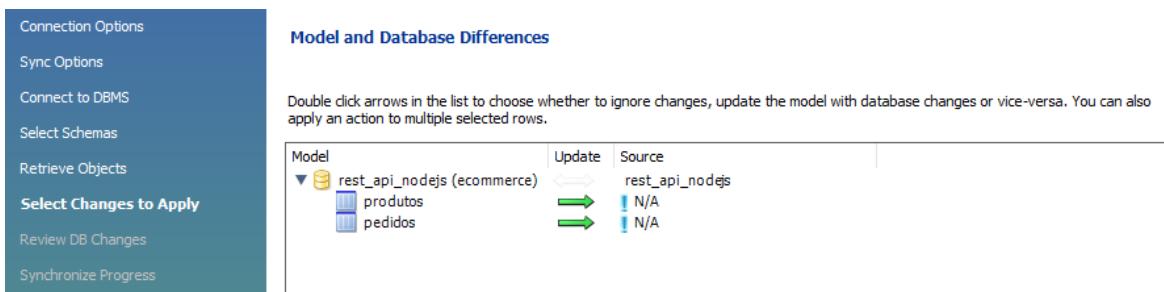
Model Schema	RDBMS Schema	
<input checked="" type="checkbox"/>	ecommerce	rest_api_nodejs

overriden

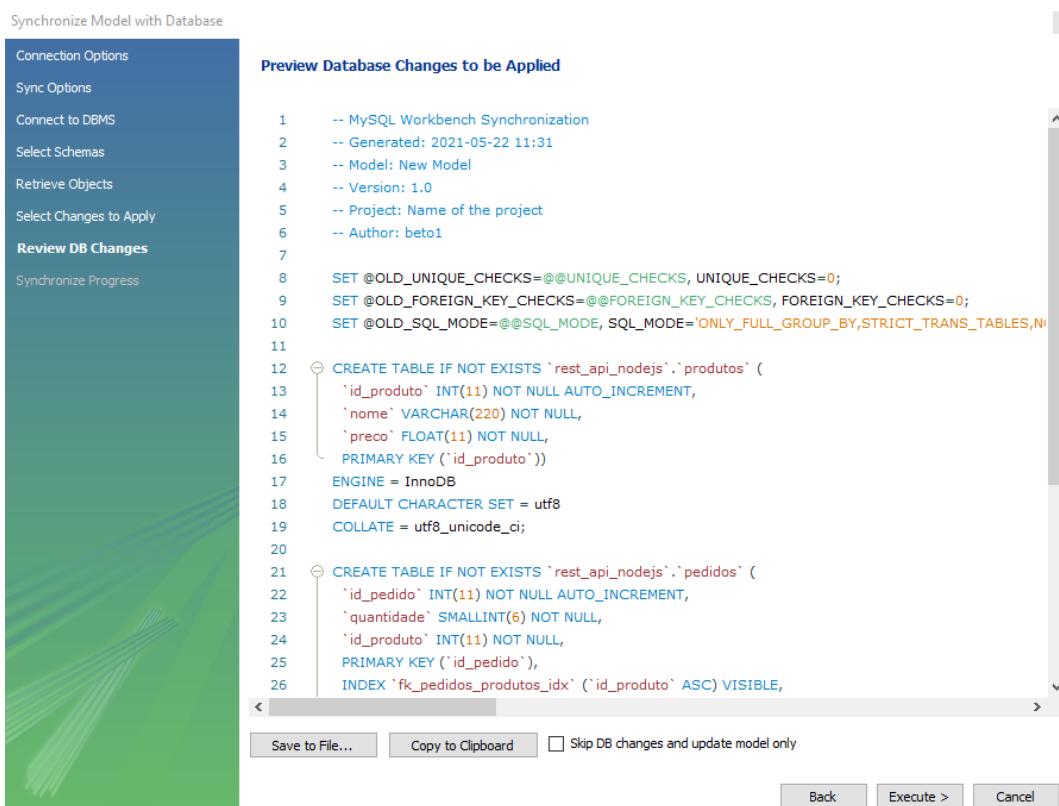
- Agora sim, clique no botão "Next" e aguarde:



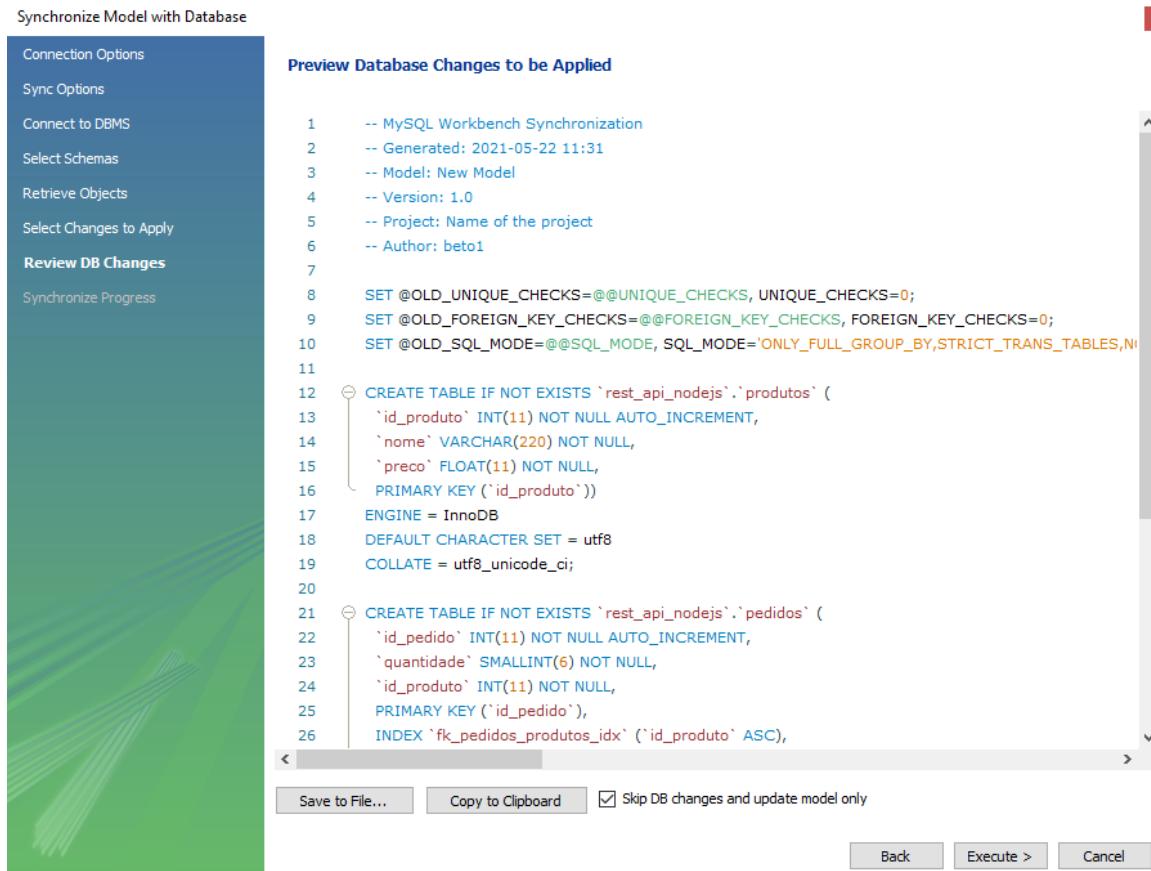
- Clique no botão "Next":



- Clique no botão "Next":



- Apague a palavra **VISIBLE** na linha 26. Se desejar salve em arquivo (Save to File...):



-- MySQL Workbench Synchronization

-- Generated: 2021-05-22 11:31

-- Model: New Model

-- Version: 1.0

-- Project: Name of the project

-- Author: beto1

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
CREATE TABLE IF NOT EXISTS `rest_api_nodejs`.`produtos` (
  `id_produto` INT(11) NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(220) NOT NULL,
  `preco` FLOAT(11) NOT NULL,
  PRIMARY KEY (`id_produto`)
)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_unicode_ci;
```

```
CREATE TABLE IF NOT EXISTS `rest_api_nodejs`.`pedidos` (
  `id_pedido` INT(11) NOT NULL AUTO_INCREMENT,
  `quantidade` SMALLINT(6) NOT NULL,
  `id_produto` INT(11) NOT NULL,
  PRIMARY KEY (`id_pedido`),
  INDEX `fk_pedidos_produtos_idx`(`id_produto` ASC),
  CONSTRAINT `fk_pedidos_produtos`
```

```

FOREIGN KEY (`id_produto`)
REFERENCES `rest_api_nodejs`.`produtos` (`id_produto`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_unicode_ci;

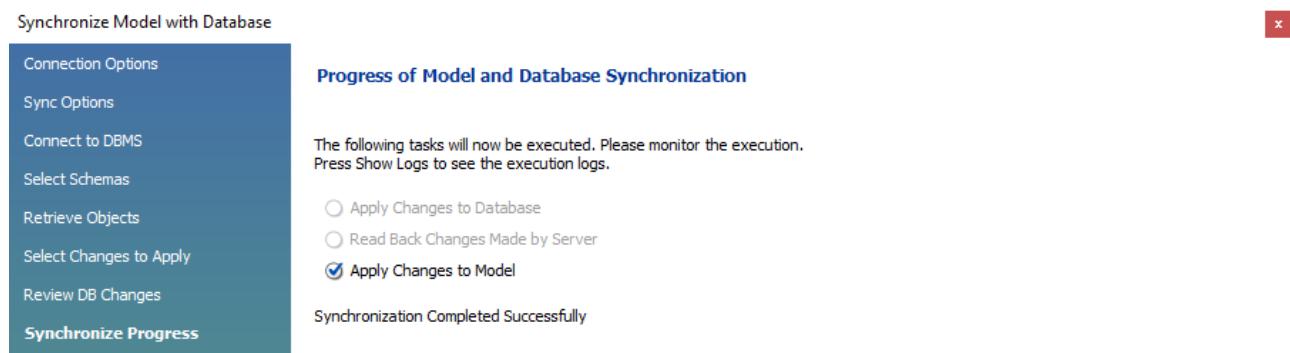
```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

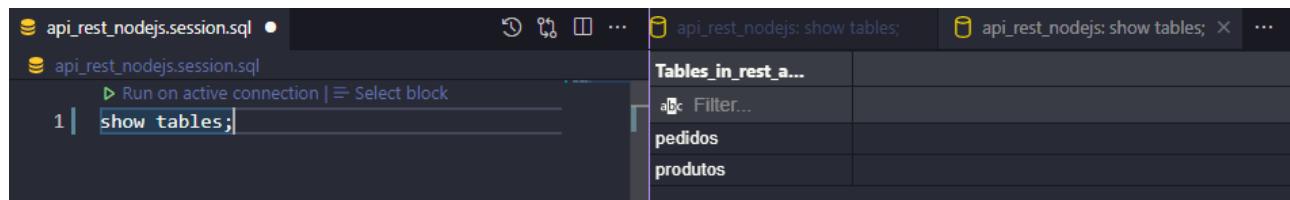
- Clique no botão "Execute":



- Clique no botão "Close":

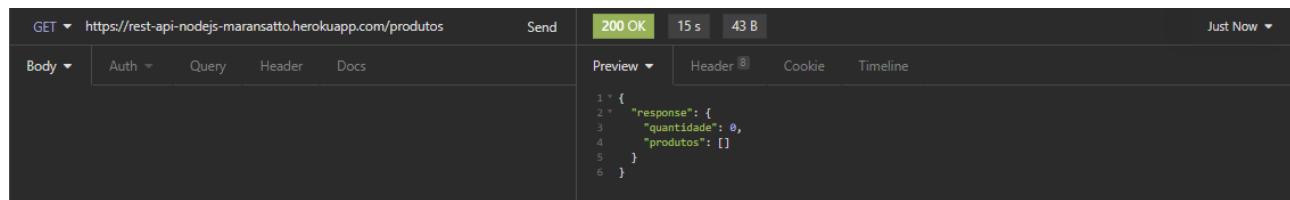
- As tabelas **produtos** e **pedidos** e seus relacionamentos foram criados. Para conferir no **SQLTools**, entre com o comando:

```
show tables;
```



- No Insomnia:

<https://rest-api-nodejs-maransatto.herokuapp.com/produtos>



- A tabela foi criada porém, ainda está sem produtos.

- A tabela de **usuários** ainda não existe, é necessário criá-la. Para isso entre, no **SQLTools** com o comando:

```
Create table usuarios (
    id_usuario integer not null primary key auto_increment,
    email varchar(220),
    senha varchar(100)
);
```

The screenshot shows two tabs in SQLTools. The left tab contains the SQL command to create the 'usuarios' table:

```
1 | Create table usuarios (
2 |     id_usuario integer not null primary key auto_increment,
3 |     email varchar(220),
4 |     senha varchar(100)
5 | );
6 |
```

The right tab shows the result of the 'show tables;' query, which returns 0 rows:

Tables_in_rest_a...
pedidos
produtos
usuarios

- No Insomnia crie dois usuários:

<https://rest-api-nodejs-maransatto.herokuapp.com/usuarios/cadastro>

The screenshot shows a POST request to <https://rest-api-nodejs-maransatto.herokuapp.com/usuarios/cadastro>. The response status is 201 Created, with a response time of 1.92 s and 120 B.

Request Body:

```
1 * {
2   "email": "fernando.maransatto@gmail.com",
3   "senha": "123456"
4 }
```

Response Preview:

```
1 * {
2   "mensagem": "Usuário cadastrado com sucesso!",
3   "usuarioCriado": {
4     "id_usuario": 1,
5     "email": "fernando.maransatto@gmail.com"
6   }
7 }
```

The screenshot shows a second POST request to <https://rest-api-nodejs-maransatto.herokuapp.com/usuarios/cadastro>. The response status is 201 Created, with a response time of 2.17 s and 117 B.

Request Body:

```
1 * {
2   "email": "roberto_pinheiro@gmail.com",
3   "senha": "123456"
4 }
```

Response Preview:

```
1 * {
2   "mensagem": "Usuário cadastrado com sucesso!",
3   "usuarioCriado": {
4     "id_usuario": 2,
5     "email": "roberto_pinheiro@gmail.com"
6   }
7 }
```

api_rest_nodejs.session.sql

t_nodejs: show tables;

api_rest_nodejs.session.sql

Run on active connection | Select block

```
1 describe produtos;
2
```

Field	Type	Null	Key
id_produto	int	NO	PRI
nome	varchar(220)	NO	
preco	float	NO	

- A coluna **imagem_produto** ainda não existe na tabela **produtos**. Para incluí-la entre com o comando:

```
alter table produtos add column imagem_produto varchar(100);
```

api_rest_nodejs.session.sql

Run on active connection | Select block

```
1 alter table produtos add column imagem_produto varchar(100);
```

Query returned 0 rows

api_rest_nodejs.session.sql

Run on active connection | Select block

```
1 describe produtos;
```

Field	Type	Null	Key
id_produto	int	NO	PRI
nome	varchar(220)	NO	
preco	float	NO	
imagem_produto	varchar(100)	YES	

nodemon.json

```
{  
  "env": {  
    "MYSQL_USER": "beto10561",  
    "MYSQL_PASSWORD": "*****",  
    "MYSQL_DATABASE": "rest_api_nodejs",  
    "MYSQL_HOST": "db4free.net",  
    "MYSQL_PORT": 3306,  
  
    "JWT_KEY": "segredo",  
    "URL_API": "http://localhost:3000/"  
  }  
}
```

- Faça as alterações nos controllers de pedidos e produtos:

controllers\pedidos-controller.js

```
const mysql = require('../mysql').pool;  
  
exports.getPedidos = (req, res, next) => {  
  mysql.getConnection((error, conn) => {  
    if(error){ return res.status(500).send({error: error}) }  
    conn.query(  
      `select pedidos.id_pedido,  
            pedidos.quantidade,  
            produtos.id_produto,  
            produtos.nome,  
            produtos.preco  
      from pedidos  
      inner join produtos  
      on produtos.id_produto = pedidos.id_produto;`,  
      (error, result, field) => {  
        if(error){ return res.status(500).send({error: error}) };  
        const response = {  
          pedidos: result.map(pedido => {  
            return{  
              id_pedido: pedido.id_pedido,  
              quantidade: pedido.quantidade,  
              produto: {  
                id_produto: pedido.id_produto,  
                nome: pedido.nome,  
                preco: pedido.preco  
              },  
              request: {  
                tipo: 'GET',  
                descricao: 'Retorna os detalhes de um pedido específico',  
                url: process.env.URL_API + 'pedidos/' + pedido.id_pedido  
              }  
            }  
          })  
        }  
        return res.status(200).send({response});  
      }  
    )  
  })  
}
```

```

        )
    });
};

exports.postPedidos = (req, res, next) => {

    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'SELECT * FROM produtos WHERE id_produto = ?',
            [req.body.id_produto],
            (error, result, field) => {
                if(error){ return res.status(500).send({error: error}) }
                if(result.length == 0){
                    return res.status(404).send({
                        mensagem: "Produto não encontrado!"
                    })
                }

                conn.query(
                    'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)',
                    [req.body.id_produto, req.body.quantidade],
                    (error, result, field) => {
                        conn.release();
                        if(error){ return res.status(500).send({error: error}) }
                        const response = {
                            mensagem: "Pedido inserido com sucesso!",
                            pedidoCriado: {
                                id_pedido: result.insertId,
                                id_produto: req.body.id_produto,
                                quantidade: req.body.quantidade,
                                request: {
                                    tipo: 'GET',
                                    descricao: 'Retorna todos os pedidos',
                                    url: process.env.URL_API + 'pedidos'
                                }
                            }
                        }
                        return res.status(201).send(response);
                    }
                )
            }
        );
    });

    exports.getUmPedido = (req, res, next) => {
        mysql.getConnection((error, conn) => {
            if(error){ return res.status(500).send({error: error}) }
            conn.query(
                'SELECT * FROM pedidos WHERE id_pedido = ?',
                [req.params.id],
                (error, result, field) => {
                    if(error){ return res.status(500).send({error: error}) };
                    if(result.length == 0){
                        return res.status(404).send({
                            mensagem: "Não foi encontrado pedido com este Id!"
                        })
                    }
                }
            );
        });
    };
}

```

```

        }
    const response = {
        pedido: {
            id_pedido: result[0].id_pedido,
            quantidade: result[0].quantidade,
            id_produto: result[0].id_produto,
            request: {
                tipo: 'GET',
                descricao: 'Retorna todos os pedidos',
                url: process.env.URL_API + 'pedidos'
            }
        }
    }
    return res.status(200).send(response);
}
}

};

exports.deletePedido = (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'DELETE FROM pedidos WHERE id_pedido = ?',
            [req.params.id],
            (error, result, field) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                const response = {
                    mensagem: "Pedido removido com sucesso!",
                    request: {
                        tipo: 'POST',
                        descricao: 'Insere um pedido',
                        url: process.env.URL_API + 'pedidos',
                        body: {
                            id_produto: 'Number',
                            quantidade: 'Number'
                        }
                    }
                }
                return res.status(200).send(response);
            }
        );
    });
};

```

controllers\produtos-controller.js

```

const mysql = require('../mysql').pool;

exports.getProdutos = (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'SELECT * FROM produtos',
            (error, result, field) => {
                if(error){ return res.status(500).send({error: error}) };
                const response = {

```

```

        quantidade: result.length,
        produtos: result.map(prod => {
            return{
                id: prod.id_produto,
                nome: prod.nome,
                preco: prod.preco,
                imagem_produto: prod.imagem_produto,
                request: {
                    tipo: 'GET',
                    descricao: 'Retorna os detalhes de um produto específico',
                    url: process.env.URL_API + 'produtos/' + prod.id_produto
                }
            }
        })
    }
    return res.status(200).send({response});
}
)
};

exports.postProduto = (req, res, next) => {
    console.log(req.file);
    const produto = {
        nome: req.body.nome,
        preco: req.body.preco
    };
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
            [req.body.nome, req.body.preco, req.file.path],
            (error, result, field) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                const response = {
                    mensagem: "Produto inserido com sucesso!",
                    produtoCriado: {
                        id: result.id_produto,
                        nome: req.body.nome,
                        preco: req.body.preco,
                        imagem_produto: req.file.path,
                        request: {
                            tipo: 'POST',
                            descricao: 'Retorna todos os produtos',
                            url: process.env.URL_API + 'produtos'
                        }
                    }
                }
                return res.status(201).send(response);
            }
        );
    });
};

exports.getUmProduto = (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(

```

```

'SELECT * FROM produtos WHERE id_produto = ?',
[req.params.id],
(error, result, field) => {
  if(error){ return res.status(500).send({error: error}) };
  if(result.length == 0){
    return res.status(404).send({
      mensagem: "Não foi encontrado produto com este Id!"
    })
  }
  const response = {
    produto: {
      id: result[0].id_produto,
      nome: result[0].nome,
      preco: result[0].preco,
      imagem_produto: result[0].imagem_produto,
      request: {
        tipo: 'GET',
        descricao: 'Retorna todos os produtos',
        url: process.env.URL_API + 'produtos'
      }
    }
  }
  return res.status(200).send(response);
}
});

exports.updateProduto = (req, res, next) => {
  var id = req.params.id;
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
      [req.body.nome, req.body.preco, id],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto atualizado com sucesso!",
          produtoAtualizado: {
            id: id,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
              tipo: 'PATCH',
              descricao: 'Retorna os detalhes de um produto específico',
              url: process.env.URL_API + 'produtos/' + id
            }
          }
        }
        return res.status(200).send(response);
      }
    );
  });
};

exports.deleteProduto = (req, res, next) => {
  mysql.getConnection((error, conn) => {

```

```

if(error){ return res.status(500).send({error: error}) }
conn.query(
  'DELETE FROM produtos WHERE id_produto = ?',
  [req.params.id],
  (error, result, field) => {
    conn.release();
    if(error){ return res.status(500).send({error: error}) }
    const response = {
      mensagem: "Produto removido com sucesso!",
      request: {
        tipo: 'POST',
        descricao: 'Insere um produto',
        url: process.env.URL_API + 'produtos',
        body: {
          nome: 'String',
          preco: 'Number'
        }
      }
    }
    return res.status(200).send(response);
  }
);
});
;

```

- Faça o commit e o deploy:

```

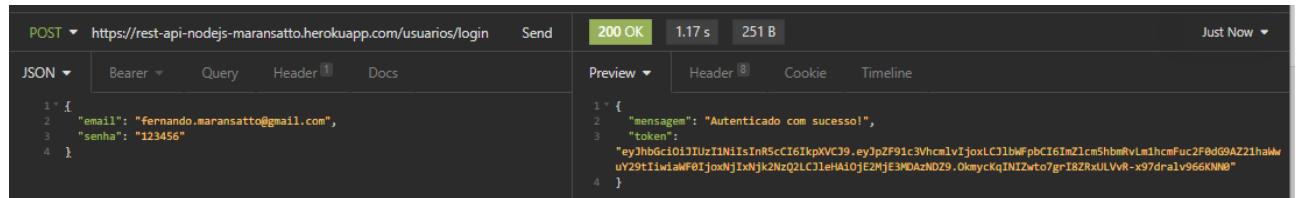
git add .
git commit -m "Alterações feitas"
git push heroku master

```

- No Heroku, crie as variáveis de ambiente:

Config Vars		Hide Config Vars
JWT_KEY	segredo	
MYSQL_DATABASE	rest_api_nodejs	
MYSQL_HOST	db4free.net	
MYSQL_PASSWORD	*****	
MYSQL_PORT	3306	
MYSQL_USER	beto10561	
URL_API	https://rest-api-nodejs-maransatto.herokuapp.com	
KEY	VALUE	Add

- Vamos obter um token para cadastrar o produto:



POST https://rest-api-nodejs-maransatto.herokuapp.com/usuarios/login Send

200 OK 1.17 s 251 B Just Now

Preview Header Cookie Timeline

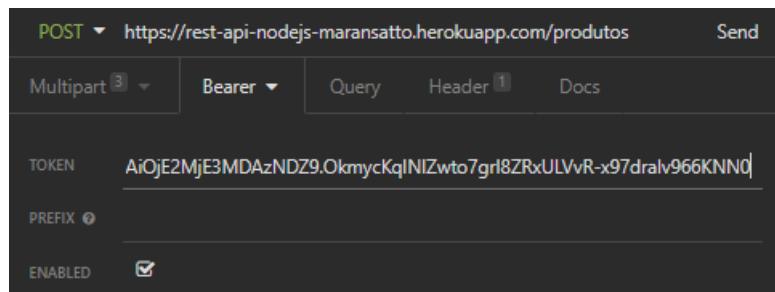
JSON Bearer Query Header 1 Docs

```
1 "email": "fernando.maransatto@gmail.com",
2 "senha": "123456"
3 }
```

```
1 "mensagem": "Autenticado com sucesso!",
2 "token":
3 "eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJpZF91c3VhcmlvIjoxLCJlbWFpbCI6ImZlcm5hbmRvLm1hcmFuc2F0dG9AZ21haW
aWwuY29tliwiaWF0IjoxNjlxNjk2NzQ2LCJleHAIoJE2MjE3MDAzNDZ9.OkmycKqINIzwt07grl8ZRxULVvR-x97dralv966KNN0"
4 }
```

```
{
  "mensagem": "Autenticado com sucesso!",
  "token":
  "eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJpZF91c3VhcmlvIjoxLCJlbWFpbCI6ImZlcm5hbmRvLm1hcmFuc2F0dG9AZ21haW
  aWwuY29tliwiaWF0IjoxNjlxNjk2NzQ2LCJleHAIoJE2MjE3MDAzNDZ9.OkmycKqINIzwt07grl8ZRxULVvR-
  x97dralv966KNN0"
}
```

- Para cadastrar um produto copie e cole o token:



POST https://rest-api-nodejs-maransatto.herokuapp.com/produtos Send

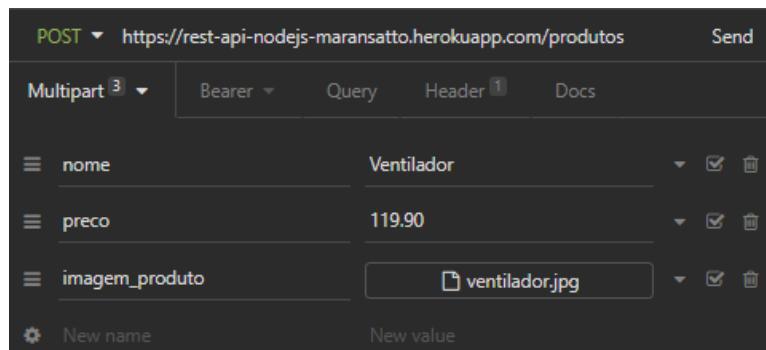
Multipart 3 Bearer Query Header 1 Docs

TOKEN AiOjE2MjE3MDAzNDZ9.OkmycKqINIzwt07grl8ZRxULVvR-x97dralv966KNN0

PREFIX ?

ENABLED

- Cadastre dois produtos:



POST https://rest-api-nodejs-maransatto.herokuapp.com/produtos Send

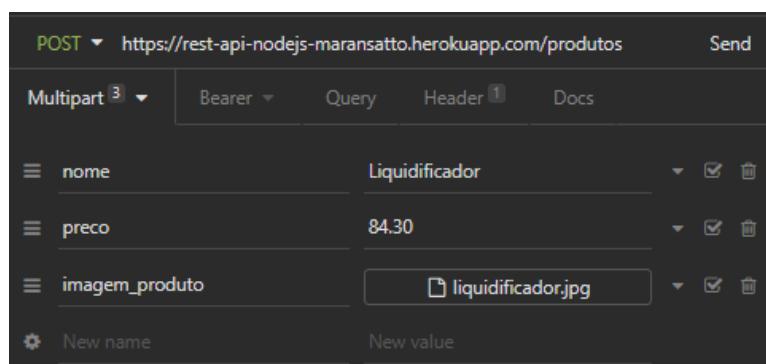
Multipart 3 Bearer Query Header 1 Docs

nome Ventilador

preco 119.90

imagem_produto ventilador.jpg

New name New value



POST https://rest-api-nodejs-maransatto.herokuapp.com/produtos Send

Multipart 3 Bearer Query Header 1 Docs

nome Liquidificador

preco 84.30

imagem_produto liquidificador.jpg

New name New value

- Dessa forma o cadastro de produtos só é possível, usando o **ClearDB MySQL** (da Heroku). Com um servidor externo não funciona.

- Portanto vamos, cadastrá-los diretamente na tabela **produtos**:

The screenshot shows the phpMyAdmin interface for a MySQL 8.0 Server. The left sidebar lists databases: Novo, information_schema, and rest_api_nodejs. The main area shows the 'rest_api_nodejs' database's structure. A table named 'produtos' is selected, displaying three rows of data. The columns are labeled: Tabela, Acções, Registos, Tipo, Agrupamento (Collation), Tamanho, and Suspensa. The data shows three entries: 'pedidos' (32.0 KB), 'produtos' (16.0 KB), and 'usuarios' (16.0 KB). The total size for the 'produtos' table is 64.0 KB.

The screenshot shows the phpMyAdmin interface for the 'produtos' table. At the top, there is a green message: "MySQL não retornou nenhum registo. (A consulta demorou 0.0028 segundos.)". Below it is a SQL query: "SELECT * FROM `produtos`". There are several operation buttons: Procurar, Estrutura, SQL, Pesquisar, Insere, Exportar, Importar, Operações, Rastreando, and Acionadores. Below the query, there is a table structure with columns: id_produto, nome, preco, and imagem_produto. A section titled "Operações resultantes das consultas" contains buttons for "Criar visualização" and "Marcar este comando SQL". There is also a "Rótulo:" input field and a checkbox for "Deixar todos os utilizadores acederem a este marcador".

- Clique na opção de menu "**Insere**" e cadastre os dois produtos:

The screenshot shows the 'Insere' (Insert) form for the 'produtos' table. The form has four fields: 'id_produto' (int), 'nome' (varchar(220)), 'preco' (float), and 'imagem_produto' (varchar(100)). The 'nome' field is populated with 'Ventilador', 'preco' is set to '119.90', and 'imagem_produto' has a dropdown menu with a checked option. The 'Coluna' column header is visible at the top of the table structure.

Servidor: MySQL 8.0 Server:3306 » Banco de dados: rest_api_nodejs » Tabela: produtos

Procurar Estrutura SQL Pesquisar Inserção Exportar Importar Operações Rastreando Acionadores

Coluna	Tipo	Funções	Nulo	Valor
id_produto	int			
nome	varchar(220)			Liquidificador
preco	float			84.30
imagem_produto	varchar(100)			<input checked="" type="checkbox"/>

Executar

Servidor: MySQL 8.0 Server:3306 » Banco de dados: rest_api_nodejs » Tabela: produtos

Procurar Estrutura SQL Pesquisar Inserção Exportar Importar Operações Rastreando Acionadores

A mostrar registos de 0 - 1 (total, A consulta demorou 0.0013 segundos.)

```
SELECT * FROM `produtos`
```

Perfil [Editar em linha] [Edita] [Explicar SQL] [Criar código PHP] [Actualizar]

Mostrar tudo | Número de registo: 25 | Filtrar registo: Pesquisar esta tabela | Ordenar pela chave: Nenhum

+ Opções

			id_produto	nome	preco	imagem_produto	
<input type="checkbox"/>	 Edita	 Copiar	 Apagar	1	Ventilador	119.9	NULL
<input type="checkbox"/>	 Edita	 Copiar	 Apagar	2	Liquidificador	84.3	NULL

↑ Marcar todos Com os seleccionados:  Edita  Copiar  Apagar  Exportar

Mostrar tudo | Número de registo: 25 | Filtrar registo: Pesquisar esta tabela | Ordenar pela chave: Nenhum

- No Insomnia:

<https://rest-api-nodejs-maransatto.herokuapp.com/produtos>

GET https://rest-api-nodejs-maransatto.herokuapp.com/produtos Send 200 OK 1.53 s 481 B

Body	Auth	Query	Header	Docs	Preview	Header	Cookie	Timeline
					<pre>1 * { 2 * "response": { 3 * "quantidade": 2, 4 * "produtos": [5 * { 6 * "id": 1, 7 * "nome": "Ventilador", 8 * "preco": 119.9, 9 * "imagem_produto": null, 10 * "request": { 11 * "tipo": "GET", 12 * "descricao": "Retorna os detalhes de um produto específico", 13 * "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos/1" 14 * } 15 * }, 16 * { 17 * "id": 2, 18 * "nome": "Liquidificador", 19 * "preco": 84.3, 20 * "imagem_produto": null, 21 * "request": { 22 * "tipo": "GET", 23 * "descricao": "Retorna os detalhes de um produto específico", 24 * "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos/2" 25 * } 26 * } 27 *] 28 * } 29 * }</pre>			

Select a body type from above



← → ⌂ rest-api-nodejs-maransatto.herokuapp.com/produtos/1

Apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstr... Cadastro de posts Tabe

```
1 // 20210522125405
2 // https://rest-api-nodejs-maransatto.herokuapp.com/produtos/1
3
4 {
5     "produto": {
6         "id": 1,
7         "nome": "Ventilador",
8         "preco": 119.9,
9         "imagem_produto": null,
10    "request": {
11        "tipo": "GET",
12        "descricao": "Retorna todos os produtos",
13        "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos"
14    }
15 }
16 }
```

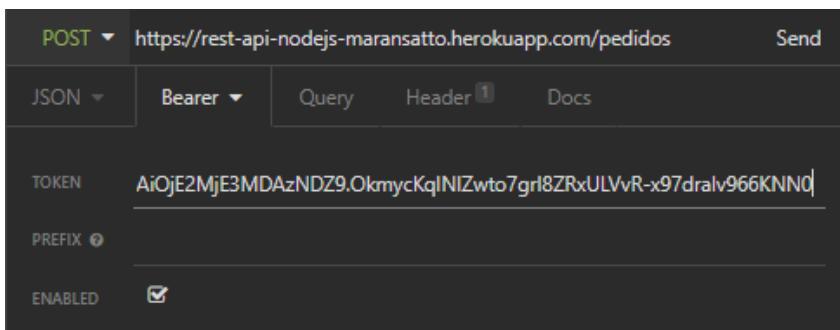
← → ⌂ rest-api-nodejs-maransatto.herokuapp.com/produtos/2

Apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstr... Cadastro de posts Tabe

```
1 // 20210522125456
2 // https://rest-api-nodejs-maransatto.herokuapp.com/produtos/2
3
4 {
5     "produto": {
6         "id": 2,
7         "nome": "Liquidificador",
8         "preco": 84.3,
9         "imagem_produto": null,
10    "request": {
11        "tipo": "GET",
12        "descricao": "Retorna todos os produtos",
13        "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos"
14    }
15 }
16 }
```

Cadastrando dois pedidos no Insomnia

- Inicialmente, copie o token para **Bearer Token**:



The screenshot shows the Insomnia API client interface. A POST request is being made to the URL <https://rest-api-nodejs-maransatto.herokuapp.com/pedidos>. The 'Header' tab is selected, showing a 'Bearer' header with the value `AiOjE2MjE3MDAzNDZ9.OkmvcKqlNIZwto7grl8ZRxULVvR-x97dralv966KNN0`. Other tabs visible include 'JSON', 'Query', and 'Docs'. A checkbox labeled 'ENABLED' is checked.

<https://rest-api-nodejs-maransatto.herokuapp.com/pedidos>

POST ▾	https://rest-api-nodejs-maransatto.herokuapp.com/pedidos	Send	201 Created	1.59 s	233 B			
JSON ▾	Bearer ▾	Query	Header ▾	Docs	Preview ▾	Header ▾	Cookie	Timeline
<pre>1 " { 2 "id_produto": 1, 3 "quantidade": 5 4 }</pre>				<pre>1 " { 2 "mensagem": "Pedido inserido com sucesso!", 3 "pedidoCriado": { 4 "id_pedido": 1, 5 "id_produto": 1, 6 "quantidade": 5, 7 "request": { 8 "tipo": "GET", 9 "descricao": "Retorna todos os pedidos", 10 "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos" 11 } 12 } 13 }</pre>				

POST ▾	https://rest-api-nodejs-maransatto.herokuapp.com/pedidos	Send	201 Created	1.1 s	234 B			
JSON ▾	Bearer ▾	Query	Header ▾	Docs	Preview ▾	Header ▾	Cookie	Timeline
<pre>1 " { 2 "id_produto": 2, 3 "quantidade": 10 4 }</pre>				<pre>1 " { 2 "mensagem": "Pedido inserido com sucesso!", 3 "pedidoCriado": { 4 "id_pedido": 2, 5 "id_produto": 2, 6 "quantidade": 10, 7 "request": { 8 "tipo": "GET", 9 "descricao": "Retorna todos os pedidos", 10 "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos" 11 } 12 } 13 }</pre>				

← → ⌂ rest-api-nodejs-maransatto.herokuapp.com/pedidos

_apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstrap...

```
1 // 20210522130431
2 // https://rest-api-nodejs-maransatto.herokuapp.com/pedidos
3
4 {
5   "response": {
6     "pedidos": [
7       {
8         "id_pedido": 1,
9         "quantidade": 5,
10        "produto": {
11          "id_produto": 1,
12          "nome": "Ventilador",
13          "preco": 119.9
14        },
15        "request": {
16          "tipo": "GET",
17          "descricao": "Retorna os detalhes de um pedido específico",
18          "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos/1"
19        }
20      },
21      {
22        "id_pedido": 2,
23        "quantidade": 10,
24        "produto": {
25          "id_produto": 2,
26          "nome": "Liquidificador",
27          "preco": 84.3
28        },
29        "request": {
30          "tipo": "GET",
31          "descricao": "Retorna os detalhes de um pedido específico",
32          "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos/2"
33        }
34      }
35    ]
36  }
37 }
```

Listando todos os pedidos no Insomnia

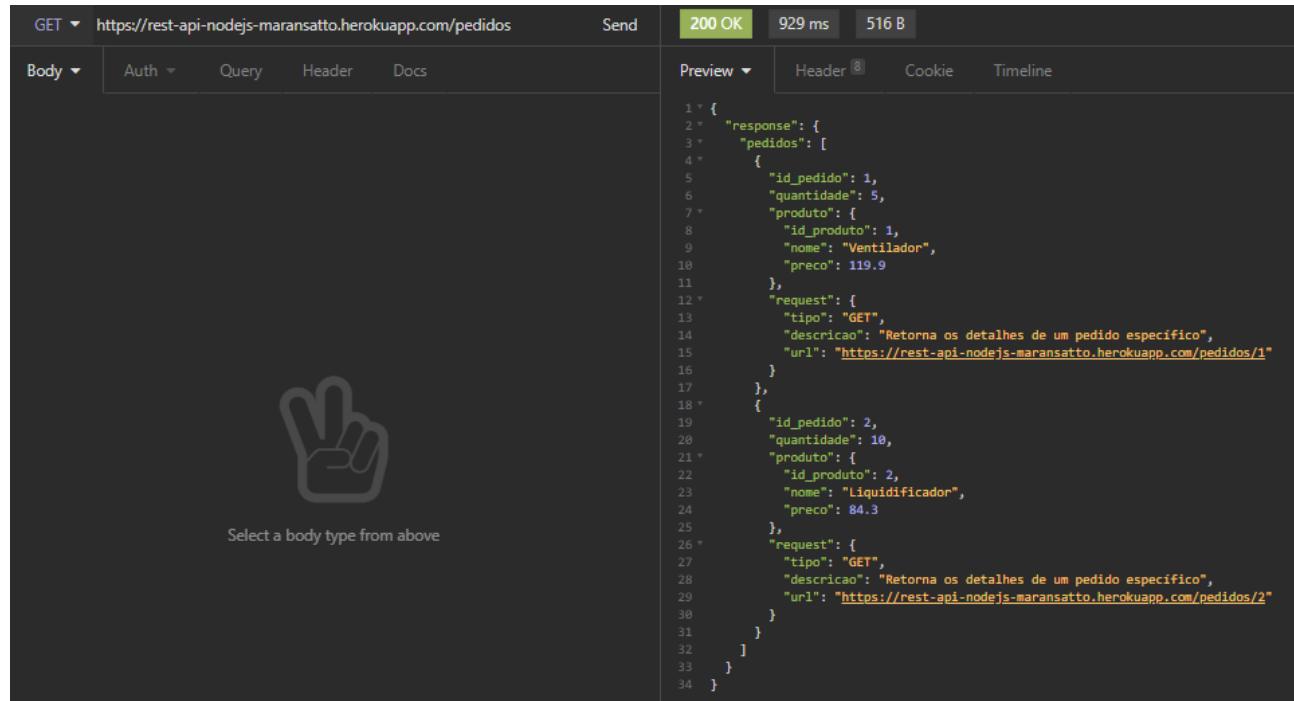
<https://rest-api-nodejs-maransatto.herokuapp.com/pedidos>

GET ▾ https://rest-api-nodejs-maransatto.herokuapp.com/pedidos Send 200 OK 929 ms 516 B

Body ▾ Auth ▾ Query Header Docs Preview ▾ Header 8 Cookie Timeline

```
1 ↑ {
2 ↑   "response": {
3 ↑     "pedidos": [
4 ↑       {
5         "id_pedido": 1,
6         "quantidade": 5,
7         "produto": {
8           "id_produto": 1,
9           "nome": "Ventilador",
10          "preco": 119.9
11        },
12        "request": {
13          "tipo": "GET",
14          "descricao": "Retorna os detalhes de um pedido específico",
15          "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos/1"
16        }
17      },
18      {
19        "id_pedido": 2,
20        "quantidade": 10,
21        "produto": {
22          "id_produto": 2,
23          "nome": "Liquidificador",
24          "preco": 84.3
25        },
26        "request": {
27          "tipo": "GET",
28          "descricao": "Retorna os detalhes de um pedido específico",
29          "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos/2"
30        }
31      }
32    ]
33  }
34 }
```

Select a body type from above



Aula 16 - Promise, Async/Await, TryCatch

Corrigindo o controller de produtos

controllers\produtos-controller.js

```
const mysql = require('../mysql').pool;

exports.getProdutos = (req, res, next) => {
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'SELECT * FROM produtos',
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) };
        const response = {
          quantidade: result.length,
          produtos: result.map(prod => {
            return{
              id: prod.id_produto,
              nome: prod.nome,
              preco: prod.preco,
              imagem_produto: prod.imagem_produto,
              request: {
                tipo: 'GET',
                descricao: 'Retorna os detalhes de um produto específico',
                url: process.env.URL_API + 'produtos/' + prod.id_produto
              }
            }
          })
        }
        return res.status(200).send({response});
      }
    )
  });
};

exports.postProduto = (req, res, next) => {
  console.log(req.file);
  const produto = {
    nome: req.body.nome,
    preco: req.body.preco
  };
  mysql.getConnection((error, conn) => {
    if(error){ return res.status(500).send({error: error}) }
    conn.query(
      'INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)',
      [req.body.nome, req.body.preco, req.file.path],
      (error, result, field) => {
        conn.release();
        if(error){ return res.status(500).send({error: error}) }
        const response = {
          mensagem: "Produto inserido com sucesso!",
          produtoCriado: {
            id: result.id_produto,
```

```

        nome: req.body.nome,
        preco: req.body.preco,
        imagem_produto: req.file.path,
        request: {
            tipo: 'POST',
            descricao: 'Retorna todos os produtos',
            url: process.env.URL_API + 'produtos'
        }
    }
}
return res.status(201).send(response);
}
}
});
};

exports.getUmProduto = (req, res, next) => {
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'SELECT * FROM produtos WHERE id_produto = ?',
[req.params.id],
(error, result, field) => {
if(error){ return res.status(500).send({error: error}) };
if(result.length == 0){
    return res.status(404).send({
        mensagem: "Não foi encontrado produto com este Id!"
    })
}
const response = {
    produto: {
        id: result[0].id_produto,
        nome: result[0].nome,
        preco: result[0].preco,
        imagem_produto: result[0].imagem_produto,
        request: {
            tipo: 'GET',
            descricao: 'Retorna todos os produtos',
            url: process.env.URL_API + 'produtos'
        }
    }
}
return res.status(200).send(response);
}
)
});
};

exports.updateProduto = (req, res, next) => {
var id = req.params.id;
mysql.getConnection((error, conn) => {
if(error){ return res.status(500).send({error: error}) }
conn.query(
'UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?',
[req.body.nome, req.body.preco, id],
(error, result, field) => {
    conn.release();
    if(error){ return res.status(500).send({error: error}) }
    const response = {

```

```

        mensagem: "Produto atualizado com sucesso!",
        produtoAtualizado: {
            id: id,
            nome: req.body.nome,
            preco: req.body.preco,
            request: {
                tipo: 'PATCH',
                descricao: 'Retorna os detalhes de um produto específico',
                url: process.env.URL_API + 'produtos/' + id
            }
        }
    }
    return res.status(200).send(response);
}
});

};

exports.deleteProduto = (req, res, next) => {
    mysql.getConnection((error, conn) => {
        if(error){ return res.status(500).send({error: error}) }
        conn.query(
            'DELETE FROM produtos WHERE id_produto = ?',
            [req.params.id],
            (error, result, field) => {
                conn.release();
                if(error){ return res.status(500).send({error: error}) }
                const response = {
                    mensagem: "Produto removido com sucesso!",
                    request: {
                        tipo: 'POST',
                        descricao: 'Insere um produto',
                        url: process.env.URL_API + 'produtos',
                        body: {
                            nome: 'String',
                            preco: 'Number'
                        }
                    }
                }
                return res.status(200).send(response);
            }
        );
    });
};

```

Utilizando Promises

mysql.js

```
const mysql = require('mysql2');

var pool = mysql.createPool({
  "user": process.env.MYSQL_USER,
  "password": process.env.MYSQL_PASSWORD,
  "database": process.env.MYSQL_DATABASE,
  "host": process.env.MYSQL_HOST,
  "port": process.env.MYSQL_PORT
});

exports.execute = (query, params=[]) => {
  return new Promise((resolve, reject) => {
    pool.getConnection((error, conn) => {
      if(error){
        reject(error);
      } else {
        conn.query(query, params, (error, result, fields) => {
          conn.release();
          if(error){
            reject(error);
          } else {
            resolve(result);
          }
        });
      }
    })
  })
}

exports.pool = pool;
```

controllers\produtos-controller.js

```
const mysql = require('../mysql');

exports.getProdutos = (req, res, next) => {
  mysql.execute("SELECT * FROM produtos;").then((result) => {
    const response = {
      quantidade: result.length,
      produtos: result.map(prod => {
        return{
          id: prod.id_produto,
          nome: prod.nome,
          preco: prod.preco,
          imagem_produto: prod.imagem_produto,
          request: {
            tipo: 'GET',
            descricao: 'Retorna os detalhes de um produto específico',
            url: process.env.URL_API + 'produtos/' + prod.id_produto
          }
        }
      })
    }
  })
}
```

```
        }
        return res.status(200).send({response});
    }).catch((error) => {
        return res.status(500).send({error: error});
    });
};
```

- No Insomnia:

localhost:3000/produtos

GET	localhost:3000/produtos	Send
Body	Auth Query Header Docs	200 OK 3.24 s 427 B
<p>Preview ▾</p> <p>Header 7</p> <p>Cookie</p> <p>Timeline</p> <pre>1 v { 2 v "response": { 3 v "quantidade": 2, 4 v "produtos": [5 v { 6 v "id": 1, 7 v "nome": "Ventilador", 8 v "preco": 119.9, 9 v "imagem_produto": null, 10 v "request": { 11 v "tipo": "GET", 12 v "descricao": "Retorna os detalhes de um produto específico", 13 v "url": "http://localhost:3000/produtos/1" 14 v } 15 v }, 16 v { 17 v "id": 2, 18 v "nome": "Liquidificador", 19 v "preco": 84.3, 20 v "imagem_produto": null, 21 v "request": { 22 v "tipo": "GET", 23 v "descricao": "Retorna os detalhes de um produto específico", 24 v "url": "http://localhost:3000/produtos/2" 25 v } 26 v } 27 v] 28 v } 29 }</pre>		
Select a body type from above		

Utilizando async/await

mysql.js

```
const mysql = require('mysql2');
pool = mysql.createPool({
  "connectionLimit": 1000,
  "user": process.env.MYSQL_USER,
  "password": process.env.MYSQL_PASSWORD,
  "database": process.env.MYSQL_DATABASE,
  "host": process.env.MYSQL_HOST,
  "port": process.env.MYSQL_PORT
});

exports.execute = (query, params=[]) => {
  return new Promise((resolve, reject) => {
    pool.query(query, params, (error, result, fields) => {
      if (error) {
        reject(error);
      } else {
        resolve(result)
      }
    });
  })
}

exports.pool = pool;
```

controllers\produtos-controller.js

```
const mysql = require('../mysql');

exports.getProdutos = async (req, res, next) => {
  try {
    const result = await mysql.execute("SELECT * FROM produtos;");
    const response = {
      quantidade: result.length,
      produtos: result.map(prod => {
        return{
          id_produto: prod.id_produto,
          nome: prod.nome,
          preco: prod.preco,
          imagem_produto: prod.imagem_produto,
          request: {
            tipo: 'GET',
            descricao: 'Retorna os detalhes de um produto específico',
            url: process.env.URL_API + 'produtos/' + prod.id_produto
          }
        }
      })
    }
    return res.status(200).send({response});
  } catch (error) {
    return res.status(500).send({error: error});
  }
};

exports.postProduto = async (req, res, next) => {
```

```

try {
  const query = "INSERT INTO produtos (nome, preco, imagem_produto) VALUES (?, ?, ?)";
  const result = await mysql.execute(query, [
    req.body.nome,
    req.body.preco,
    req.file.path
  ]);
  const response = {
    mensagem: "Produto inserido com sucesso!",
    produtoCriado: {
      id_produto: result.id_produto,
      nome: req.body.nome,
      preco: req.body.preco,
      imagem_produto: req.file.path,
      request: {
        tipo: 'POST',
        descricao: 'Retorna todos os produtos',
        url: process.env.URL_API + 'produtos'
      }
    }
  }
  return res.status(201).send(response);
} catch (error) {
  return res.status(500).send({error: error});
}
};


```

```

exports.getUmProduto = async (req, res, next) => {
  try {
    const query = "SELECT * FROM produtos WHERE id_produto = ?";
    const result = await mysql.execute(query, [
      req.params.id
    ]);
    const response = {
      produto: {
        id_produto: result[0].id_produto,
        nome: result[0].nome,
        preco: result[0].preco,
        imagem_produto: result[0].imagem_produto,
        request: {
          tipo: 'GET',
          descricao: 'Retorna todos os produtos',
          url: process.env.URL_API + 'produtos'
        }
      }
    }
    return res.status(200).send(response);
  } catch (error) {
    return res.status(500).send({error: error});
  }
};


```

```

exports.updateProduto = async (req, res, next) => {
  try {
    const query = "UPDATE produtos SET nome = ?, preco = ? WHERE id_produto = ?";
    await mysql.execute(query, [
      req.body.nome,
      req.body.preco,
      req.params.id
    ])
  }
};


```

```

]);
const response = {
  mensagem: 'Produto atualizado com sucesso',
  produtoAtualizado: {
    id_produto: req.params.id,
    nome: req.body.nome,
    preco: req.body.preco,
    request: {
      type: 'GET',
      description: 'Retorna os detalhes de um produto específico',
      url: process.env.URL_API + 'produtos/' + req.params.id
    }
  }
}
return res.status(202).send(response);
} catch (error) {
  return res.status(500).send({ error: error });
}
};

exports.deleteProduto = async (req, res, next) => {
try {
  const query = "DELETE FROM produtos WHERE id_produto = ?";
  await mysql.execute(query, [
    [req.params.id]
  ]);
  const response = {
    mensagem: "Produto removido com sucesso!",
    request: {
      tipo: 'POST',
      descricao: 'Insere um produto',
      url: process.env.URL_API + 'produtos',
      body: {
        nome: 'String',
        preco: 'Number'
      }
    }
  }
  return res.status(200).send(response);
} catch (error) {
  return res.status(500).send({error: error});
}
};

```

controllers\pedidos-controller.js

```
const mysql = require('../mysql');

exports.getPedidos = async (req, res, next) => {

  try {
    const query = `select
      pedidos.id_pedido,
      pedidos.quantidade,
      produtos.id_produto,
      produtos.nome,
      produtos.preco
    from pedidos
    inner join produtos
    on produtos.id_produto = pedidos.id_produto`;
    const result = await mysql.execute(query);
    const response = {
      pedidos: result.map(pedido => {
        return{
          id_pedido: pedido.id_pedido,
          quantidade: pedido.quantidade,
          produto: {
            id_produto: pedido.id_produto,
            nome: pedido.nome,
            preco: pedido.preco
          },
          request: {
            tipo: 'GET',
            descricao: 'Retorna os detalhes de um pedido específico',
            url: process.env.URL_API + 'pedidos/' + pedido.id_pedido
          }
        }
      })
    }
    return res.status(200).send({response});
  } catch (error) {
    return res.status(500).send({error: error});
  }
};

exports.postPedido = async (req, res, next) => {
  try {
    const queryProduto = 'SELECT * FROM produtos WHERE id_produto = ?';
    const resultProduto = await mysql.execute(queryProduto, [req.body.id_produto]);

    if (resultProduto.length == 0) {
      return res.status(404).send({ message: 'Produto não encontrado'});
    }

    const queryPedido = 'INSERT INTO pedidos (id_produto, quantidade) VALUES (?,?)';
    const resultPedido = await mysql.execute(queryPedido, [
      req.body.id_produto,
      req.body.quantidade
    ]);

    const response = {
      mensagem: 'Pedido inserido com sucesso',
```

```

pedidoCriado: {
    id_pedido: resultPedido.insertId,
    id_produto: req.body.id_produto,
    quantidade: req.body.quantidade,
    request: {
        tipo: 'GET',
        descrição: 'Retorna todos os pedidos',
        url: process.env.URL_API + 'pedidos'
    }
}
}

return res.status(201).send(response);

} catch (error) {
    return res.status(500).send({ error: error });
}
};

exports.getUmPedido = async (req, res, next) => {
try {
    const query = "SELECT * FROM pedidos WHERE id_pedido = ?";
    const result = await mysql.execute(query, [req.params.id]);
    const response = {
        pedido: {
            id_pedido: result[0].id_pedido,
            quantidade: result[0].quantidade,
            id_produto: result[0].id_produto,
            request: {
                tipo: 'GET',
                descrição: 'Retorna todos os pedidos',
                url: process.env.URL_API + 'pedidos'
            }
        }
    }
    return res.status(200).send(response);
} catch (error) {
    return res.status(500).send({error: error});
}
};

exports.deletePedido = async (req, res, next) => {
try {
    const query = "DELETE FROM pedidos WHERE id_pedido = ?";
    const result = await mysql.execute(query, [req.params.id]);
    const response = {
        mensagem: "Pedido removido com sucesso!",
        request: {
            tipo: 'POST',
            descrição: 'Insere um pedido',
            url: process.env.URL_API + 'pedidos',
            body: {
                id_produto: 'Number',
                quantidade: 'Number'
            }
        }
    }
    return res.status(200).send(response);
} catch (error) {
    return res.status(500).send({error: error});
}
};

```

```
    }  
};
```

- Apague todos os registros da tabela pedidos:

```
truncate pedidos;
```

The screenshot shows a SQL interface with two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs: truncate pedidos;'. The second tab contains the query 'truncate pedidos;'. A green status bar at the bottom right indicates 'Query returned 0 rows'.

- Apague todos os registros da tabela produtos:

```
SET FOREIGN_KEY_CHECKS = 0;  
TRUNCATE produtos;  
SET FOREIGN_KEY_CHECKS = 1;
```

The screenshot shows a SQL interface with two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs: SET FOREIGN_KEY_CHECKS = 0;'. The second tab contains the query 'SET FOREIGN_KEY_CHECKS = 0;'. A green status bar at the bottom right indicates 'Query returned 0 rows'.

The screenshot shows a SQL interface with two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs: SET FOREIGN_KEY_CHECKS = 1;'. The second tab contains the query 'TRUNCATE produtos;'. A green status bar at the bottom right indicates 'Query returned 0 rows'.

The screenshot shows a SQL interface with two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs: SET FOREIGN_KEY_CHECKS = 1;'. The second tab contains the query 'SET FOREIGN_KEY_CHECKS = 1;'. A green status bar at the bottom right indicates 'Query returned 0 rows'.

routes\produtos.js

```
const express = require('express');
const router = express.Router();
const multer = require('multer');
const login = require('../middleware/login');

const ProdutosController = require('../controllers/produtos-controller');

const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, './uploads/');
  },
  filename: function( req, file, cb ){
    let data = new Date().toISOString().replace(/:/g, '-') + '-';
    cb(null, data + file.originalname );
  }
});

const fileFilter = (req, file, cb) => {
  if(file.mimetype === 'image/jpeg' || file.mimetype === 'image/png'){
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

router.get('/', ProdutosController.getProdutos);
router.post('/', 
  upload.single('produto_imagem'),
  login,
  ProdutosController.postProduto
);
router.get('/:id', ProdutosController.getUmProduto);
router.patch('/:id', login, ProdutosController.updateProduto);
router.delete('/:id', login, ProdutosController.deleteProduto);

module.exports = router;
```

- No Insomnia, cadastre 4 produtos:

```

POST ▾ localhost:3000/usuarios/login
Send
200 OK 4.95 s 251 B Just Now ▾
JSON ▾ Auth ▾ Query Header [1] Docs
Preview ▾ Header [7] Cookie Timeline
1 v {
2   "email": "fernando.maransatto@gmail.com",
3   "senha": "123456"
4 }
1 v {
2   "mensagem": "Autenticado com sucesso!",
3   "token":
4     "eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZF91c3VhcmvljoxLCJlbWFpbCI6ImZlcm5hbmRvLm1hcmFuc2F0dG9AZ21h
5     aWwuY29tliwiaWF0ljoxNjlxNzc4Njg0LCJleHAIoJE2MjE3OTMwODR9.LdfinBbtk2IWO_1hRZtqEAUtFMPuboV7PFnuKKez
6     wc8"
7
}
  
```

```
{
  "mensagem": "Autenticado com sucesso!",
  "token":
  "eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZF91c3VhcmvljoxLCJlbWFpbCI6ImZlcm5hbmRvLm1hcmFuc2F0dG9AZ21h
  aWwuY29tliwiaWF0ljoxNjlxNzc4Njg0LCJleHAIoJE2MjE3OTMwODR9.LdfinBbtk2IWO_1hRZtqEAUtFMPuboV7PFnuKKez
  wc8"
}
```

```

POST ▾ localhost:3000/produtos
Send
201 Created 765 ms 269 B
Multipart [3] ▾ Bearer ▾ Query Header [2] Docs
Preview ▾ Header [7] Cookie Timeline
1 v {
2   "mensagem": "Produto inserido com sucesso!",
3   "produtoCriado": [
4     {
5       "nome": "Ventilador",
6       "preco": "120",
7       "imagem_produto": "uploads\\2021-05-23T13-18-094Z-ventilador.jpg",
8       "request": {
9         "tipo": "POST",
10        "descricao": "Retorna todos os produtos",
11        "url": "http://localhost:3000/produtos"
12      }
13    }
  
```

```

POST ▾ localhost:3000/produtos
Send
201 Created 2.32 s 276 B
Multipart [3] ▾ Bearer ▾ Query Header [2] Docs
Preview ▾ Header [7] Cookie Timeline
1 v {
2   "mensagem": "Produto inserido com sucesso!",
3   "produtoCriado": [
4     {
5       "nome": "Liquidificador",
6       "preco": "80",
7       "imagem_produto": "uploads\\2021-05-23T13-19-28.886Z-liquidificador.jpg",
8       "request": {
9         "tipo": "POST",
10        "descricao": "Retorna todos os produtos",
11        "url": "http://localhost:3000/produtos"
12      }
13    }
  
```

```

POST ▾ localhost:3000/produtos
Send
201 Created 802 ms 275 B
Multipart [3] ▾ Bearer ▾ Query Header [2] Docs
Preview ▾ Header [7] Cookie Timeline
1 v {
2   "mensagem": "Produto inserido com sucesso!",
3   "produtoCriado": [
4     {
5       "nome": "Rádio Relógio",
6       "preco": "6",
7       "imagem_produto": "uploads\\2021-05-23T13-21-06.886Z-radio_relogio.jpg",
8       "request": {
9         "tipo": "POST",
10        "descricao": "Retorna todos os produtos",
11        "url": "http://localhost:3000/produtos"
12      }
13    }
  
```

POST <localhost:3000/produtos>

Multipart 3	Bearer	Query	Header 2	Docs	Send	201 Created	566 ms	266 B	
<input checked="" type="checkbox"/> nome	Batedeira	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		Preview	Header 7	Cookie	Timeline
<input checked="" type="checkbox"/> preco	90	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
<input checked="" type="checkbox"/> produto_imagem	<input checked="" type="checkbox"/> batedeirajpg	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
<input checked="" type="checkbox"/> name	value	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
<input checked="" type="checkbox"/> New name	New value								

```

1 + {
2   "mensagem": "Produto inserido com sucesso!",
3   "produtoCriado": {
4     "nome": "Batedeira",
5     "preco": "90",
6     "imagem_produto": "uploads\\2021-05-23T13-22-11.177Z-batedeira.jpg",
7   }
8   "request": {
9     "tipo": "POST",
10    "descricao": "Retorna todos os produtos",
11    "url": "http://localhost:3000/produtos"
12  }
13 }

```

- Liste todos os produtos cadastrados:

GET <localhost:3000/produtos>

Body	Auth	Query	Header	Docs	Send	200 OK	491 ms	1025 B	
						Preview	Header 7	Cookie	Timeline
Select a body type from above									

```

1 + {
2   "response": {
3     "quantidade": 4,
4     "produtos": [
5       {
6         "id_produto": 1,
7         "nome": "Ventilador",
8         "preco": 120,
9         "imagem_produto": "uploads\\2021-05-23T14-07-03.853Z-ventilador.jpg",
10        "request": {
11          "tipo": "GET",
12          "descricao": "Retorna os detalhes de um produto específico",
13          "url": "http://localhost:3000/produtos/1"
14        }
15      },
16      {
17        "id_produto": 2,
18        "nome": "Liquificador",
19        "preco": 88,
20        "imagem_produto": "uploads\\2021-05-23T14-08-03.711Z-liquidificador.jpg",
21        "request": {
22          "tipo": "GET",
23          "descricao": "Retorna os detalhes de um produto específico",
24          "url": "http://localhost:3000/produtos/2"
25        }
26      },
27      {
28        "id_produto": 3,
29        "nome": "Rádio Relógio",
30        "preco": 60,
31        "imagem_produto": "uploads\\2021-05-23T14-09-00.470Z-radio_relogio.jpg",
32        "request": {
33          "tipo": "GET",
34          "descricao": "Retorna os detalhes de um produto específico",
35          "url": "http://localhost:3000/produtos/3"
36        }
37      },
38      {
39        "id_produto": 4,
40        "nome": "Batedeira",
41        "preco": 90,
42        "imagem_produto": "uploads\\2021-05-23T14-09-29.094Z-batedeira.jpg",
43        "request": {
44          "tipo": "GET",
45          "descricao": "Retorna os detalhes de um produto específico",
46          "url": "http://localhost:3000/produtos/4"
47        }
48      }
49    ]
50  }
51 }

```

- Exiba os detalhes do produto de id 4:

GET <localhost:3000/produtos/4>

Body	Auth	Query	Header	Docs	Send	200 OK	605 ms	229 B	
						Preview	Header 7	Cookie	Timeline

```

1 + {
2   "produto": {
3     "id_produto": 4,
4     "nome": "Batedeira",
5     "preco": 90,
6     "imagem_produto": "uploads\\2021-05-23T14-09-29.094Z-batedeira.jpg",
7   }
8   "request": {
9     "tipo": "GET",
10    "descricao": "Retorna todos os produtos",
11    "url": "http://localhost:3000/produtos"
12  }
13 }

```

- Altere o preço da batedeira de 90 para 80:

PATCH	localhost:3000/produtos/4	Send	202 Accepted	1.63 s	242 B
JSON	Bearer	Query	Header	Docs	Preview
1 = { 2 "nome": "Batedeira", 3 "preco": 80 4 }]			1 = { 2 "mensagem": "Produto atualizado com sucesso", 3 "produtoAtualizado": { 4 "id_produto": "4", 5 "nome": "Batedeira", 6 "preco": 80, 7 "request": { 8 "type": "GET", 9 "description": "Retorna os detalhes de um produto específico", 10 "url": "http://localhost:3000/produtos/4" 11 } 12 } 13 }		

- Exclua esse produto:

DELETE	localhost:3000/produtos/4	Send	200 OK	999 ms	183 B
Body	Bearer	Query	Header	Docs	Preview
TOKEN	eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZF91c3Vhcm1vljoxLCJlbWFj				Header
PREFIX	eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZF91c3Vhcm1vljoxLCJlbWFj				Cookie
ENABLED	<input checked="" type="checkbox"/>				Timeline

routes\pedidos.js

```
const express = require('express');
const router = express.Router();
const login = require('../middleware/login');

const PedidosController = require('../controllers/pedidos-controller');

router.get('/', PedidosController.getPedidos);
router.post('/', PedidosController.postPedido);
router.get('/:id', PedidosController.getUmPedido);
router.delete('/:id', login, PedidosController.deletePedido);

module.exports = router;
```

- No Insomnia, cadastre 2 pedidos:

POST ▾	localhost:3000/pedidos				Send	201 Created	10.9 s	207 B
JSON ▾	Bearer ▾	Query	Header 1	Docs	Preview ▾	Header 7	Cookie	Timeline
<pre>1 ▶ { 2 "id_produto": 1, 3 "quantidade": 2 4 }</pre>					<pre>1 ▶ { 2 "mensagem": "Pedido inserido com sucesso", 3 "pedidoCriado": { 4 "id_pedido": 1, 5 "id_produto": 1, 6 "quantidade": 2, 7 "request": { 8 "tipo": "GET", 9 "descrição": "Retorna todos os pedidos", 10 "url": "http://localhost:3000/pedidos" 11 } 12 } 13 }</pre>			

POST ▾	localhost:3000/pedidos				Send	201 Created	2.14 s	208 B
JSON ▾	Bearer ▾	Query	Header 1	Docs	Preview ▾	Header 7	Cookie	Timeline
<pre>1 ▶ { 2 "id_produto": 2, 3 "quantidade": 10 4 }</pre>					<pre>1 ▶ { 2 "mensagem": "Pedido inserido com sucesso", 3 "pedidoCriado": { 4 "id_pedido": 2, 5 "id_produto": 2, 6 "quantidade": 10, 7 "request": { 8 "tipo": "GET", 9 "descrição": "Retorna todos os pedidos", 10 "url": "http://localhost:3000/pedidos" 11 } 12 } 13 }</pre>			

- Liste todos os pedidos:

GET ▾	localhost:3000/pedidos	Send	200 OK	1.65 s	458 B			
Body ▾	Auth ▾	Query	Header	Docs	Preview ▾	Header [7]	Cookie	Timeline
 Select a body type from above					1 + { 2 + "response": { 3 + "pedidos": [4 + { 5 + "id_pedido": 1, 6 + "quantidade": 2, 7 + "produto": { 8 + "id_produto": 1, 9 + "nome": "Ventilador", 10 + "preco": 120 11 + }, 12 + "request": { 13 + "tipo": "GET", 14 + "descricao": "Retorna os detalhes de um pedido específico", 15 + "url": "http://localhost:3000/pedidos/1" 16 + }, 17 + }, 18 + { 19 + "id_pedido": 2, 20 + "quantidade": 10, 21 + "produto": { 22 + "id_produto": 2, 23 + "nome": "Liquidificador", 24 + "preco": 80 25 + }, 26 + "request": { 27 + "tipo": "GET", 28 + "descricao": "Retorna os detalhes de um pedido específico", 29 + "url": "http://localhost:3000/pedidos/2" 30 + }, 31 + } 32 +] 33 + } 34 +}			

- Exiba os detalhes do pedido 2:

GET ▾	localhost:3000/pedidos/2	Send	200 OK	624 ms	159 B			
Body ▾	Auth ▾	Query	Header	Docs	Preview ▾	Header [7]	Cookie	Timeline
					1 + { 2 + "pedido": { 3 + "id_pedido": 2, 4 + "quantidade": 10, 5 + "id_produto": 2, 6 + "request": { 7 + "tipo": "GET", 8 + "descricao": "Retorna todos os pedidos", 9 + "url": "http://localhost:3000/pedidos" 10 + }, 11 + } 12 + }			

- Exclua o pedido de id 2:

DELETE ▾	localhost:3000/pedidos/2	Send	200 OK	2.95 s	191 B					
Body ▾	Bearer ▾	Query	Header	Docs	Preview ▾	Header [7]	Cookie	Timeline		
TOKEN	vJxE3OTMwODR9.LdfinBbtk2IWO_IhRZtqEAUtfMPuboV7PFnuKKewc8						1 + { 2 + "mensagem": "Pedido removido com sucesso!", 3 + "request": { 4 + "tipo": "POST", 5 + "descricao": "Insere um pedido", 6 + "url": "http://localhost:3000/pedidos", 7 + "body": { 8 + "id_produto": "Number", 9 + "quantidade": "Number" 10 + } 11 + } 12 + }			
PREFIX	②									
ENABLED	<input checked="" type="checkbox"/>									

- Faça o commit e envie as alterações para o GitHub:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   api_rest_nodejs.session.sql
    modified:   controllers/pedidos-controller.js
    modified:   controllers/produtos-controller.js
    modified:   controllers/usuarios-controller.js
    modified:   mysql.js
    modified:   routes/pedidos.js
    modified:   routes/produtos.js

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    uploads/

no changes added to commit (use "git add" and/or "git commit -a")
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git add .
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   api_rest_nodejs.session.sql
    modified:   controllers/pedidos-controller.js
    modified:   controllers/produtos-controller.js
    modified:   controllers/usuarios-controller.js
    modified:   mysql.js
    modified:   routes/pedidos.js
    modified:   routes/produtos.js
    new file:  uploads/2021-05-23T14-07-03.853Z-ventilador.jpg
    new file:  uploads/2021-05-23T14-08-03.711Z-liquidificador.jpg
    new file:  uploads/2021-05-23T14-09-00.470Z-radio_relogio.jpg
    new file:  uploads/2021-05-23T14-09-29.094Z-batedeira.jpg
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git commit -m "Alterado controllers de produtos e pedidos para padrão async/await e alterado o tempo de expiração do token de 1h para 4h no controller de usuários"
[master bf50fa0] Alterado controllers de produtos e pedidos para padrão async/await e alterado o tempo de expiração do token de 1h para 4h no controller de usuários
11 files changed, 272 insertions(+), 304 deletions(-)
rewrite controllers/pedidos-controller.js (92%)
rewrite controllers/produtos-controller.js (96%)
create mode 100644 uploads/2021-05-23T14-07-03.853Z-ventilador.jpg
create mode 100644 uploads/2021-05-23T14-08-03.711Z-liquidificador.jpg
create mode 100644 uploads/2021-05-23T14-09-00.470Z-radio_relogio.jpg
create mode 100644 uploads/2021-05-23T14-09-29.094Z-batedeira.jpg
```

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/maransatto/rest (master)
$ git push heroku master
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 2 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (16/16), 135.47 KiB | 4.10 MiB/s, done.
Total 16 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: ----> Building on the Heroku-20 stack
remote: ----> Using buildpack: heroku/nodejs
remote: ----> Node.js app detected
remote:
remote: ----> Creating runtime environment
remote:
remote:      NPM_CONFIG_LOGLEVEL=error
remote:      NODE_VERBOSE=false
remote:      NODE_ENV=production
remote:      NODE_MODULES_CACHE=true
```

← → ⌂ 🔒 rest-api-nodejs-maransatto.herokuapp.com/produtos

_apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstr...

```
1 // 20210523131948
2 // https://rest-api-nodejs-maransatto.herokuapp.com/produtos
3
4 +
5     "response": {
6         "quantidade": 3,
7         "produtos": [
8             {
9                 "id_produto": 1,
10                "nome": "Ventilador",
11                "preco": 120,
12                "imagem_produto": "uploads\\2021-05-23T14-07-03.853Z-ventilador.jpg",
13                "request": {
14                    "tipo": "GET",
15                    "descricao": "Retorna os detalhes de um produto específico",
16                    "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos/1"
17                }
18            },
19            {
20                "id_produto": 2,
21                "nome": "Liquidificador",
22                "preco": 80,
23                "imagem_produto": "uploads\\2021-05-23T14-08-03.711Z-liquidificador.jpg",
24                "request": {
25                    "tipo": "GET",
26                    "descricao": "Retorna os detalhes de um produto específico",
27                    "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos/2"
28                }
29            },
30            {
31                "id_produto": 3,
32                "nome": "Rádio Relógio",
33                "preco": 60,
34                "imagem_produto": "uploads\\2021-05-23T14-09-00.470Z-radio_relogio.jpg",
35                "request": {
36                    "tipo": "GET",
37                    "descricao": "Retorna os detalhes de um produto específico",
38                    "url": "https://rest-api-nodejs-maransatto.herokuapp.com/produtos/3"
39                }
40            }
41        ]
42    }
43 }
```

← → ⌂ 🔒 rest-api-nodejs-maransatto.herokuapp.com/pedidos

_apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstr...

```
1 // 20210523132053
2 // https://rest-api-nodejs-maransatto.herokuapp.com/pedidos
3
4 +
5     "response": {
6         "pedidos": [
7             {
8                 "id_pedido": 1,
9                 "quantidade": 2,
10                "produto": {
11                    "id_produto": 1,
12                    "nome": "Ventilador",
13                    "preco": 120
14                },
15                "request": {
16                    "tipo": "GET",
17                    "descricao": "Retorna os detalhes de um pedido específico",
18                    "url": "https://rest-api-nodejs-maransatto.herokuapp.com/pedidos/1"
19                }
20            ]
21        ]
22    }
23 }
```



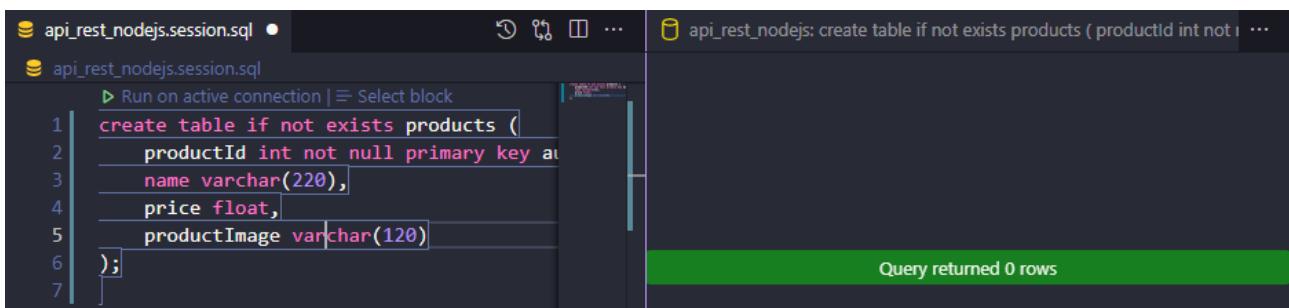
Aula 18 - Refactoring para Inglês

- Inicialmente, apague as três tabelas

Recriando as tabelas

Tabela products

```
create table if not exists products (
    productId int not null primary key auto_increment,
    name varchar(220),
    price float,
    productImage varchar(120)
);
```

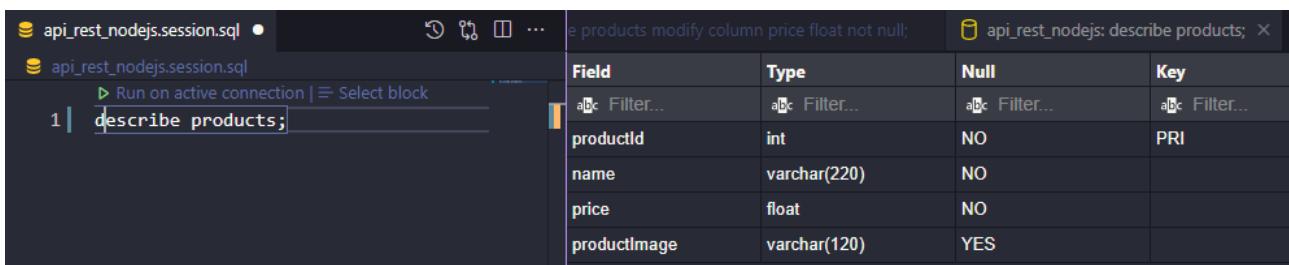


The screenshot shows a SQL editor interface with two panes. The left pane contains the SQL code for creating the 'products' table. The right pane shows the results of the query, which returned 0 rows.

```
api_rest_nodejs.session.sql | Run on active connection | Select block
1 | create table if not exists products (
2 |     productId int not null primary key auto_increment,
3 |     name varchar(220),
4 |     price float,
5 |     productImage varchar(120)
6 | );
7 | 
```

Query returned 0 rows

```
alter table products modify column name varchar(220) not null;
alter table products modify column price float not null;
describe products;
```



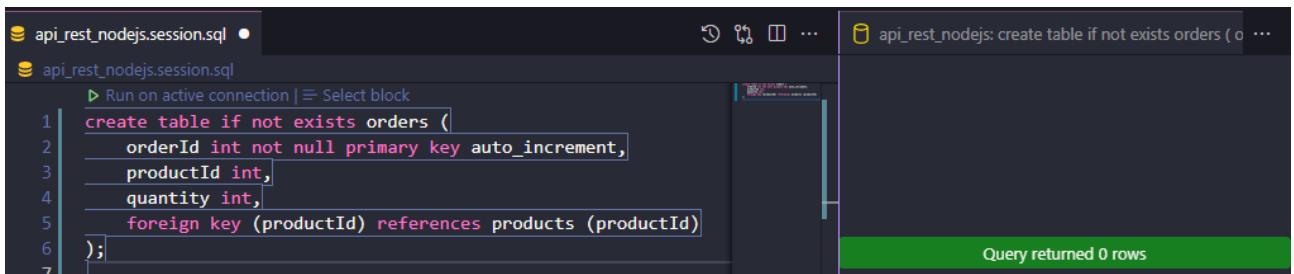
The screenshot shows a SQL editor interface with two panes. The left pane contains the SQL code for modifying the 'products' table and describing it. The right pane shows the results of the 'describe products' command, displaying the table structure.

```
api_rest_nodejs.session.sql | Run on active connection | Select block
1 | describe products;
```

Field	Type	Null	Key
productId	int	NO	PRI
name	varchar(220)	NO	
price	float	NO	
productImage	varchar(120)	YES	

Tabela orders

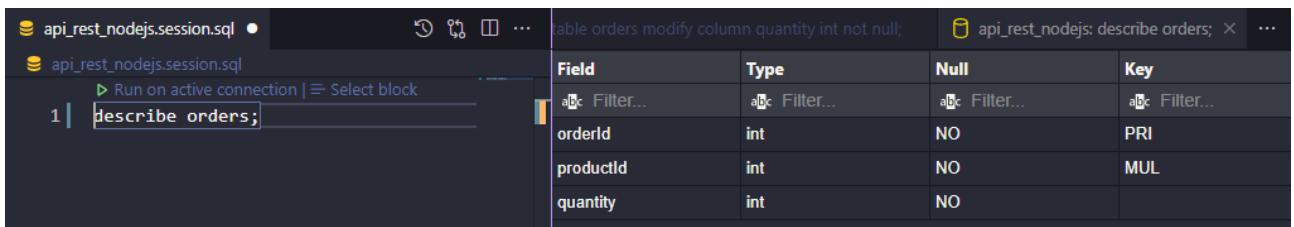
```
create table if not exists orders (
    orderId int not null primary key auto_increment,
    productId int,
    quantity int,
    foreign key (productId) references products (productId)
);
```



A screenshot of a SQL editor interface. On the left, there are two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs.session.sql'. The main area contains the SQL code for creating the 'orders' table. A tooltip 'Run on active connection | Select block' is visible above the code. The status bar at the bottom right shows 'Query returned 0 rows'.

```
1 create table if not exists orders (
2     orderId int not null primary key auto_increment,
3     productId int,
4     quantity int,
5     foreign key (productId) references products (productId)
6 );
7
```

```
alter table orders modify column productId int not null;
alter table orders modify column quantity int not null;
```

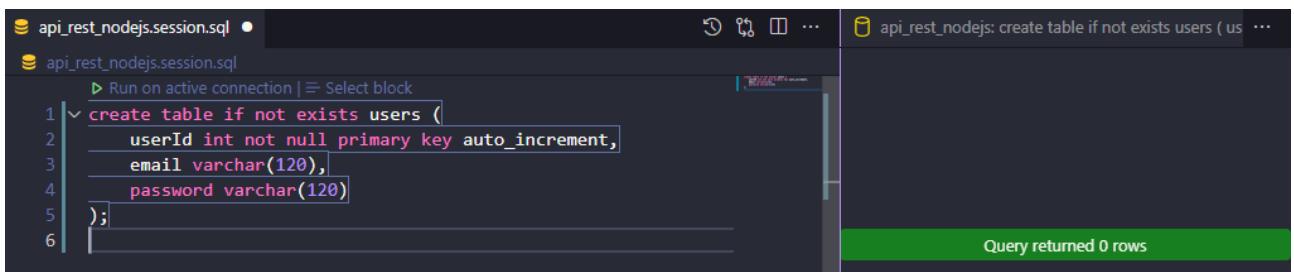


A screenshot of a SQL editor interface. On the left, there are two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs.session.sql'. The main area contains the SQL code for modifying the 'orders' table. A tooltip 'Run on active connection | Select block' is visible above the code. To the right, there is a table titled 'table orders modify column quantity int not null;' showing the structure of the 'orders' table. The table has four columns: 'Field', 'Type', 'Null', and 'Key'. The data is as follows:

Field	Type	Null	Key
orderId	int	NO	PRI
productId	int	NO	MUL
quantity	int	NO	

Tabela users

```
create table if not exists users (
    userId int not null primary key auto_increment,
    email varchar(120),
    password varchar(120)
);
```

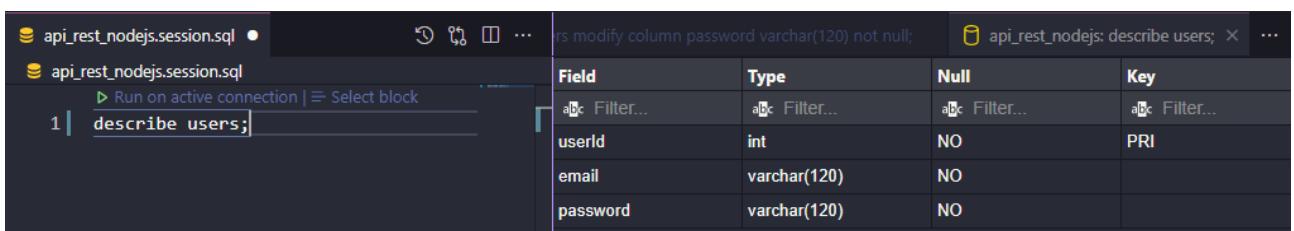


The screenshot shows a SQL editor interface with two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs.session.sql'. The left tab contains the SQL code for creating the 'users' table. The right tab shows the results of the query, which returned 0 rows.

```
1 | create table if not exists users (
2 |     userId int not null primary key auto_increment,
3 |     email varchar(120),
4 |     password varchar(120)
5 | );
6 | 
```

Query returned 0 rows

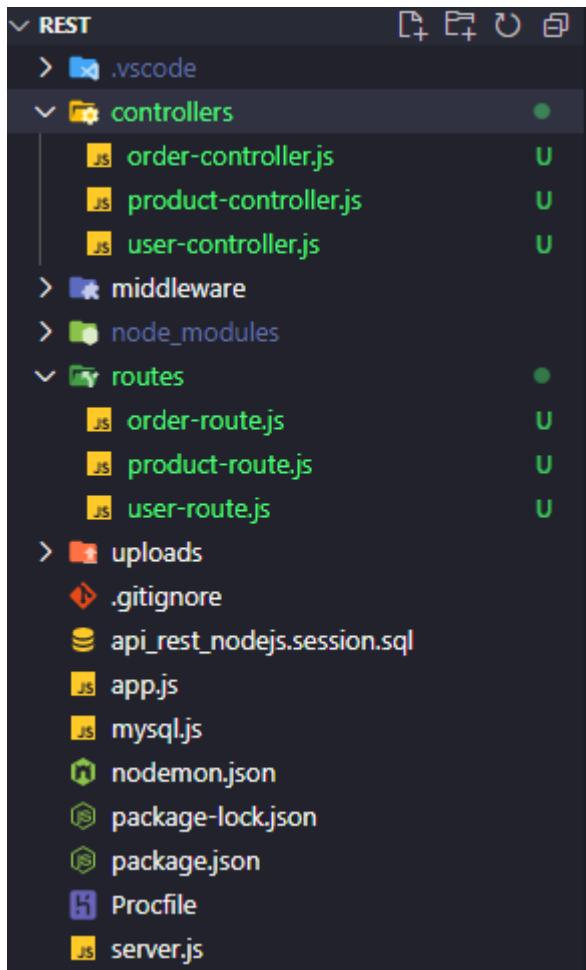
```
alter table users modify column email varchar(120) not null;
alter table users modify column password varchar(120) not null;
describe users;
```



The screenshot shows a SQL editor interface with two tabs: 'api_rest_nodejs.session.sql' and 'api_rest_nodejs.session.sql'. The left tab contains the SQL code for modifying the 'users' table and describing it. The right tab shows the results of the 'describe users;' command, displaying the table structure with three columns: 'userId' (int, primary key, not null), 'email' (varchar(120)), and 'password' (varchar(120)).

Field	Type	Null	Key
userId	int	NO	PRI
email	varchar(120)	NO	
password	varchar(120)	NO	

Alterando o nome de pastas e arquivos



Alterando o controller de produtos

controllers\product-controller.js

```
const mysql = require('../mysql');

exports.getProducts = async (req, res, next) => {
  try {
    const result = await mysql.execute("SELECT * FROM products;")
    const response = {
      quantity: result.length,
      products: result.map(product => {
        return{
          productId: product.productId,
          name: product.name,
          price: product.price,
          productImage: product.productImage,
          request: {
            type: 'GET',
            description: 'Retorna os detalhes de um produto específico',
            url: process.env.URL_API + 'products/' + product.productId
          }
        }
      })
      res.json(response)
    }
  } catch (err) {
    next(err)
  }
}
```

```

        }
    })
}
return res.status(200).send({response});
} catch (error) {
    return res.status(500).send({error: error});
}
};

exports.postProduct = async (req, res, next) => {
try {
    const query = "INSERT INTO products (name, price, productImage) VALUES (?, ?, ?)";
    const result = await mysql.execute(query, [
        req.body.name,
        req.body.price,
        req.file.path
    ]);
    const response = {
        message: "Produto inserido com sucesso!",
        createdProduct: {
            productId: result.insertId,
            name: req.body.name,
            price: req.body.price,
            productImage: req.file.path,
            request: {
                type: 'POST',
                description: 'Retorna todos os produtos',
                url: process.env.URL_API + 'products'
            }
        }
    }
    return res.status(201).send(response);
} catch (error) {
    return res.status(500).send({error: error});
}
};

exports.getProductDetail = async (req, res, next) => {
try {
    const query = "SELECT * FROM products WHERE productId = ?";
    const result = await mysql.execute(query, [
        req.params.productId
    ]);
    const response = {
        product: {
            productId: result[0].productId,
            name: result[0].name,
            price: result[0].price,
            productImage: result[0].productImage,
            request: {
                type: 'GET',
                description: 'Retorna todos os produtos',
                url: process.env.URL_API + 'products'
            }
        }
    }
    return res.status(200).send(response);
} catch (error) {
    return res.status(500).send({error: error});
}
};

```

```

        }
    };

exports.updateProduct = async (req, res, next) => {
    try {
        const query = "UPDATE products SET name = ?, price = ? WHERE productId = ?";
        await mysql.execute(query, [
            req.body.name,
            req.body.price,
            req.params.productId
        ]);
        const response = {
            message: 'Produto atualizado com sucesso',
            updatedProduct: {
                productId: req.params.productId,
                name: req.body.nome,
                price: req.body.preco,
                request: {
                    type: 'GET',
                    description: 'Retorna os detalhes de um produto específico',
                    url: process.env.URL_API + 'products/' + req.params.productId
                }
            }
        }
        return res.status(202).send(response);
    } catch (error) {
        return res.status(500).send({ error: error });
    }
};

exports.deleteProduct = async (req, res, next) => {
    try {
        const query = "DELETE FROM products WHERE productId = ?";
        await mysql.execute(query, [
            [req.params.productId]
        ]);
        const response = {
            message: "Produto removido com sucesso!",
            request: {
                type: 'POST',
                description: 'Insere um produto',
                url: process.env.URL_API + 'products',
                body: {
                    name: 'String',
                    price: 'Number'
                }
            }
        }
        return res.status(200).send(response);
    } catch (error) {
        return res.status(500).send({error: error});
    }
};

```

Alterando o controller de pedidos

controllers\order-controller.js

```
const mysql = require('../mysql');

exports.getOrders = async (req, res, next) => {

  try {
    const query = `select
      orders.orderId,
      orders.quantity,
      orders.productId,
      products.name,
      products.price
    from orders
    inner join products
    on products.productId = orders.productId`;
    const result = await mysql.execute(query);
    const response = {
      orders: result.map(order => {
        return{
          orderId: order.orderId,
          quantity: order.quantity,
          product: {
            productId: order.productId,
            name: order.name,
            price: order.price
          },
          request: {
            type: 'GET',
            description: 'Retorna os detalhes de um pedido específico',
            url: process.env.URL_API + 'orders/' + order.orderId
          }
        }
      })
    }
    return res.status(200).send({response});
  } catch (error) {
    return res.status(500).send({error: error});
  }
};

exports.postOrder = async (req, res, next) => {
  try {
    const queryProduct = 'SELECT * FROM products WHERE productId = ?';
    const resultProduct = await mysql.execute(queryProduct, [req.body.productId]);

    if (resultProduct.length == 0) {
      return res.status(404).send({ message: 'Produto não encontrado'});
    }

    const queryOrder = 'INSERT INTO orders (productId, quantity) VALUES (?,?)';
    const resultOrder = await mysql.execute(queryOrder, [
      req.body.productId,
      req.body.quantity
    ]);
  }
};
```

```

const response = {
  message: 'Pedido inserido com sucesso',
  createdOrder: {
    orderId: resultOrder.insertId,
    id_produto: req.body.productId,
    quantidade: req.body.quantity,
    request: {
      type: 'GET',
      description: 'Retorna todos os pedidos',
      url: process.env.URL_API + 'orders'
    }
  }
}
return res.status(201).send(response);

} catch (error) {
  return res.status(500).send({ error: error });
}
};

exports.getOrderDetail = async (req, res, next) => {
try {
  const query = "SELECT * FROM orders WHERE orderId = ?";
  const result = await mysql.execute(query, [req.params.orderId]);
  const response = {
    order: {
      orderId: result[0].orderId,
      quantity: result[0].quantity,
      productId: result[0].productId,
      request: {
        type: 'GET',
        description: 'Retorna todos os pedidos',
        url: process.env.URL_API + 'orders'
      }
    }
  }
  return res.status(200).send(response);
} catch (error) {
  return res.status(500).send({error: error});
}
};

exports.deleteOrder = async (req, res, next) => {
try {
  const query = "DELETE FROM orders WHERE orderId = ?";
  const result = await mysql.execute(query, [req.params.orderId]);
  const response = {
    message: "Pedido removido com sucesso!",
    request: {
      type: 'POST',
      description: 'Insere um pedido',
      url: process.env.URL_API + 'orders',
      body: {
        productId: 'Number',
        quantity: 'Number'
      }
    }
  }
  return res.status(200).send(response);
}

```

```

} catch (error) {
    return res.status(500).send({error: error});
}
};

```

Alterando o controller de usuários

controllers\user-controller.js

```

const mysql = require('../mysql');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');

exports.createUser = async (req, res, next) => {

    try {
        var query = `SELECT * FROM users WHERE email = ?`;
        var result = await mysql.execute(query, [req.body.email]);

        if (result.length > 0) {
            return res.status(409).send({ message: 'Usuário já cadastrado' })
        }

        const hash = await bcrypt.hashSync(req.body.password, 10);

        query = 'INSERT INTO users (email, password) VALUES (?,?)';
        const results = await mysql.execute(query, [req.body.email,hash]);

        const response = {
            message: 'Usuário criado com sucesso',
            createdUser: {
                userId: results.insertId,
                email: req.body.email
            }
        }
        return res.status(201).send(response);
    } catch (error) {
        return res.status(500).send({ error: error });
    }
};

exports.Login = async (req, res, next) => {

    try {
        const query = `SELECT * FROM users WHERE email = ?`;
        var results = await mysql.execute(query, [req.body.email]);

        if (results.length < 1) {
            return res.status(401).send({ message: 'Falha na autenticação' })
        }

        if (await bcrypt.compareSync(req.body.password, results[0].password)) {
            const token = jwt.sign({
                userId: results[0].userId,
                email: results[0].email
            }

```

```
},
process.env.JWT_KEY,
{
  expiresIn: "4h"
});
return res.status(200).send({
  message: 'Autenticado com sucesso',
  token: token
});
}
return res.status(401).send({ message: 'Falha na autenticação' })

} catch (error) {
  return res.status(500).send({ message: 'Falha na autenticação' });
}
};
```

Alterando a rota de produtos

routes\product-route.js

```
const express = require('express');
const router = express.Router();
const multer = require('multer');
const login = require('../middleware/login');

const ProductsController = require('../controllers/product-controller');

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, './uploads/');
  },
  filename: function (req, file, cb) {
    cb(null, new Date().toDateString() + "-" + file.originalname);
  }
});

const fileFilter = (req, file, cb) => {
  if (file.mimetype === 'image/jpeg' || file.mimetype === 'image/png') {
    cb(null, true);
  } else {
    cb(null, false);
  }
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024 * 5
  },
  fileFilter: fileFilter
});

router.get('/', ProductsController.getProducts);
router.post('/', [
  login,
  upload.single('productImage'),
  ProductsController.postProduct
]);
router.get('/:productId', ProductsController.getProductDetail);
router.patch('/:productId', login, ProductsController.updateProduct);
router.delete('/:productId', login, ProductsController.deleteProduct);

module.exports = router;
```

Alterando a rota de pedidos

routes\order-route.js

```
const express = require('express');
const router = express.Router();
const login = require('../middleware/login');

const orderController = require('../controllers/order-controller');

router.get('/', orderController.getOrders);
router.post('/', orderController.postOrder);
router.get('/:orderId', orderController.getOrderDetail);
router.delete('/:orderId', login, orderController.deleteOrder);

module.exports = router;
```

Alterando a rota de usuários

routes\user-route.js

```
const express = require('express');
const router = express.Router();

const userController = require('../controllers/user-controller');

router.post('/', userController.createUser);
router.post('/login', userController.Login)

module.exports = router;
```

Alterando app.js

```
const express = require('express');
const app = express();
const morgan = require('morgan');
const bodyParser = require('body-parser');

const productRoute = require('./routes/product-route');
const orderRoute = require('./routes/order-route');
const userRoute = require('./routes/user-route');

app.use(morgan('dev'));
app.use('/uploads', express.static('uploads'));

app.use(bodyParser.urlencoded({ extended: false })); // apenas dados simples
app.use(bodyParser.json()); // json de entrada no body

app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header(
    'Access-Control-Allow-Headers',
    'Origin, X-Requested-With, Content-Type, Accept, Authorization'
  );
  if (req.method === 'OPTIONS') {
    res.header('Access-Control-Allow-Methods', 'PUT, POST, PATCH, DELETE, GET');
    return res.status(200).send({});
  }
  next();
});

app.use('/products', productRoute);
app.use('/orders', orderRoute);
app.use('/users', userRoute);

// Tratamento quando não for encontrada rota
app.use((req, res, next) => {
  const erro = new Error('Rota não encontrada!');
  erro.status = 404;
  next(erro);
});

app.use((error, req, res, next) => {
  res.status(error.status || 500);
  return res.send({
    erro: {
      mensagem: error.message
    }
  });
});

module.exports = app;
```

Cadastrando produtos:

POST [localhost:3000/products](#)

Multipart	Bearer	Query	Header	Docs	Send	201 Created	3.5 s	274 B	
<input checked="" type="checkbox"/> name	Ventilador					Preview	Header	Cookie	Timeline
<input checked="" type="checkbox"/> price	120					1 + { 2 "message": "Produto inserido com sucesso", 3 "createdProduct": { 4 "productId": 1, 5 "name": "Ventilador", 6 "price": "120", 7 "productImage": "uploads\\Wed May 26 2021-ventilador.jpg", 8 "request": { 9 "type": "POST", 10 "description": "Retorna todos os produtos", 11 "url": "http://localhost:3000/products" 12 } 13 } 14 }			
<input checked="" type="checkbox"/> productImage	<input type="file" value="ventilador.jpg"/>								
<input checked="" type="checkbox"/> name	value								
<input checked="" type="checkbox"/> New name	New value								

POST [localhost:3000/products](#)

Multipart	Bearer	Query	Header	Docs	Send	201 Created	405 ms	281 B	
<input checked="" type="checkbox"/> name	Liquidificador					Preview	Header	Cookie	Timeline
<input checked="" type="checkbox"/> price	80					1 + { 2 "message": "Produto inserido com sucesso", 3 "createdProduct": { 4 "productId": 2, 5 "name": "Liquidificador", 6 "price": "80", 7 "productImage": "uploads\\Wed May 26 2021-liquidificador.jpg", 8 "request": { 9 "type": "POST", 10 "description": "Retorna todos os produtos", 11 "url": "http://localhost:3000/products" 12 } 13 } 14 }			
<input checked="" type="checkbox"/> productImage	<input type="file" value="liquidificador.jpg"/>								
<input checked="" type="checkbox"/> name	value								
<input checked="" type="checkbox"/> New name	New value								

POST [localhost:3000/products](#)

Multipart	Bearer	Query	Header	Docs	Send	201 Created	573 ms	280 B	
<input checked="" type="checkbox"/> name	Rádio relógio					Preview	Header	Cookie	Timeline
<input checked="" type="checkbox"/> price	6					1 + { 2 "message": "Produto inserido com sucesso", 3 "createdProduct": { 4 "productId": 3, 5 "name": "Rádio relógio", 6 "price": "6", 7 "productImage": "uploads\\Wed May 26 2021-radio_relogio.jpg", 8 "request": { 9 "type": "POST", 10 "description": "Retorna todos os produtos", 11 "url": "http://localhost:3000/products" 12 } 13 } 14 }			
<input checked="" type="checkbox"/> productImage	<input type="file" value="radio_relogio.jpg"/>								
<input checked="" type="checkbox"/> name	value								
<input checked="" type="checkbox"/> New name	New value								

POST [localhost:3000/products](#)

Multipart	Bearer	Query	Header	Docs	Send	201 Created	456 ms	272 B	
<input checked="" type="checkbox"/> name	Batedeira					Preview	Header	Cookie	Timeline
<input checked="" type="checkbox"/> price	110					1 + { 2 "message": "Produto inserido com sucesso", 3 "createdProduct": { 4 "productId": 4, 5 "name": "Batedeira", 6 "price": "110", 7 "productImage": "uploads\\Wed May 26 2021-batedeira.jpg", 8 "request": { 9 "type": "POST", 10 "description": "Retorna todos os produtos", 11 "url": "http://localhost:3000/products" 12 } 13 } 14 }			
<input checked="" type="checkbox"/> productImage	<input type="file" value="batedeira.jpg"/>								
<input checked="" type="checkbox"/> name	value								
<input checked="" type="checkbox"/> New name	New value								

Atualizando produto:

PATCH [localhost:3000/products/3](#)

JSON	Bearer	Query	Header	Docs	Send	202 Accepted	369 ms	206 B		
<pre>1 + { 2 "name": "Rádio relógio", 3 "price": 60 4 }</pre>						Preview	Header	Cookie	Timeline	
						1 + { 2 "message": "Produto atualizado com sucesso", 3 "updatedProduct": { 4 "productId": "3", 5 "request": { 6 "type": "GET", 7 "description": "Retorna os detalhes de um produto específico", 8 "url": "http://localhost:3000/products/3" 9 } 10 } 11 }				

Listando produtos:

GET ▾ localhost:3000/products

Send 200 OK 286 ms 984 B

Body ▾ Auth ▾ Query Header Docs

Preview ▾ Header ▾ Cookie Timeline

```
1 + {
2 +   "response": [
3 +     {
4 +       "quantity": 4,
5 +       "products": [
6 +         {
7 +           "productId": 1,
8 +           "name": "Ventilador",
9 +           "price": 128,
10 +          "productImage": "uploads\\Wed May 26 2021-ventilador.jpg",
11 +          "request": {
12 +            "type": "GET",
13 +            "description": "Retorna os detalhes de um produto específico",
14 +            "url": "http://localhost:3000/products/1"
15 +          }
16 +        },
17 +        {
18 +          "productId": 2,
19 +          "name": "Liquidificador",
20 +          "price": 88,
21 +          "productImage": "uploads\\Wed May 26 2021-liquidificador.jpg",
22 +          "request": {
23 +            "type": "GET",
24 +            "description": "Retorna os detalhes de um produto específico",
25 +            "url": "http://localhost:3000/products/2"
26 +          }
27 +        },
28 +        {
29 +          "productId": 3,
30 +          "name": "Rádio relógio",
31 +          "price": 68,
32 +          "productImage": "uploads\\Wed May 26 2021-radio_relogio.jpg",
33 +          "request": {
34 +            "type": "GET",
35 +            "description": "Retorna os detalhes de um produto específico",
36 +            "url": "http://localhost:3000/products/3"
37 +          }
38 +        },
39 +        {
40 +          "productId": 4,
41 +          "name": "Batedeira",
42 +          "price": 110,
43 +          "productImage": "uploads\\Wed May 26 2021-batedeira.jpg",
44 +          "request": {
45 +            "type": "GET",
46 +            "description": "Retorna os detalhes de um produto específico",
47 +            "url": "http://localhost:3000/products/4"
48 +          }
49 +      ]
50 +    }
51 + }
```

Select a body type from above



Exibindo detalhes de um produto específico:

GET ▾ localhost:3000/products/3

Send 200 OK 241 ms 229 B

Body ▾ Auth ▾ Query Header Docs

Preview ▾ Header ▾ Cookie Timeline

```
1 + {
2 +   "product": {
3 +     "productId": 3,
4 +     "name": "Rádio relógio",
5 +     "price": 68,
6 +     "productImage": "uploads\\Wed May 26 2021-radio_relogio.jpg",
7 +     "request": {
8 +       "type": "GET",
9 +       "description": "Retorna todos os produtos",
10 +      "url": "http://localhost:3000/products"
11 +    }
12 +  }
13 + }
```

Excluindo um produto:

DELETE ▾ localhost:3000/products/4

Send 200 OK 445 ms 184 B

Body ▾ Bearer ▾ Query Header Docs

TOKEN eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJcIj0j1c2VvSWQjOj1simVtYWlsIjoicm9iZXJ0b19waW5oZWlyb0BnbWFpbC

PREFIX

ENABLED

Preview ▾ Header ▾ Cookie Timeline

```
1 + {
2 +   "message": "Produto removido com sucesso!",
3 +   "request": {
4 +     "type": "POST",
5 +     "description": "Insere um produto",
6 +     "url": "http://localhost:3000/products",
7 +     "body": {
8 +       "name": "String",
9 +       "price": "Number"
10 +     }
11 +   }
12 + }
```

Cadastrando pedidos:

POST	localhost:3000/orders	Send	201 Created	685 ms	203 B
JSON	Bearer	Query	Header	Docs	Preview
1 = { 2 "productId": 2, 3 "quantity": 5 4 }					1 + { 2 "message": "Pedido inserido com sucesso", 3 "createdOrder": { 4 "orderId": 1, 5 "id_produto": 2, 6 "quantidade": 5, 7 "request": { 8 "type": "GET", 9 "description": "Retorna todos os pedidos", 10 "url": "http://localhost:3000/orders" 11 } 12 } 13 }

POST	localhost:3000/orders	Send	201 Created	725 ms	203 B
JSON	Bearer	Query	Header	Docs	Preview
1 = { 2 "productId": 1, 3 "quantity": 2 4 }					1 + { 2 "message": "Pedido inserido com sucesso", 3 "createdOrder": { 4 "orderId": 2, 5 "id_produto": 1, 6 "quantidade": 2, 7 "request": { 8 "type": "GET", 9 "description": "Retorna todos os pedidos", 10 "url": "http://localhost:3000/orders" 11 } 12 } 13 }

Listando pedidos:

GET	localhost:3000/orders	Send	200 OK	255 ms	448 B
Body	Auth	Query	Header	Docs	Preview
Select a body type from above					1 + { 2 "response": { 3 "orders": [4 { 5 "orderId": 2, 6 "quantity": 2, 7 "product": { 8 "productId": 1, 9 "name": "Ventilador", 10 "price": 120 11 }, 12 "request": { 13 "type": "GET", 14 "description": "Retorna os detalhes de um pedido específico", 15 "url": "http://localhost:3000/orders/2" 16 } 17 }, 18 { 19 "orderId": 1, 20 "quantity": 5, 21 "product": { 22 "productId": 2, 23 "name": "Liquificador", 24 "price": 80 25 }, 26 "request": { 27 "type": "GET", 28 "description": "Retorna os detalhes de um pedido específico", 29 "url": "http://localhost:3000/orders/1" 30 } 31 } 32] 33 } 34 }

Listando detalhes de um pedido específico:

GET	localhost:3000/orders/1	Send	200 OK	795 ms	153 B
Body	Auth	Query	Header	Docs	Preview
					1 + { 2 "order": { 3 "orderId": 1, 4 "quantity": 5, 5 "productId": 2, 6 "request": { 7 "type": "GET", 8 "description": "Retorna todos os pedidos", 9 "url": "http://localhost:3000/orders" 10 } 11 } 12 }

```

1 + {
2 +   "order": {
3 +     "orderId": 2,
4 +     "quantity": 2,
5 +     "productId": 1,
6 +     "request": {
7 +       "type": "GET",
8 +       "description": "Retorna todos os pedidos",
9 +       "url": "http://localhost:3000/orders"
10 +     }
11 +   }
12 + }

```

Excluindo um pedido:

```

1 + {
2 +   "message": "Pedido removido com sucesso!",
3 +   "request": {
4 +     "type": "POST",
5 +     "description": "Insere um pedido",
6 +     "url": "http://localhost:3000/orders",
7 +     "body": {
8 +       "productId": "Number",
9 +       "quantity": "Number"
10 +     }
11 +   }
12 + }

```

mysql.js

```

const mysql = require('mysql2');

var pool = mysql.createPool({
  "connectionLimit": 10000,
  "user": process.env.MYSQL_USER,
  "password": process.env.MYSQL_PASSWORD,
  "database": process.env.MYSQL_DATABASE,
  "host": process.env.MYSQL_HOST,
  "port": process.env.MYSQL_PORT
});

exports.execute = (query, params=[]) => {
  return new Promise((resolve, reject) => {
    pool.query(query, params, (error, result, fields) => {
      if (error) {
        reject(error);
      } else {
        resolve(result)
      }
    });
  });
}

exports.pool = pool;

```

