**UNIVERSIDADE FEDERAL DO PIAUÍ**

**CENTRO DE CIÊNCIAS DA NATUREZA – CCN**

**DEPARTAMENTO DE COMPUTAÇÃO**

**ENGENHARIA DE SOFTWARE II**

**ALUNOS(A):**

Carlos Sávio Fontenele Reis

Gabriel Reis Alencar Nunes

Paulo Eduardo Borges do Vale

Paulo Eduardo Ramos de Araújo

Vicente Cleyton Gonçalves de Sousa Carvalho

**PROFESSOR(A):**

Armando Soares Sousa

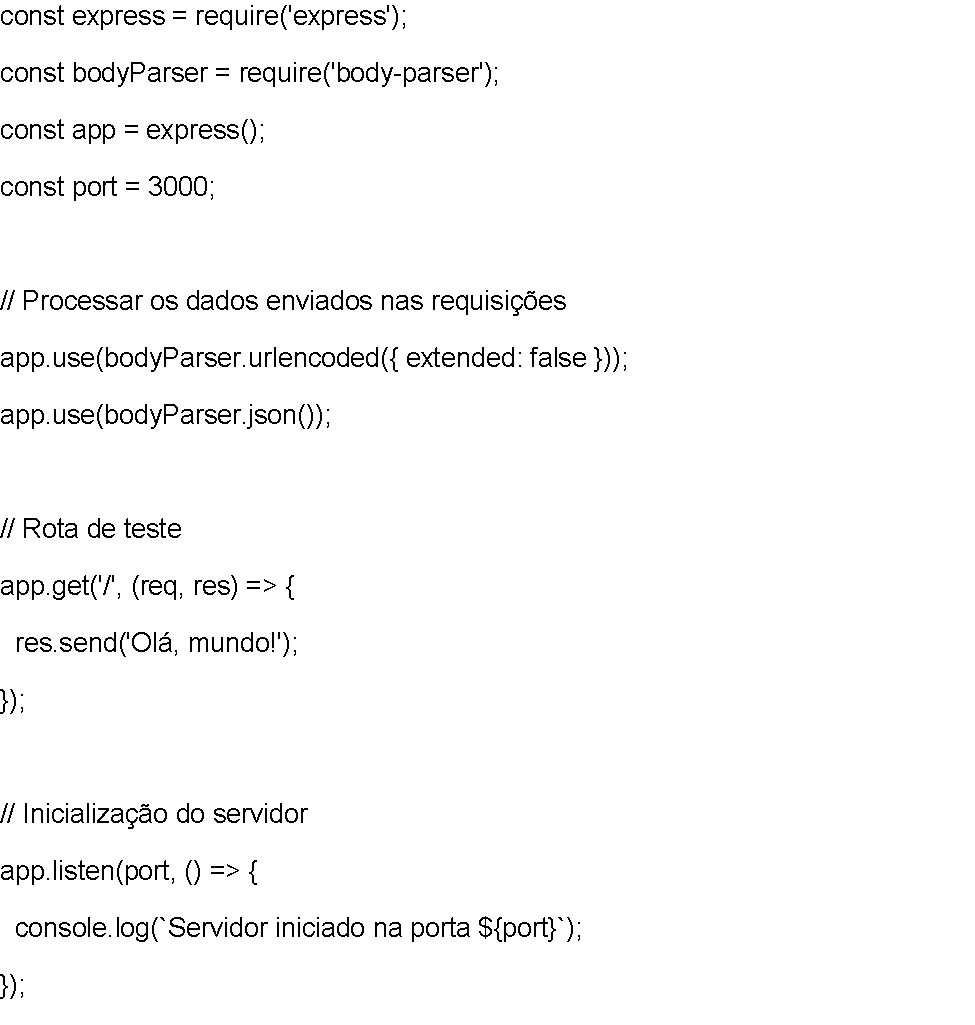
**Teresina – PI**

15-06-2023

**Instalando a IDE Visual Studio Code**

1. Acesse o site oficial
   1. Vá para o site oficial do Visual Studio Code em https://code.visualstudio.com.
   2. Na página de download, o site deve detectar automaticamente o seu sistema operacional e fornecer a opção de download correta. Se isso não acontecer, selecione manualmente o seu sistema operacional no menu suspenso.
2. Inicie o download
   * 1. Clique no botão de download para iniciar o processo. Dependendo do seu sistema operacional, você pode ser solicitado a confirmar o download ou escolher um local para salvar o arquivo de instalação.
3. Instale o Visual Studio Code
   1. Windows
      1. Após o download, execute o arquivo de instalação (.exe). Você pode ser solicitado a confirmar permissões de administrador. Siga as instruções na tela e aceite os termos de uso para concluir a instalação.
   2. macOS
      1. Após o download, abra o arquivo .dmg. Arraste e solte o aplicativo Visual Studio Code na pasta "Aplicativos". O Visual Studio Code está agora instalado e pronto para ser usado.
   3. Linux
      1. Após o download, abra um terminal e navegue até o local onde o arquivo de instalação foi baixado. Execute o comando “tar -xvf nome\_do\_arquivo.tar.gz” para extrair o pacote
      2. Navegue para o diretório extraído e execute o comando “./Code” para iniciar o Visual
      3. Você também pode adicionar um atalho no menu do sistema para facilitar o acesso ao Visual Studio Code.
4. Configurações adicionais
   1. Após a instalação, o Visual Studio Code estará pronto para ser usado. Pode-se também autenticar com a aplicação e instalar extensões para auxiliar no uso.

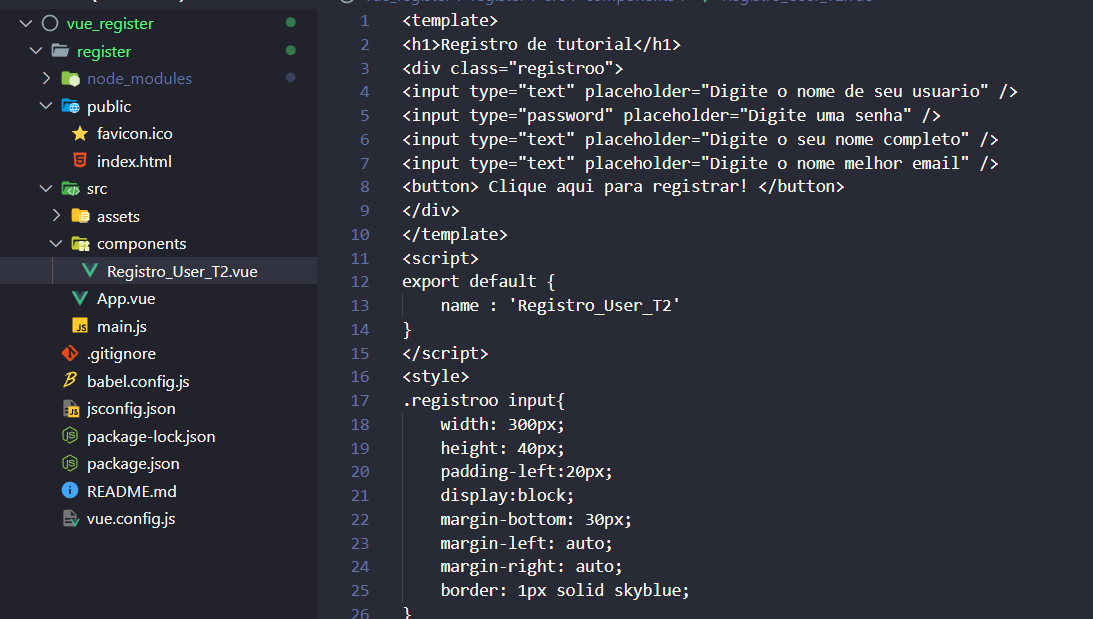
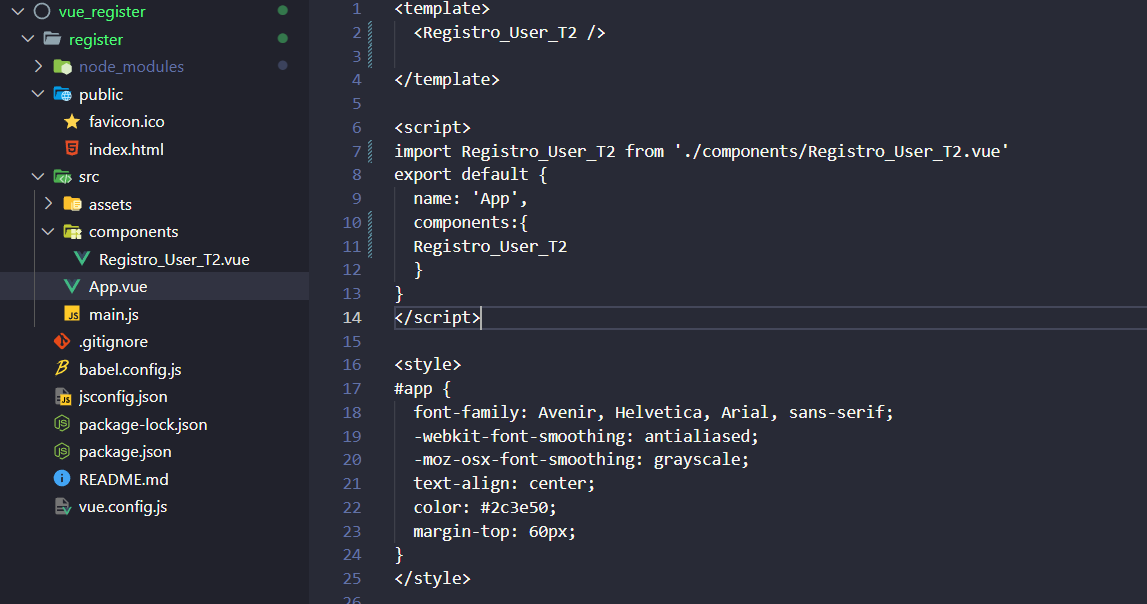
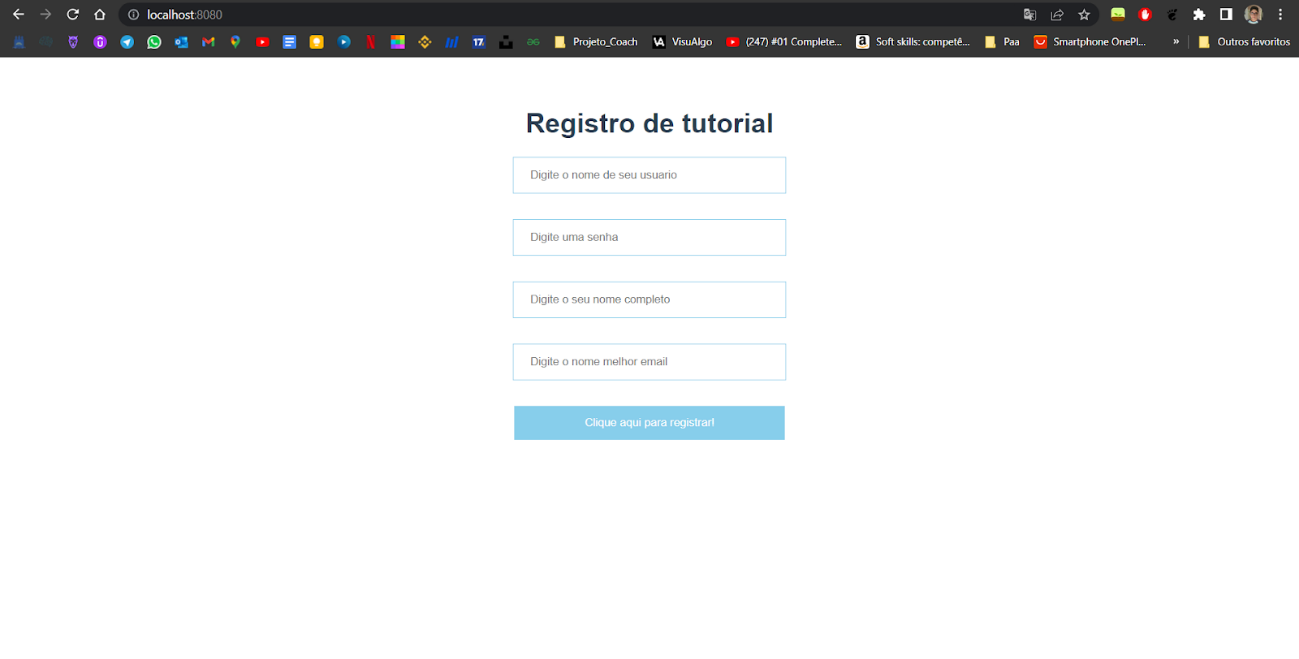
**Inicializando e Configurando o Ambiente de Desenvolvimento com NodeJS**

1. Baixe e instale o Node.js
   1. Acesse o site oficial do Node.js em <https://nodejs.org>.
   2. Na página inicial, você verá botões para baixar a versão recomendada (LTS) do Node.js. Selecione o botão apropriado para o seu sistema operacional (Windows, macOS ou Linux).
   3. Após o download, execute o instalador e siga as instruções para instalar o Node.js no seu sistema.
2. Configuração do projeto
   1. Crie uma nova pasta para o seu projeto.
   2. Abra um terminal ou prompt de comando e navegue até a pasta do seu projeto.
   3. Execute o comando “npm init -y” para inicializar um novo projeto Node.js com as configurações padrão.
3. Instalação de pacotes
   1. Agora, você precisa instalar o framework Express.js para inicializar o servidor web.
   2. Execute os comandos “npm install express” e “npm install body-parser”
      1. O pacote express é o framework que nos permite criar a API, e o pacote body-parser é utilizado para processar dados enviados nas requisições HTTP.
4. Criando uma rota de teste
   1. Crie o arquivo “server.js” na pasta do seu projeto. Abra-o em um editor de texto e adicione o seguinte código: 
   2. O código cria um servidor Express, define uma rota GET ("/") que responde com "Olá, mundo!" quando acessada e inicia o servidor na porta 3000.
5. Inicialização da API
   1. No terminal execute o comando “node server.js” para iniciar a API.
   2. A mensagem "Servidor iniciado na porta 3000" será mostrada se o servidor inicializar com sucesso.
   3. No navegador, acesse o endereço <http://localhost:3000/>, você receberá a resposta "Olá, mundo!".

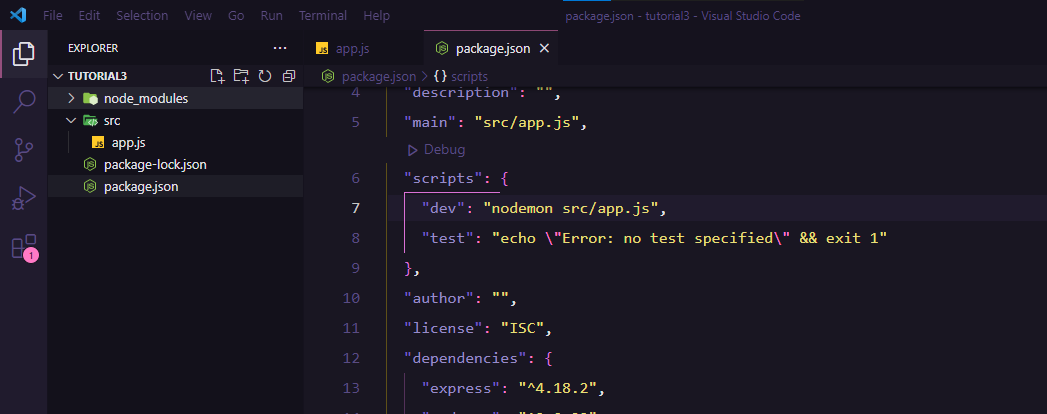
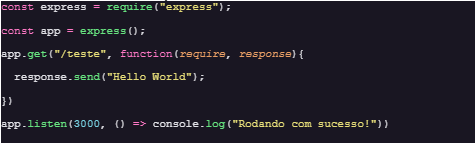
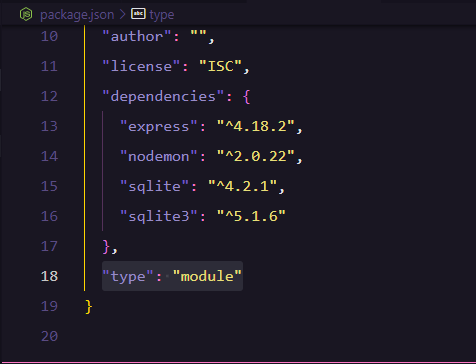
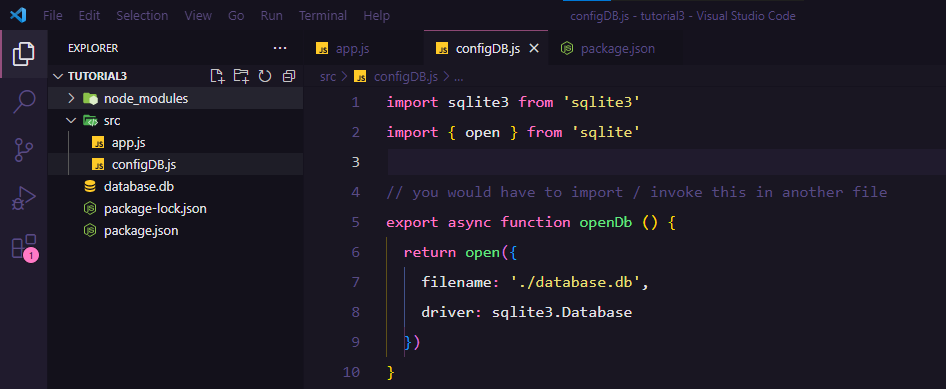
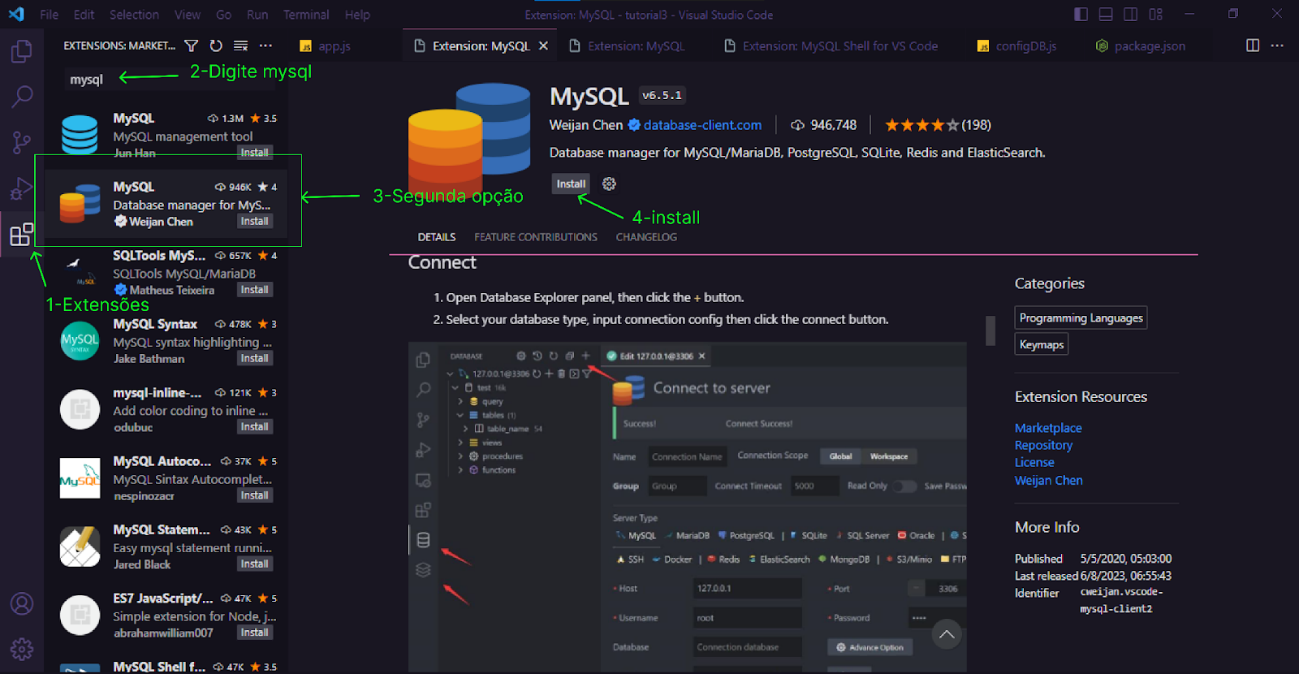
**Inicializando e Configurando o Ambiente de Desenvolvimento com VueJS**

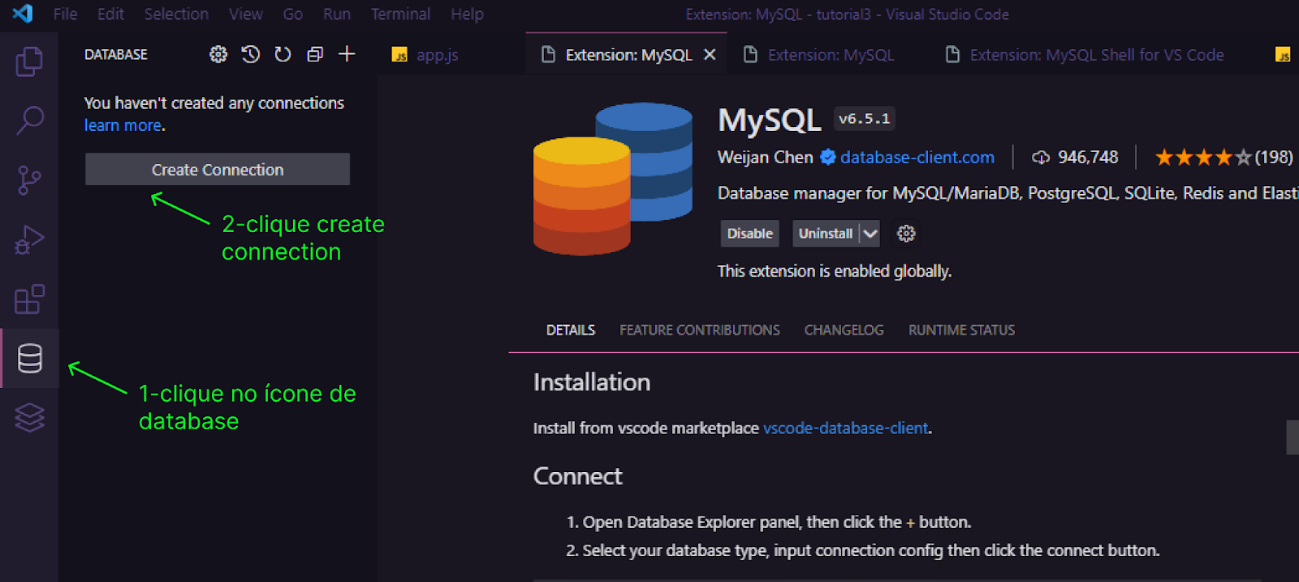
1. Instalação do Node e NPM
   1. Antes de começar, verifique se você tem o Node.js e o NPM (Node Package Manager) instalados no seu sistema. O NPM é instalado automaticamente junto com o Node.js
2. Instalação do Vue CLI
   1. O Vue CLI é uma ferramenta de linha de comando que nos ajuda a criar e configurar projetos Vue. Para instalá-lo, abra um terminal ou prompt de comando e execute o comando “npm install -g @vue/cli”
3. Configuração do projeto
   1. Navegue até o diretório onde deseja criar o projeto e execute o comando “vue create nome-do-projeto”
   2. O Vue CLI irá perguntar algumas opções de configuração para o projeto. Escolha a configuração padrão, que inclui o Babel e o ESLint e espere até que todas as dependências sejam instaladas.
4. Instalando módulos
   1. Após a criação do projeto ou instalação de pacotes *npm*, navegue para o diretório do projeto e execute o comando “npm install” para atualizar as dependências.
5. Iniciando servidor web
   1. execute o comando “npm run serve” para iniciar o servidor de desenvolvimento.
   2. No navegador, acesso o endereço <http://localhost:8080> para visualizar a aplicação Vue em execução.

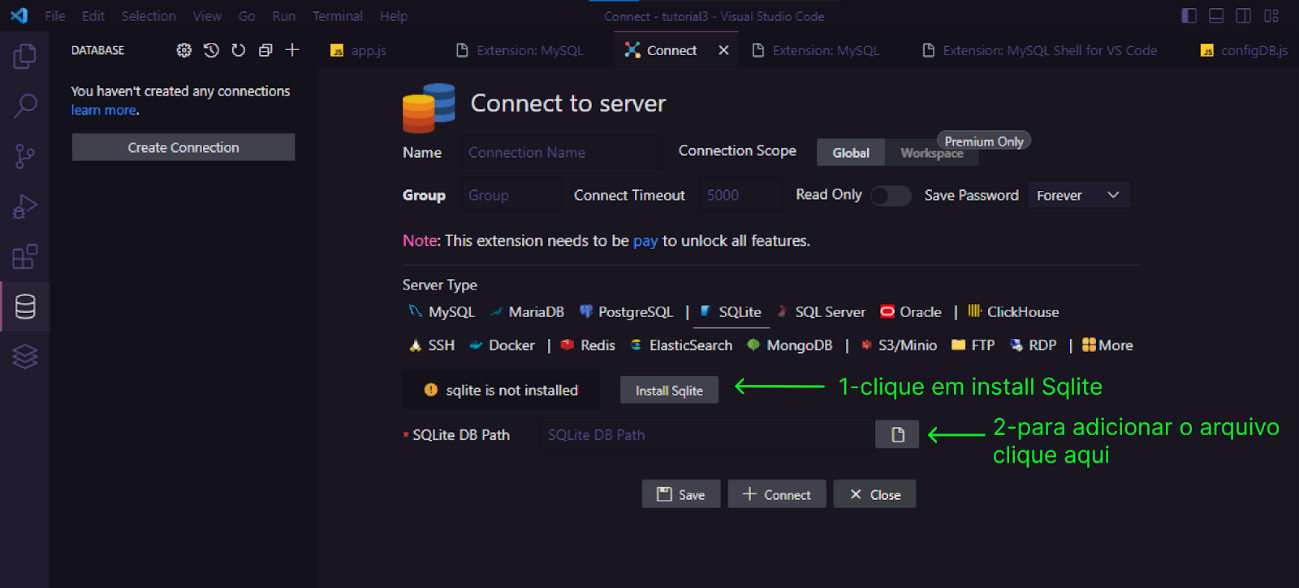
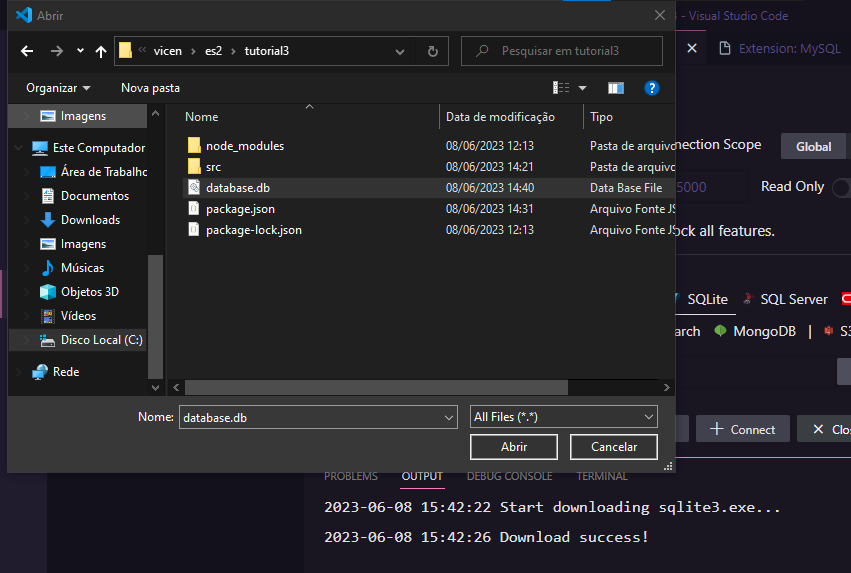
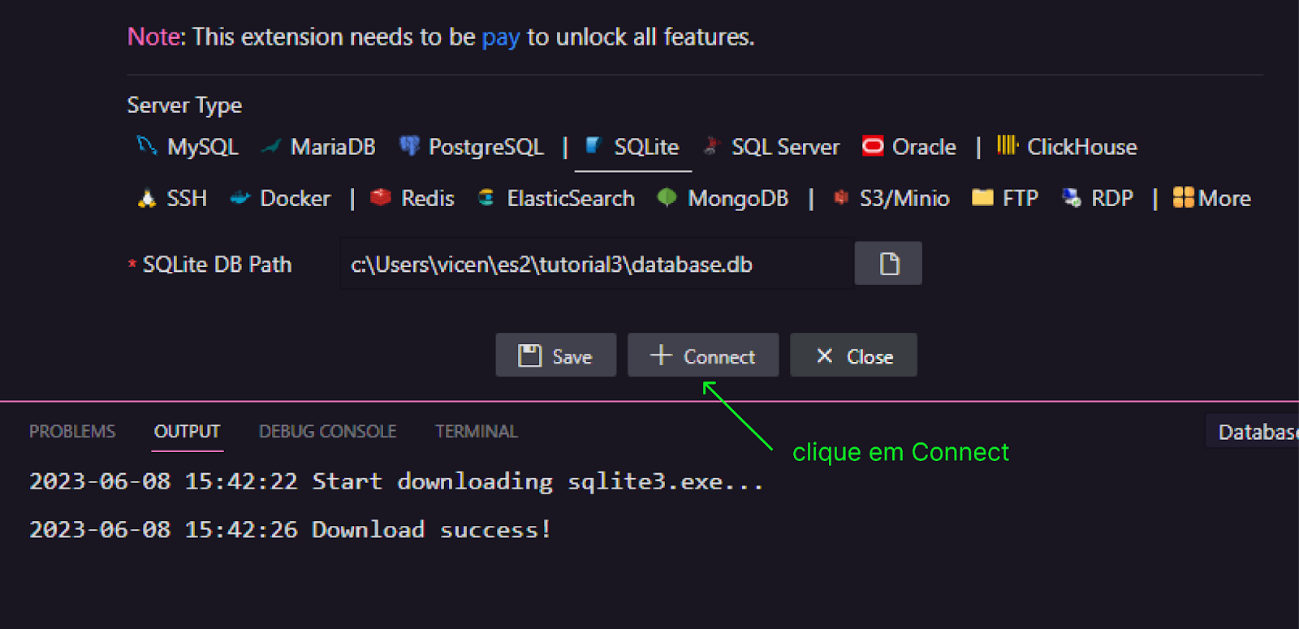
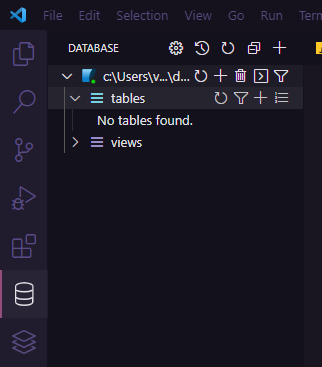
**Criando uma aplicação web básica de formulário com VueJS**

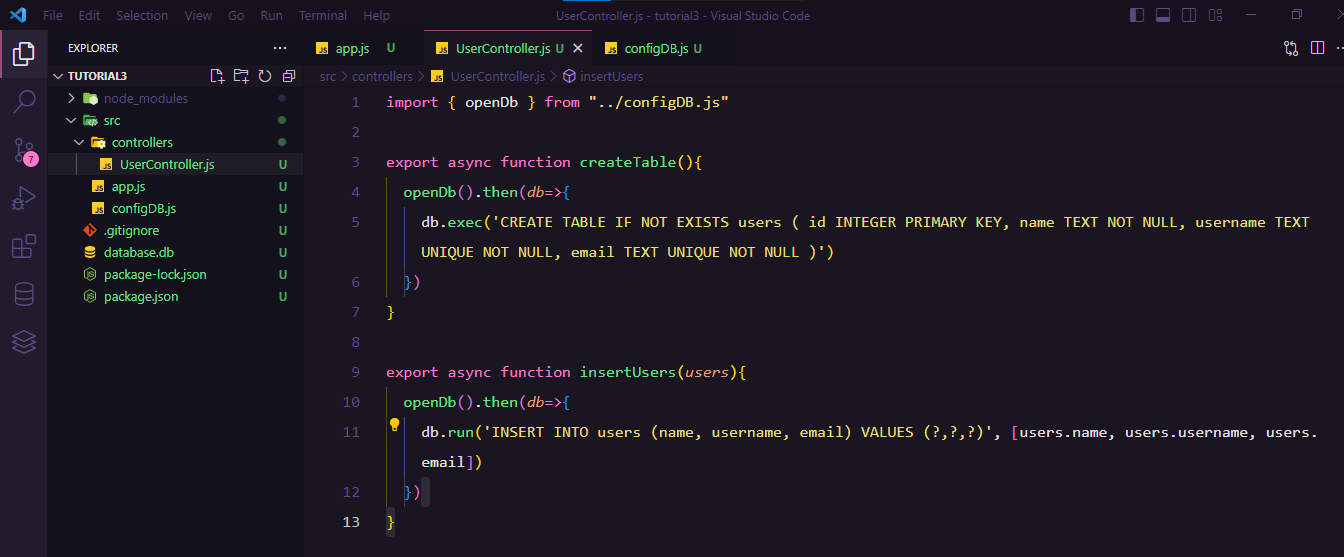
1. Criando projeto
   1. Com o Node, NPM e Vue CLI instalados, execute o comando “vue create nome-do-projeto” para criar os arquivos básicos de uma aplicação Vue.
   2. Navegue até a pasta do projeto com o comando “cd nome-do-projeto” e em seguida execute “npm run serve” para iniciar o servidor de aplicação.
   3. Após o utilizar os comandos acima, arquivos e subpastas são criados. O mais importante para agora está no “/src/componentes” que contém o arquivo principal que vai inicializado quando é aberto a instância do servidor: App.vue.
   4. Dentro de components, como exemplo, para a criação de um formulário, é criado um arquivo chamado Registro\_User\_T2.vue, como exemplo para criação de um formulário.
   5. Dentro de tags chamadas <template>, instanciamos o nosso furmulário de saída para após isso, ser instanciada em App.vue.
   6. Segue abaixo o pseudocódigo de como é criado um furmulário de exemplo. 
   7. Agora iremos para o APP.vue instanciar o Registro\_User\_T2.vue para ser mostrado na página inicial: 
   8. Com isso, a pagina web abaixo será exibida com um formulário para cadastro de usuários: 

**Como Instalar e Configurar o Banco de Dados**

1. Crie uma pasta para o seu projeto
2. Execute o comando “npm init -y” para inicializar um novo projeto Node.js com as configurações padrão.
3. Instale o express: npm install express ou npm i express
4. Crie uma pasta src no seu projeto e logo em seguida crie o arquivo app.js dentro de src
5. vamos instalar o nodemon utilizando o comando npm install --save nodemon(o nodemon reinicia a aplicação automaticamente após salvo alguma mudança).
6. vamos fazer um script para o nosso comando de execução, para isso abra o arquivo package.json e em scripts digite: "dev": "nodemon src/app.js",. 
7. agora, vamos começar nossa aplicação de teste. Em app.js digite: 
8. Para executar a aplicação de teste digite: npm run dev, se exibir “Rodando com sucesso!” está tudo ok.
9. localhost:3000/teste, se exibir “Hello World” está tudo ok.
10. Instalando o Sqlite
    1. Digite no seu terminal do projeto: npm install sqlite3 –save
    2. Em seguida: npm install sqlite –save
11. Para trabalhar com importação de módulos (para que o script possa importar outros arquivos JS), vamos especificar no package.json
    1. Digite: "type": "module" logo após dependencies
    2. 
12. vamos criar dentro da pasta src o arquivo configDB.js e inserir o código que nos vai permitir criar o arquivo database.db. 
13. agora vamos adaptar o código do arquivo app.js para que possa receber a importação do arquivo configDB.js e criar um método post.
    1. No arquivo app.js digite o código: 
14. instalar uma extensão que nos permite visualizar as tabelas do banco de dados.
    1. 
15. como usar a extensão mysql.
    1. Clique no ícone de database na barra à esquerda e depois em create connection.



* 1. Clique em install Sqlite e depois em SQLite DB path para adicionar o arquivo do database.db. 
  2. Localize a pasta do seu projeto e adicione database.db. 
  3. Clique em Connect. 
  4. Após clicar em Connect, vai ser exibido um banco de dados sem tabelas. 

1. vamos criar uma pasta controllers dentro de src e um arquivo UserController.js dentro da pasta controller(src/controller/UserController.js).
   1. Código para ser digitado em UserController.js : 
2. vamos fazer algumas mudanças no código de app.js para podermos importar as funcionalidades do arquivo UserController.js.
   1. Código para ser digitado em app.js : 
3. Pronto, agora só testar no navegador chamando “localhost:3000/users”

**Como Persistir e Recuperar Dados de um Usuário no Banco de Dados**

1. Insira o comando em um diretório npm install express sqlite.
2. Crie um arquivo index.js com o código a seguir.

const express = require('express');

const db = require('./conexao.js');

const PORT = 3000;

const app = express();

app.use(express.urlencoded({ extended: false }));

app.use(express.json());

db.init();

app.set('view engine', 'ejs');

app.get('/user', function (req, res) {

res.render('cadastrar');

});

app.post('/userSet', function(req, res){

const a = { id: req.body.id,

nome : req.body.nome,

email :req.body.email,

numero :req.body.telefone,

usuario : req.body.usuario};

db.db.run('INSERT INTO usuario(id, nome, email, numero, usuario) VALUES(?, ?,?,?,?)',

[req.body.id, req.body.nome, req.body.email,req.body.telefone,req.body.usuario]

, (err) => {

if(err) {

return console.log(err.message);

}});

db.db.all('SELECT \* FROM usuario', [], (err, rows) => {

if (err) {

throw err;

}

rows.forEach((row) => {

a.id = row.nome;

a.nome = row.nome;

a.email = row.email;

a.numero = row.numero;

a.usuario = row.usuario;

});

res.render('usuarios', {a});

});

});

app.listen(PORT, function (err) {

if (err) console.log(err);

console.log("Server listening on PORT", PORT);

});

1. Após selecionar esse código e copiar para um arquivo java script, crie um arquivo conexao.js para o banco de dados com o seguinte código.
2. let sqlite3 = require('sqlite3').verbose();
3. let db = new sqlite3.Database('./sqlite.db');
4. let init = function () {
5. db.run("CREATE TABLE if not exists usuario (" +
6. "id INTEGER PRIMARY KEY AUTOINCREMENT," +
7. " nome TEXT," +
8. " email TEXT," +
9. " numero INT," +
10. " usuario TEXT" +
11. ")");
12. };
13. module.exports = {
14. init: init,
15. db: db
16. };

4. Crie um arquivo usuario.ejs com o seguinte código.

<!DOCTYPE html>

<html>

<body>

<h1>Usuarios</h1>

<div class="flea-container">

{% for as in a %}

<div>

Id : {{ as.id }}<br> Nome: {{ as.nome }}<br> Email : {{ as.email }}<br> Numero : {{ as.numero }}<br> usuario : {{ as.usuario }}

</div>

{% endfor %}

</div>

</body>

</html>

5.Crie um arquivo com o nome cadastrar.ejs com o seguinte código.

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<!-- Pagina cadastro usuario -->

<div class="cadastrar ">Cadastrar Usúario</div>

<div class="cadastrar id">id</div>

<div class="cadastrar nome">nome</div>

<div class="cadastrar email">email</div>

<div class="cadastrar telefone">telefone </div>

<div class="cadastrar usuario">usuario</div>

<div class="cadastrar enviar">Enviar</div>

<form action="/userSet/", method="POST">

<div>

<input class="cadastrar caixaid type="text" name="id">

</input>

</div>

<div>

<input class="cadastrar caixanome" type="number" name="nome">

</input>

</div>

<div>

<input class="cadastrar caixaemail" type="text" name="email">

</input>

</div>

<input class="cadastrar caixausuario type="text" name="usuario">

</input>

</div>

<div>

<div>

<input class="cadastrar caixatelefone type="number" name="telefone">

</input>

</div>

<div ><input class="cadastrar caixaEnviar" type="submit" value="Enviar">

</input></div>

<div>

</form>

</body>

</html>

6. Crie um arquivo views e coloque os arquivos ejs criados nele.

7.Use o comando node index.js .

**Como Fazer Upload de Arquivos para Aplicação Web**

1. Inicialize o projeto Node.js:

# $ npm init -y

1. Instale as dependências necessárias:

# $ npm install express multer --save

1. Crie um arquivo **server.js** e adicione o seguinte código:

const express = require('express'); const multer = require('multer'); const path = require('path'); const fs = require('fs');

const app = express();

//TRATAMENTO DAS IMAGENS ENVIADAS

const storage = multer.diskStorage({ destination: function (req, file, cb) {

cb(null, 'uploads/');

},//RENOMEIA A IMAGEM

filename: function (req, file, cb) { cb(null, file.originalname + Date.now() + path.extname(file.originalname));

}

})

const upload = multer({ storage })

//RECEBE A IMAGEM

app.post('/upload', upload.single('image'), (req, res) => { res.send({ message: 'Arquivo recebido com sucesso!' });

});

//RECUPERA AS IMAGENS DO DIRETÓRIO UPLOADS

app.get('/images', (req, res) => {

const uploadsDir = path.join( dirname, 'uploads');

fs.readdir(uploadsDir, (err, files) => { if (err) {

console.error(err);

res.status(500).send({ error: 'Erro ao listar os arquivos.' });

return;

}

const images = files.map(file => { return {

filename: file,

path: path.join('/uploads', file)

};

});

});

});

res.json({ images });

app.listen(3000, () => {

console.log('Servidor rodando na porta 3000');

});

Agora, vamos criar a aplicação Vue.js:

1. Certifique-se de ter o Vue CLI instalado. Se não tiver, você pode instalá- lo globalmente com o seguinte comando:

# $ npm install -g @vue/cli

1. Crie um novo projeto Vue.js com o Vue CLI:

# $ vue create image-upload

1. Escolha a opção "default" e aguarde a criação do projeto.
2. Navegue até o diretório do projeto Vue.js:

# $ cd image-upload

1. Instale a biblioteca Axios para fazer as chamadas HTTP:

# $ npm install axios --save

1. Substitua o conteúdo do arquivo **src/components/HelloWorld.vue** com o seguinte código:

<template>

<div>

<h2>Upload de Imagens</h2>

<input type="file" ref="fileInput" accept="image/\*" />

<button @click="uploadImage"

:disabled="uploading">Enviar</button>

<h2>Lista de Imagens</h2>

<ul>

<li v-for="image in images" :key="image.filename">

<img :src="getImageUrl(image.path)" style="max-width: 200px; max-height: 200px;" />

</li>

</ul>

</div>

</template>

<script>

import axios from 'axios';

export default { data() {

return { uploading: false, images: [],

};

},

methods: { uploadImage() {

const fileInput = this.$refs.fileInput; const file = fileInput.files[0];

if (!file) { return;

}

this.uploading = true;

const formData = new FormData(); formData.append('image', file);

axios

.post('http://localhost:3000/upload', formData)

.then(response => { console.log(response.data); this.uploading = false; fileInput.value = ''; this.getImages();

})

.catch(error => { console.error(error); this.uploading = false; fileInput.value = '';

});

},

async getImages() { try {

const response = await axios.get('http://localhost:3000/images');

if (response.data && response.data.images) { this.images = response.data.images;

} else {

console.error('Resposta inválida do servidor:', response.data);

}

} catch (error) {

console.error('Erro ao obter a lista de imagens:', error);

}

},

getImageUrl(path) {

return `http://localhost:3000/${path}`;

},

},

mounted() { this.getImages();

},

};

</script>

A aplicação Vue.js possui um componente **HelloWorld** que exibe um campo de upload de arquivo e uma lista de imagens. Ele usa a biblioteca Axios para fazer chamadas HTTP para o servidor Node.js.

Por fim, para iniciar o servidor Node.js e a aplicação Vue.js, abra duas janelas de terminal separadas e execute os seguintes comandos:

* Para iniciar o servidor Node.js (no diretório raiz do servidor):

# $ node server.js

* Para iniciar a aplicação Vue.js (no diretório raiz da aplicação Vue.js):

# $ npm run serve