

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
 - a. Vá para o site oficial do Visual Studio Code em <https://code.visualstudio.com>.
 - b. 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
 - i. 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
 - a. Windows
 - i. 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.
 - b. macOS
 - i. 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.
 - c. Linux
 - i. 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
 - ii. Navegue para o diretório extraído e execute o comando `"./Code"` para iniciar o Visual
 - iii. Você também pode adicionar um atalho no menu do sistema para facilitar o acesso ao Visual Studio Code.
4. Configurações adicionais
 - a. 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
 - a. Acesse o site oficial do Node.js em <https://nodejs.org>.
 - b. 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).
 - c. Após o download, execute o instalador e siga as instruções para instalar o Node.js no seu sistema.
2. Configuração do projeto
 - a. Crie uma nova pasta para o seu projeto.
 - b. Abra um terminal ou prompt de comando e navegue até a pasta do seu projeto.
 - c. Execute o comando `"npm init -y"` para inicializar um novo projeto Node.js com as configurações padrão.
3. Instalação de pacotes
 - a. Agora, você precisa instalar o framework Express.js para inicializar o servidor web.
 - b. Execute os comandos `"npm install express"` e `"npm install body-parser"`
 - i. 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

- a. Crie o arquivo “server.js” na pasta do seu projeto. Abra-o em um editor de texto e adicione o seguinte código:

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

const bodyParser = require('body-parser');

const app = express();

const port = 3000;


// Processar os dados enviados nas requisições
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());


// Rota de teste
app.get('/', (req, res) => {
  res.send('Olá, mundo!');
});


// Inicialização do servidor
app.listen(port, () => {
  console.log(`Servidor iniciado na porta ${port}`);
});
```

- b. 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

- a. No terminal execute o comando “node server.js” para iniciar a API.
- b. A mensagem "Servidor iniciado na porta 3000" será mostrada se o servidor inicializar com sucesso.
- c. 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

- a. 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

- a. 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”

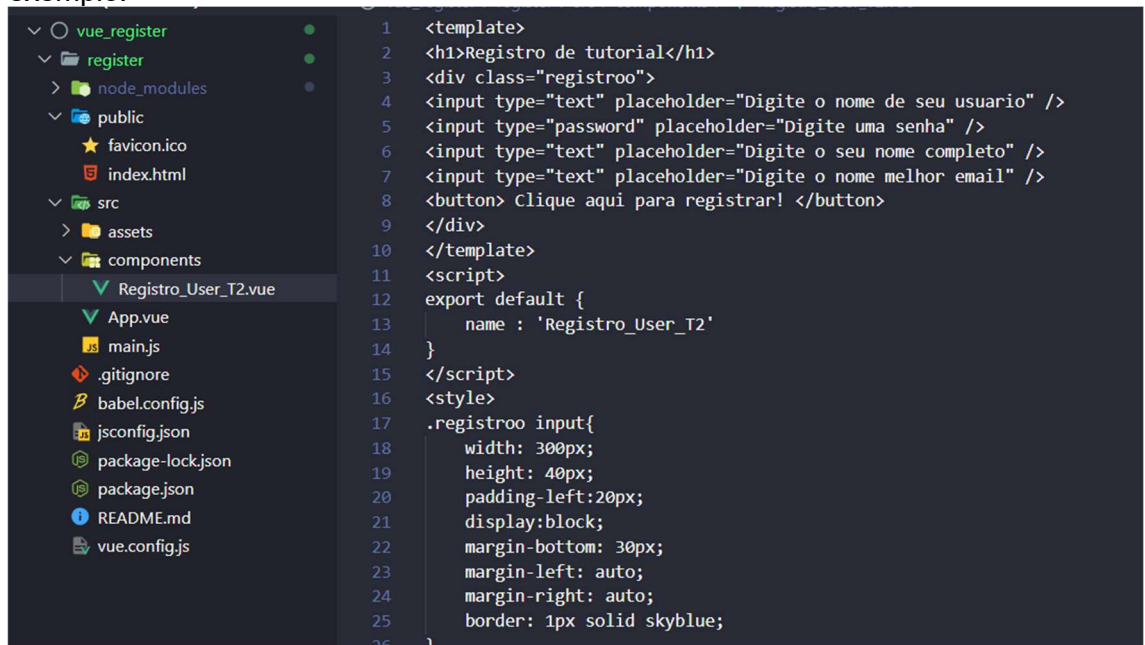
3. Configuração do projeto

- a. Navegue até o diretório onde deseja criar o projeto e execute o comando “vue create nome-do-projeto”

- b. 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
 - a. 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
 - a. execute o comando “npm run serve” para iniciar o servidor de desenvolvimento.
 - b. No navegador, acesso o endereço <http://localhost:8080> para visualizar a aplicação Vue em execução.

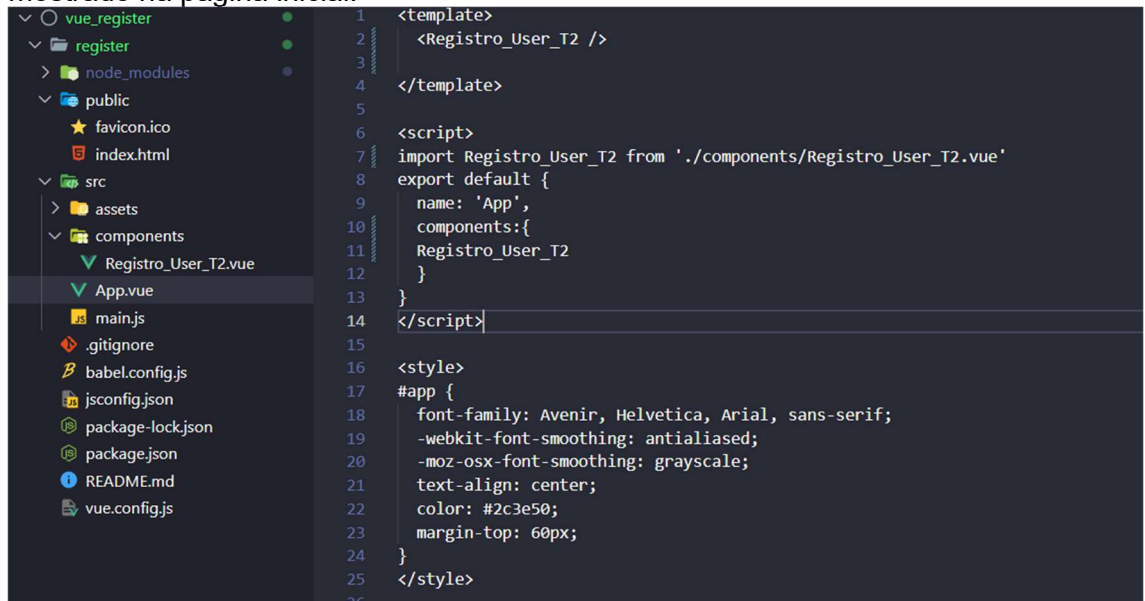
Criando uma aplicação web básica de formulário com VueJS

1. Criando projeto
 - a. 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.
 - b. 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.
 - c. 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.
 - d. 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.
 - e. Dentro de tags chamadas <template>, instanciamos o nosso formulário de saída para após isso, ser instanciada em App.vue.
 - f. Segue abaixo o pseudocódigo de como é criado um formulário de exemplo.



```
1 <template>
2 <h1>Registro de tutorial</h1>
3 <div class="registro">
4   <input type="text" placeholder="Digite o nome de seu usuario" />
5   <input type="password" placeholder="Digite uma senha" />
6   <input type="text" placeholder="Digite o seu nome completo" />
7   <input type="text" placeholder="Digite o nome melhor email" />
8   <button> Clique aqui para registrar! </button>
9 </div>
10 </template>
11 <script>
12 export default {
13   name : 'Registro_User_T2'
14 }
15 </script>
16 <style>
17 .registro input{
18   width: 300px;
19   height: 40px;
20   padding-left:20px;
21   display:block;
22   margin-bottom: 30px;
23   margin-left: auto;
24   margin-right: auto;
25   border: 1px solid skyblue;
26 }
```

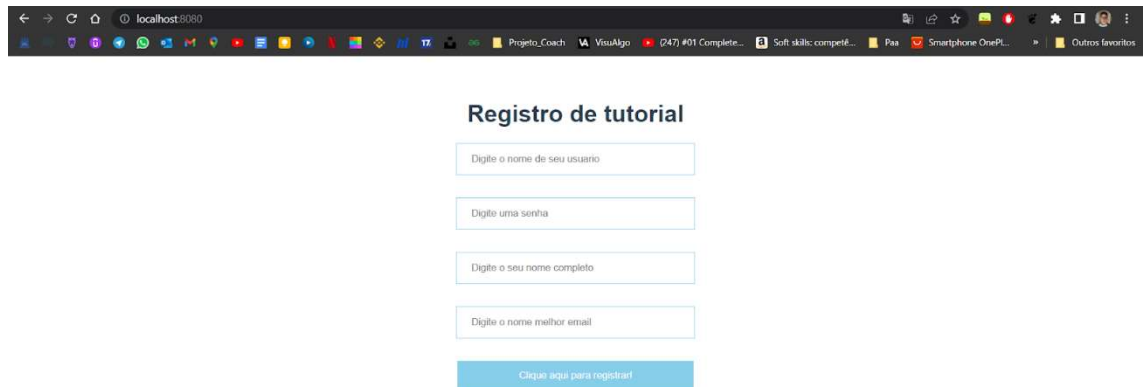
- g. Agora iremos para o APP.vue instanciar o Registro_User_T2.vue para ser mostrado na página inicial:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'register', 'node_modules', 'public', and 'src'. The 'src' folder contains 'assets', 'components', and 'App.vue'. The 'components' folder contains 'Registro_User_T2.vue'. The code editor shows the content of 'App.vue'.

```
1 <template>
2   <Registro_User_T2 />
3
4 </template>
5
6 <script>
7   import Registro_User_T2 from './components/Registro_User_T2.vue'
8   export default {
9     name: 'App',
10    components:{
11      Registro_User_T2
12    }
13  }
14 </script>
15
16 <style>
17 #app {
18   font-family: Avenir, Helvetica, Arial, sans-serif;
19   -webkit-font-smoothing: antialiased;
20   -moz-osx-font-smoothing: grayscale;
21   text-align: center;
22   color: #2c3e50;
23   margin-top: 60px;
24 }
25 </style>
26
```

- h. Com isso, a pagina web abaixo será exibida com um formulário para cadastro de usuários:

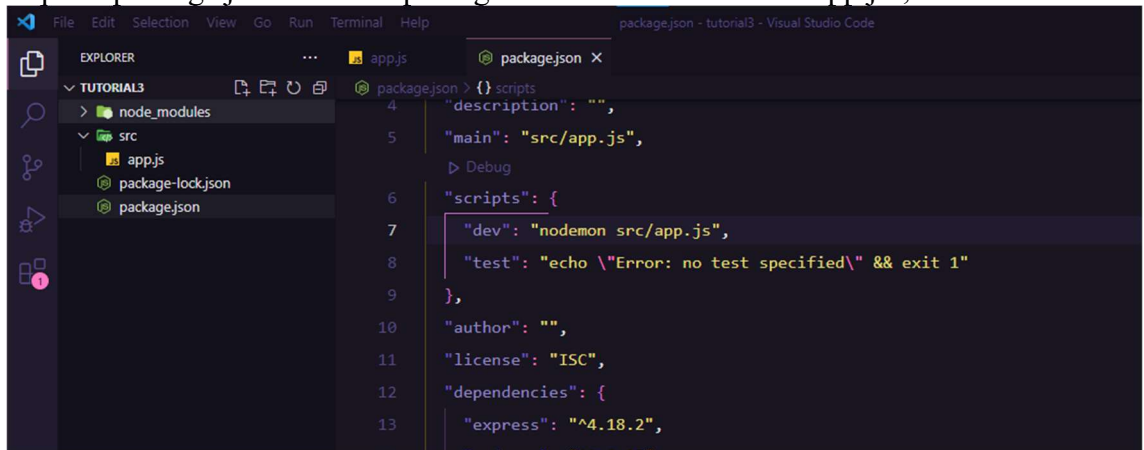


The screenshot shows a web browser displaying a registration form titled "Registro de tutorial". The form has four input fields for user registration: "Digite o nome de seu usuario", "Digite uma senha", "Digite o seu nome completo", and "Digite o nome melhor email". Below the input fields is a blue button labeled "Clique aqui para registrar!".

Como Instalar e Configurar o Banco de Dados

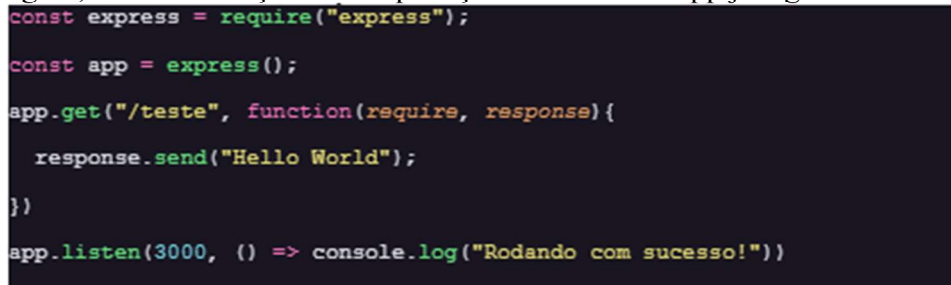
1. Crie uma pasta para o seu projeto
2. Execute o comando “npm init -y” para inicializar um novo projeto Node.js com as configurações padrão.
3. Instale o express: npm install express ou npm i express
4. Crie uma pasta src no seu projeto e logo em seguida crie o arquivo app.js dentro de src
5. vamos 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",.



```
package.json - tutorial3 - Visual Studio Code
package.json { } scripts
4   "description": "",
5   "main": "src/app.js",
6   "scripts": {
7     "dev": "nodemon src/app.js",
8     "test": "echo \\Error: no test specified\\ && exit 1"
9   },
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "express": "^4.18.2",
```

7. agora, vamos começar nossa aplicação de teste. Em app.js digite:



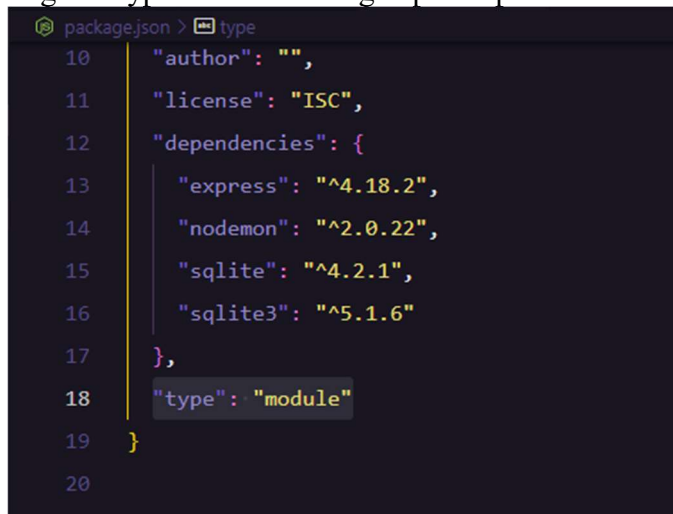
```
const express = require("express");

const app = express();

app.get("/teste", function(require, response){
  response.send("Hello World");
})

app.listen(3000, () => console.log("Rodando com sucesso!"))
```

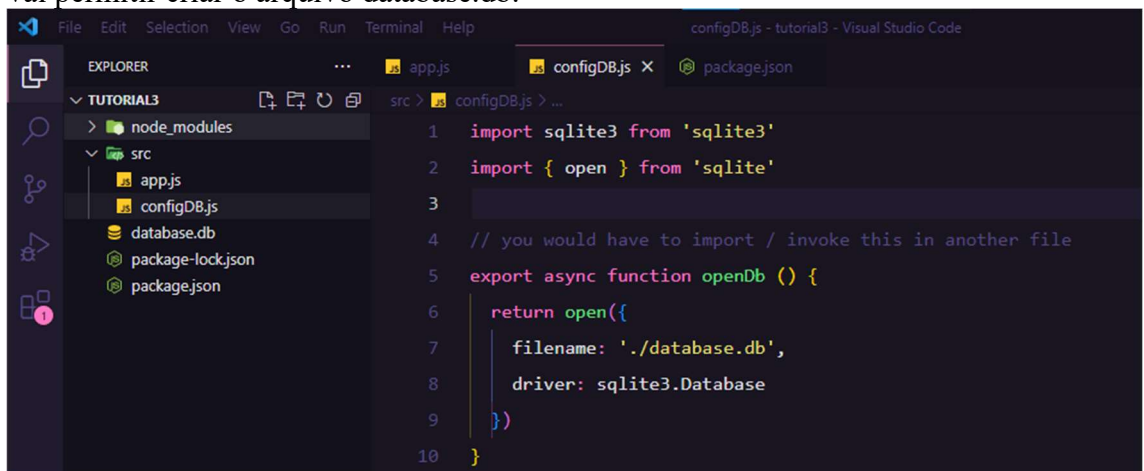
8. Para executar a aplicação de teste digite: npm run dev, se exibir “Rodando com sucesso!” está tudo ok.
9. localhost:3000/teste, se exibir “Hello World” está tudo ok.
10. Instalando o Sqlite
- Digite no seu terminal do projeto: npm install sqlite3 –save
 - Em seguida: npm install sqlite –save
11. Para trabalhar com importação de módulos (para que o script possa importar outros arquivos JS), vamos especificar no package.json
- Digite: "type": "module" logo após dependencies



```
package.json > type
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "express": "^4.18.2",
14    "nodemon": "^2.0.22",
15    "sqlite": "^4.2.1",
16    "sqlite3": "^5.1.6"
17  },
18  "type": "module"
19  }
20
```

- b.

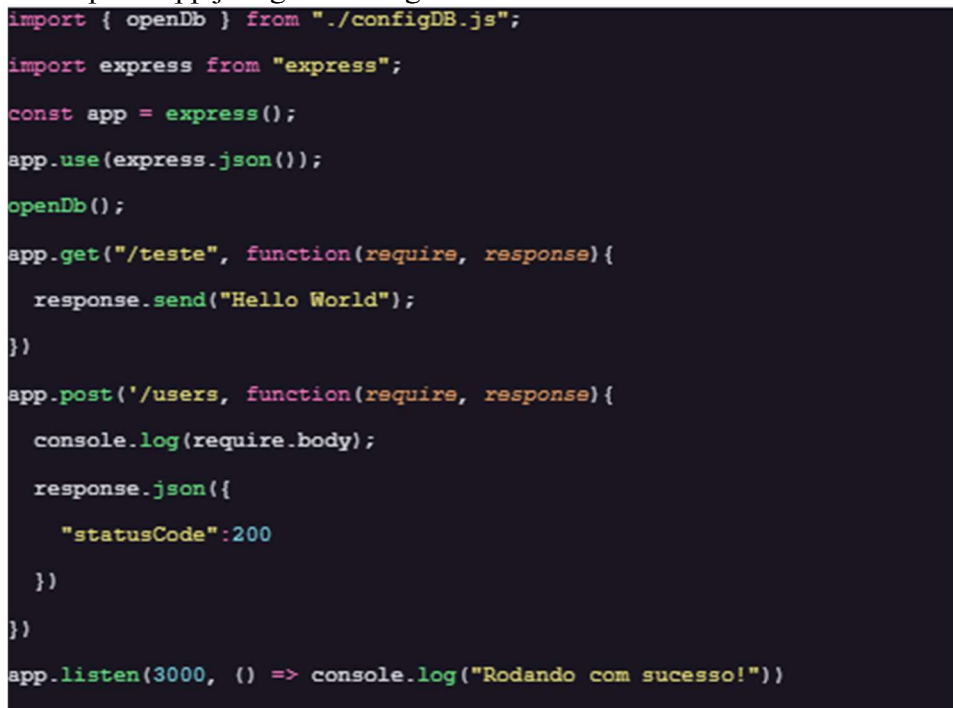
12. vamos criar dentro da pasta src o arquivo configDB.js e inserir o código que nos vai permitir criar o arquivo database.db.



```
1 import sqlite3 from 'sqlite3'
2 import { open } from 'sqlite'
3
4 // you would have to import / invoke this in another file
5 export async function openDb () {
6   return open({
7     filename: './database.db',
8     driver: sqlite3.Database
9   })
10 }
```

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.

a. No arquivo app.js digite o código:



```
import { openDb } from "../configDB.js";
import express from "express";

const app = express();

app.use(express.json());

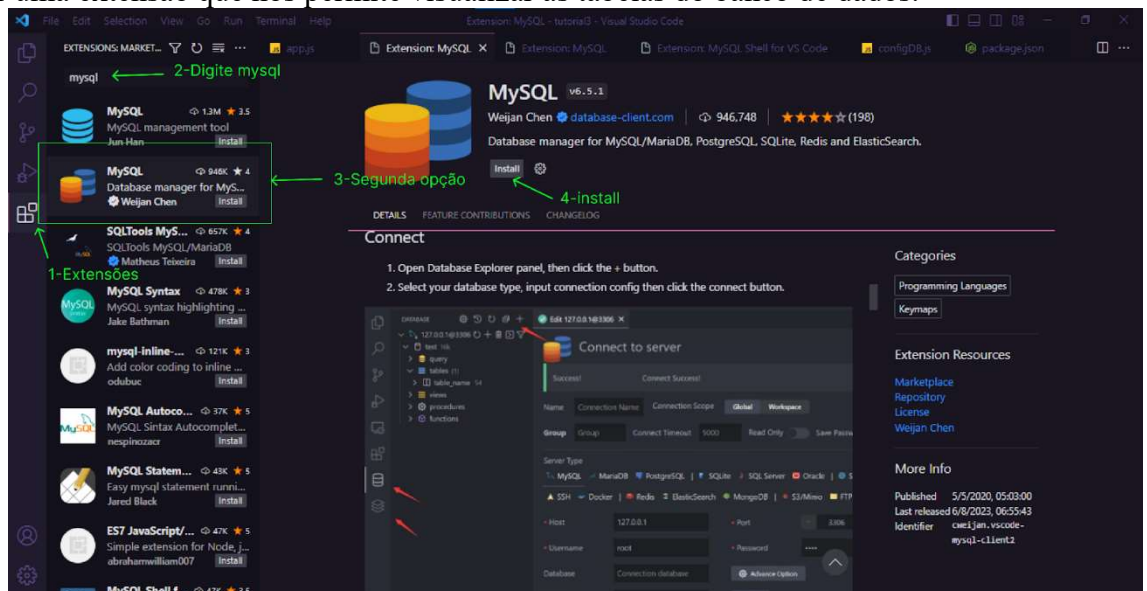
openDb();

app.get("/teste", function(require, response){
  response.send("Hello World");
})

app.post('/users, function(require, response){
  console.log(require.body);
  response.json({
    "statusCode":200
  })
})

app.listen(3000, () => console.log("Rodando com sucesso!"))
```

14. instalar uma extensão que nos permite visualizar as tabelas do banco de dados.



1-Extensões

2-Digite mysql

3-Segunda opção

4-install

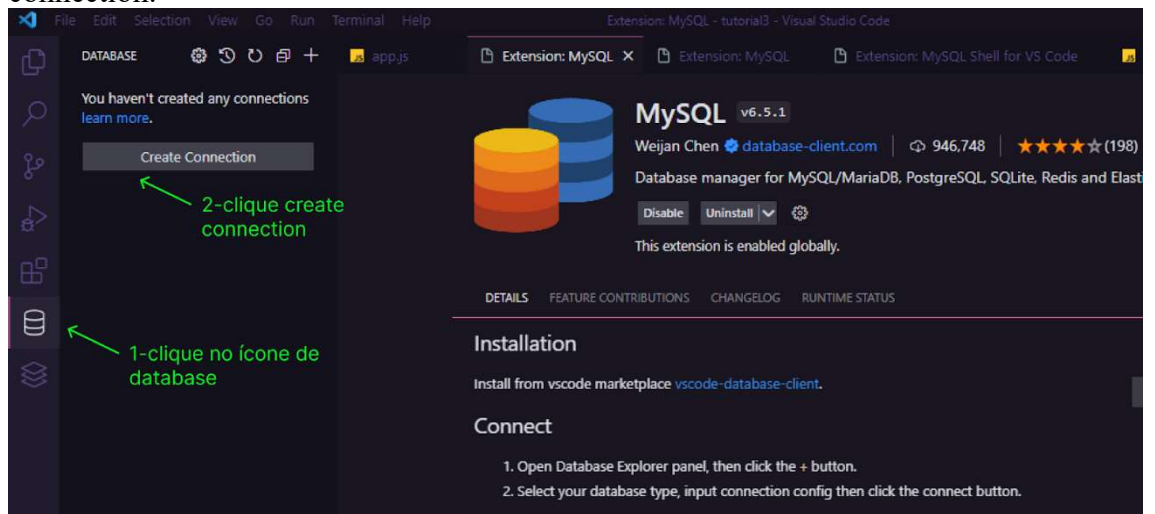
1. Open Database Explorer panel, then click the + button.

2. Select your database type, input connection config then click the connect button.

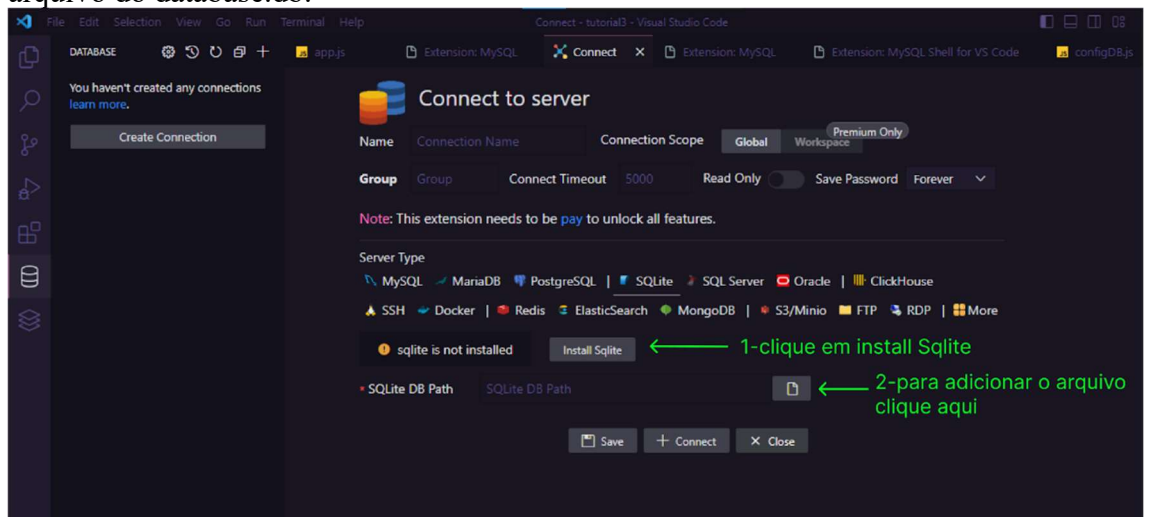
a.

15. como usar a extensão mysql.

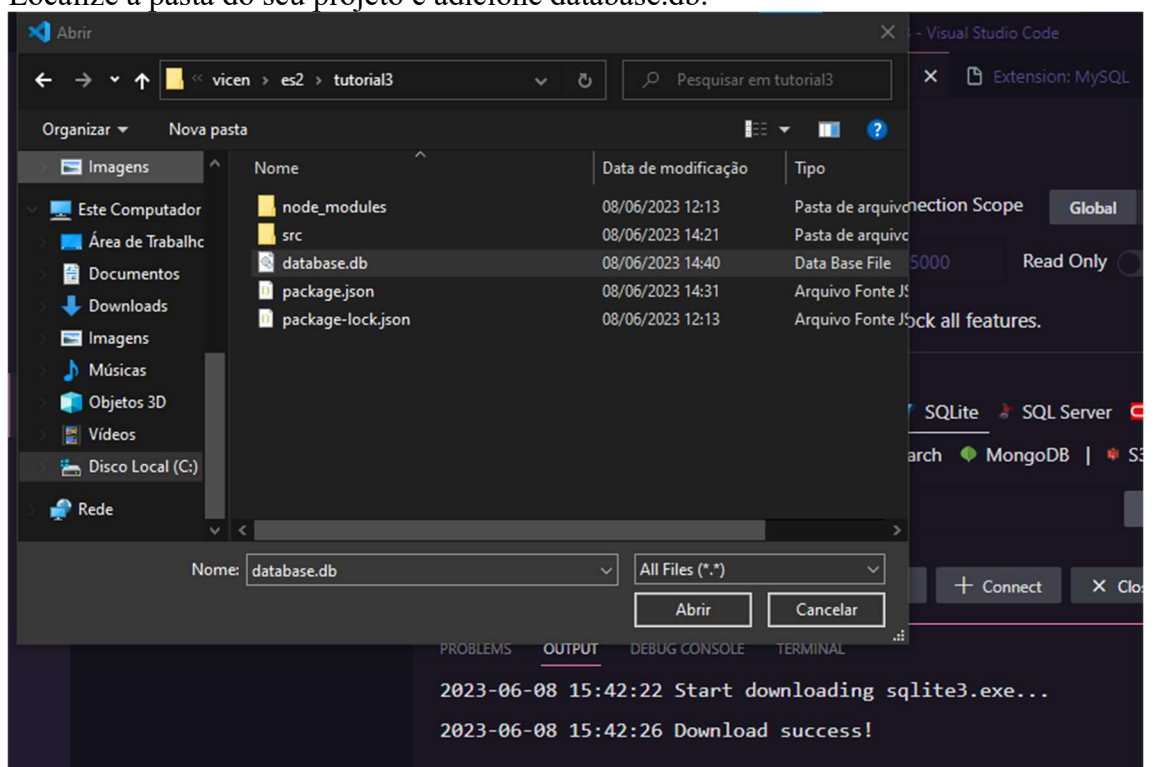
- a. Clique no ícone de database na barra à esquerda e depois em create connection.



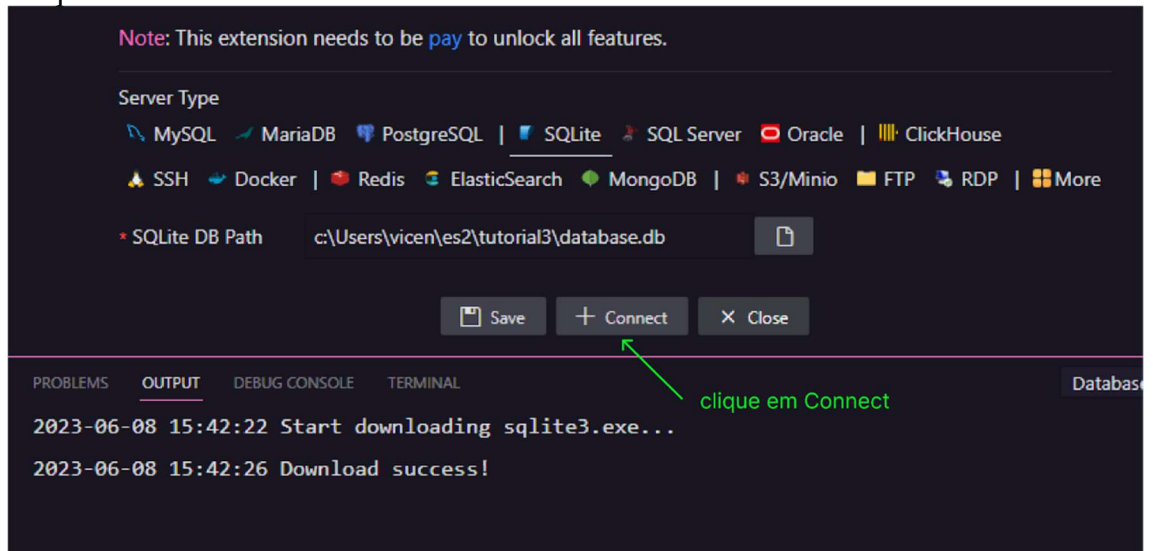
- b. Clique em install Sqlite e depois em SQLite DB path para adicionar o arquivo do database.db.



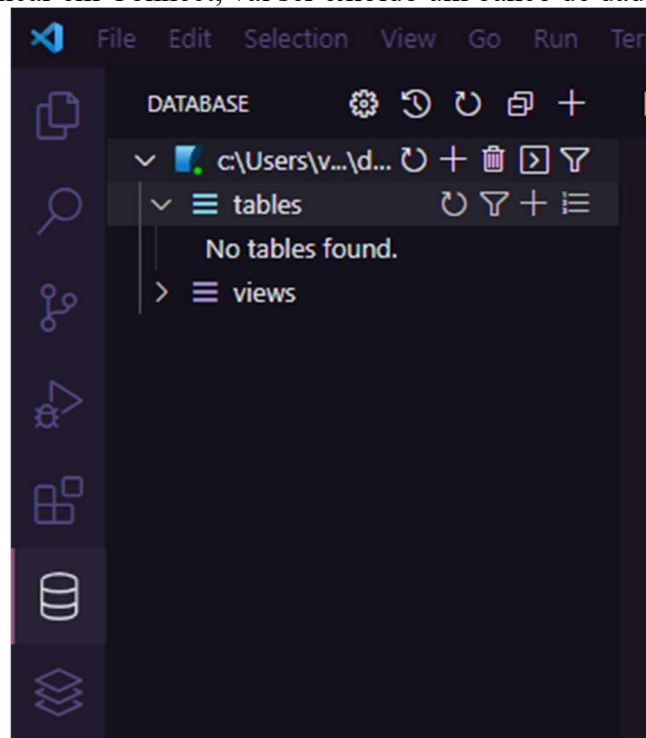
- c. Localize a pasta do seu projeto e adicione database.db.



d. Clique em Connect.



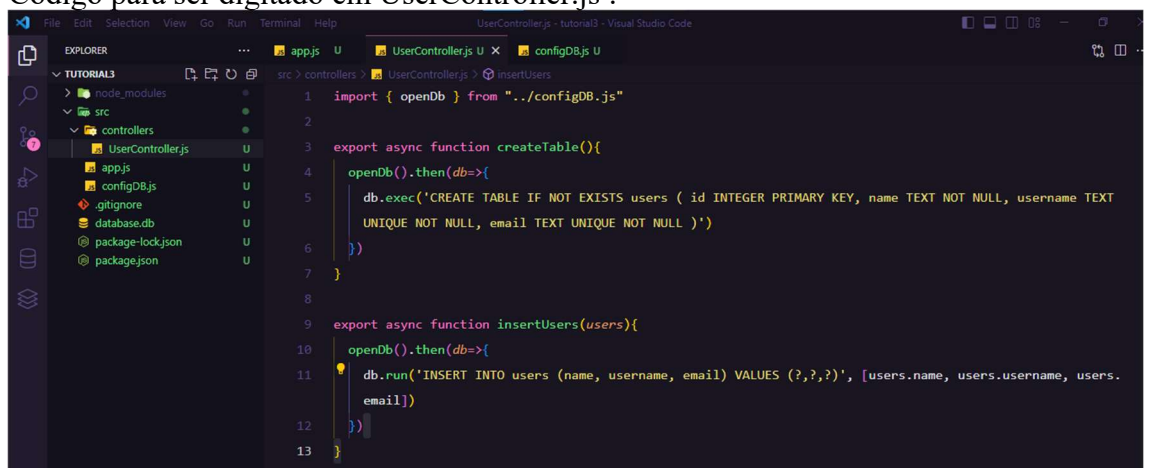
e. Após clicar em Connect, vai ser exibido um banco de dados sem



tabelas.

16. vamos criar uma pasta controllers dentro de src e um arquivo UserController.js dentro da pasta controller(src/controller/UserController.js).

a. Código para ser digitado em UserController.js :



17. vamos fazer algumas mudanças no código de app.js para podermos importar as funcionalidades do arquivo UserController.js.

- a. Código para ser digitado em app.js :

```
import { createTable, insertUsers } from
"./controllers/UserController.js";
import express from "express";
const app = express();
app.use(express.json());
createTable();
app.post('/users', function(require, response){
    insertUsers(require.body);
    response.json({
        "statusCode":200
    })
})
app.listen(3000, () => console.log("Rodando com sucesso!"))
```

18. 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.id;

            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);

});
```

3. 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.

```
4. let sqlite3 = require('sqlite3').verbose();
5.
6. let db = new sqlite3.Database('./sqlite.db');
7.
8. let init = function () {
9.   db.run("CREATE TABLE if not exists usuario (" +
10.     "id INTEGER PRIMARY KEY AUTOINCREMENT," +
11.     " nome TEXT," +
12.     " email TEXT," +
13.     " numero INT," +
14.     " usuario TEXT" +
15.     ")");
16. };
17.
18. module.exports = {
19.   init: init,
20.   db: db
21. };
22.
```

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 Usuário</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 caixa nome" type="number" name="nome">

</input>

</div>

<div>

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

</input>

</div>

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

</input>

</div>

<div>

<div>

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

</input>

</div>

<div>

<input class="cadastrar caixa Enviar" 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
```

2. Instale as dependências necessárias:

```
$ npm install express multer --save
```

3. 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 osarquivos.'
  });
  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
```

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

```
$ vue create image-upload
```

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

```
$ cd image-upload
```

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

```
$ npm install axios --save
```

6. Substitua o conteúdo do arquivo **src/components/HelloWorld.vue** como 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">
        
      </li>
    </ul>
  </div>

```

```
</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 {
```



TUTORIAL DAS FERRAMENTAS

```
        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