

## TRILHAS

### Trilhas Necessárias Anterior à Maratona

[Guia Estelar de Programação](#) OK

[Equipando sua nave](#) OK

[Guia Estelar de HTTP](#) OK

Guia Estelar de Git

Guia Estelar de GitHub

Guia Estelar de JavaScript

NodeJS

NodeJS com EJS

NodeJS com SQLite

### Material Complementar

<https://www.notion.so/Maratona-Discover-59f97d2dc9d54e89a484741d67930cbb>

### Trilha de Front-End

<https://www.youtube.com/playlist?list=PLLeLKux5eT3kY2mvZUi7IM5T548vfKxGq5>

## NODE

- Instalar Node.js
  - Link para download: <https://nodejs.org/en/>
- O que faz?:
  - Ele conversa com a nossa máquina, por exemplo: se precisarmos abrir um arquivo através de um programa é preciso um ambiente para que o programa converse com nosso ambiente do sistema operacional. Ao utilizarmos o arquivo de front-end index.html, estamos fazendo isso - pegando um arquivo na máquina e quem faz isso é o Node.js;
  - Ele é construído em JavaScript, portanto, vamos dar comandos para o Node em JavaScript para que o Node converse com a máquina.
- O que compõe o Node?:
  - Com o Node eu estou controlando a minha máquina via javascript e para isso o Node se conecta ao motor do Chrome para rodar o javascript na minha máquina e eu só rodo javascript na minha máquina se eu tiver o node.

## GIT BASH

- Instalar Git para windows
  - Link para download: <https://gitforwindows.org/>

## BEEKEEPER

- O que é: Não é um banco de dados é um programa que exibe o que tem no nosso arquivo de banco de dados
- Onde baixar: <https://www.beekeeperstudio.io/>

## INSTALANDO FIRACODE

- O que é: É uma extensão da fonte Fira Mono que contém um conjunto de ligaduras para combinações de vários caracteres de programação comuns. Este é apenas um recurso de renderização de fonte: o código subjacente permanece compatível com ASCII. Isso ajuda a ler e entender o código mais rapidamente.
- Como instalar no VS Code:  
<https://blog.rocketseat.com.br/terminal-com-oh-my-zsh-spaceship-dracula-e-mais/>

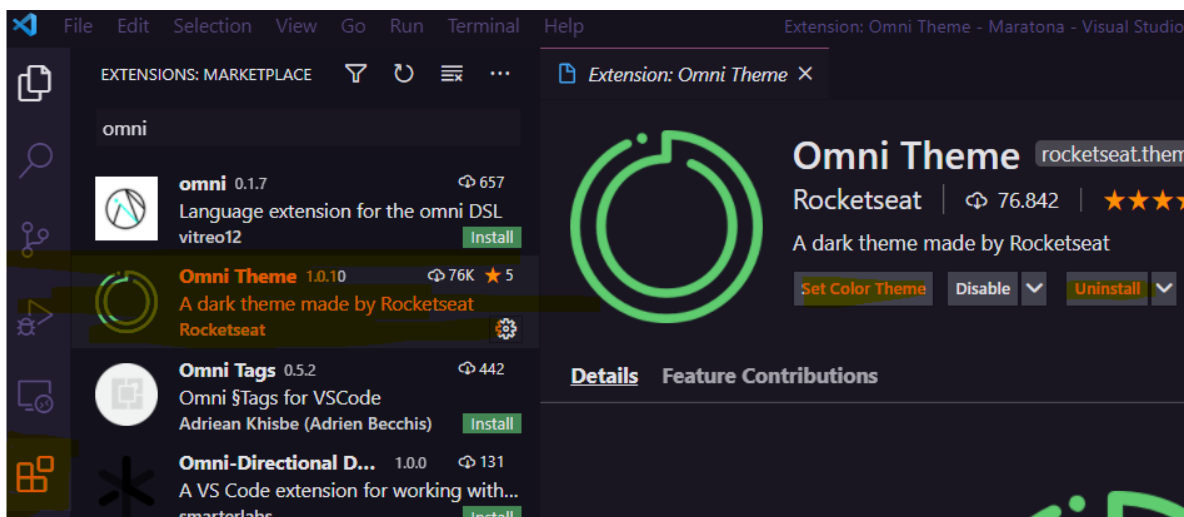
## VS CODE

### Download

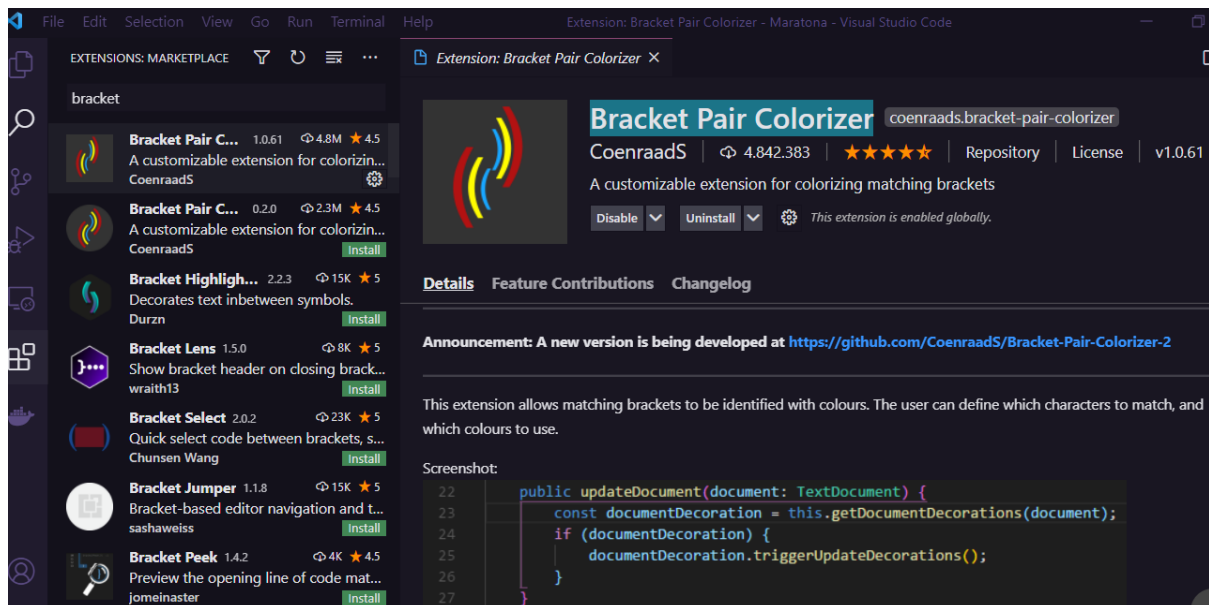
- Download VS Code: <https://code.visualstudio.com/download>

### Temas e Extensões

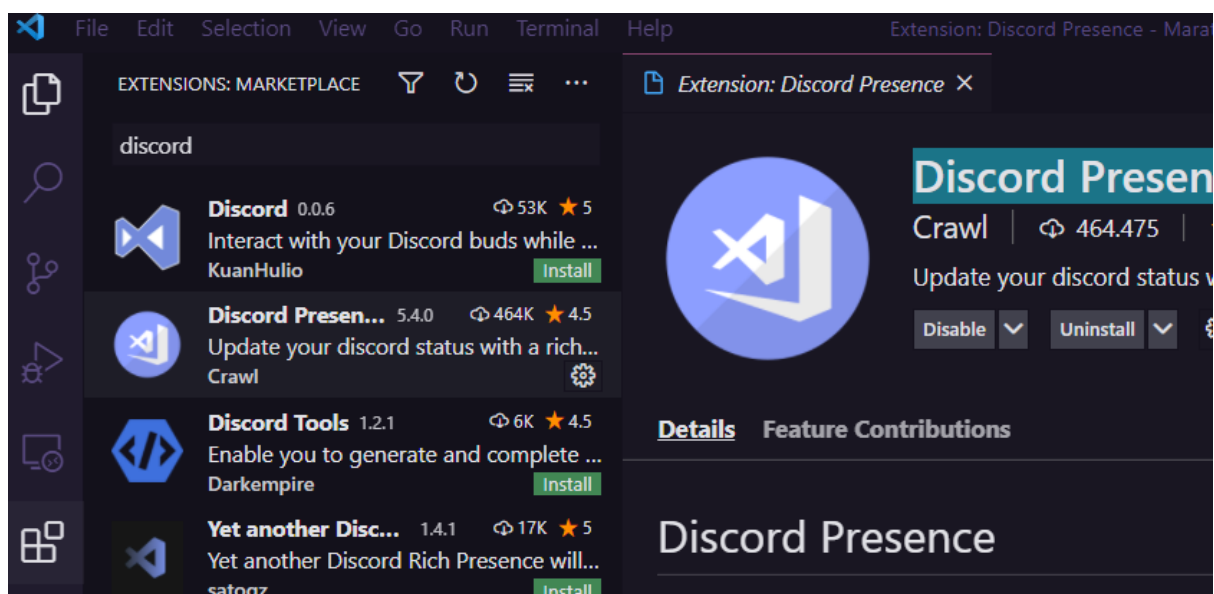
- Omni Theme: para deixar seu VS Code com uma cor mais bonita;
  - Clique em extensões;
  - Busque por Omni Theme;
  - Instale;
  - Clique em Set Color Theme.



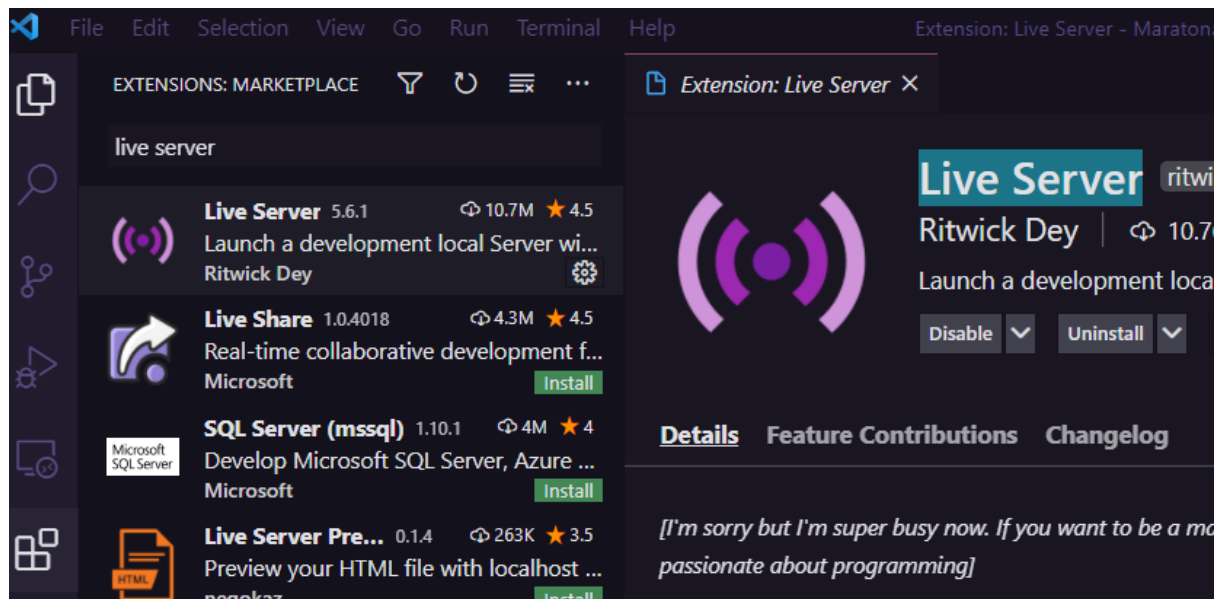
- Bracket Pair Colorizer: cores diferentes no código para saber o que eu abri e em que momento eu fechei;
  - Clique em extensões;
  - Busque por Bracket Pair Colorizer;
  - Instale.



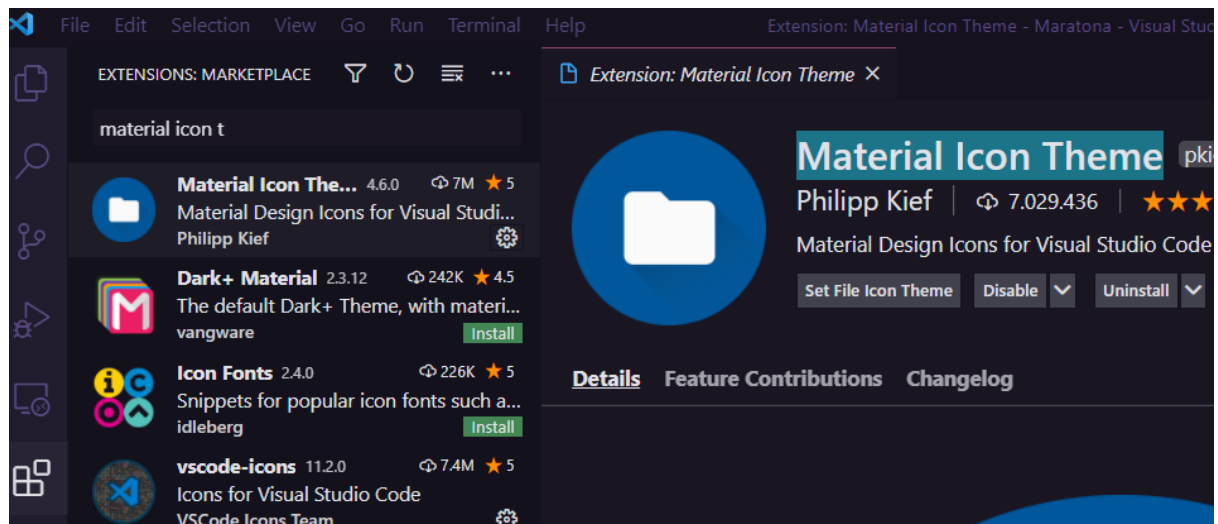
- Discord Presence: mostra no Discord quando estamos codando;
  - Clique em extensões;
  - Busque por Discord Presence;
  - Instale.



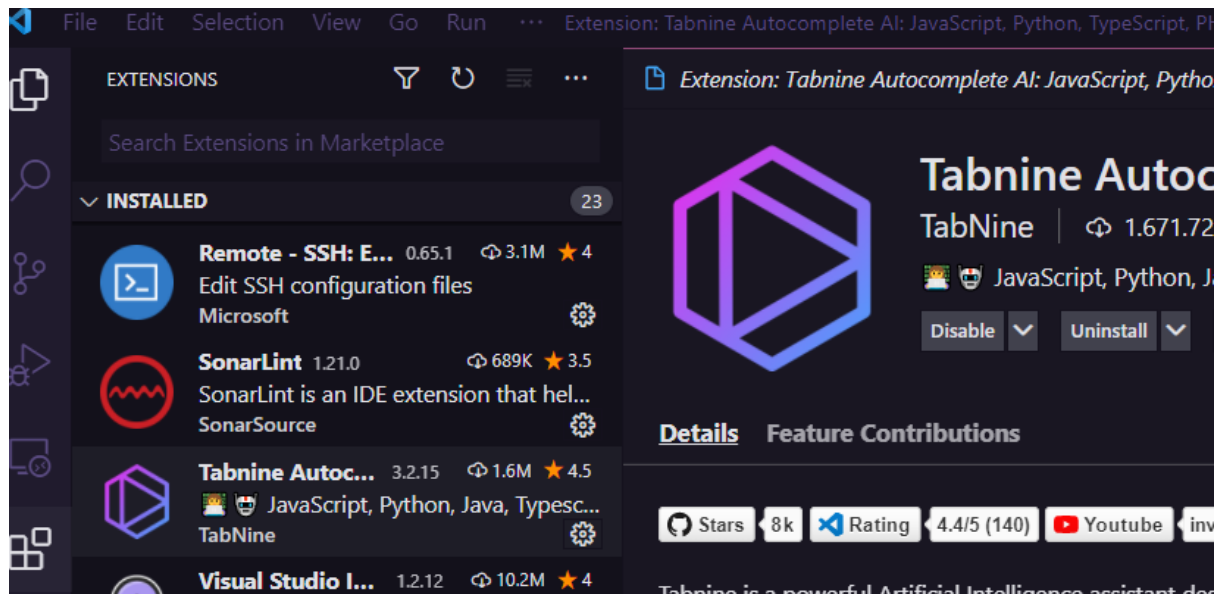
- Live Server: servidor do próprio .NET;
  - Clique em extensões;
  - Busque por Live Server;
  - Instale.



- Material Icon Theme: ele deixa os diretórios e arquivos com os ícones da linguagem utilizada;
  - Clique em extensões;
  - Busque por Live Server;
  - Instale;
  - Set File Icon Theme.



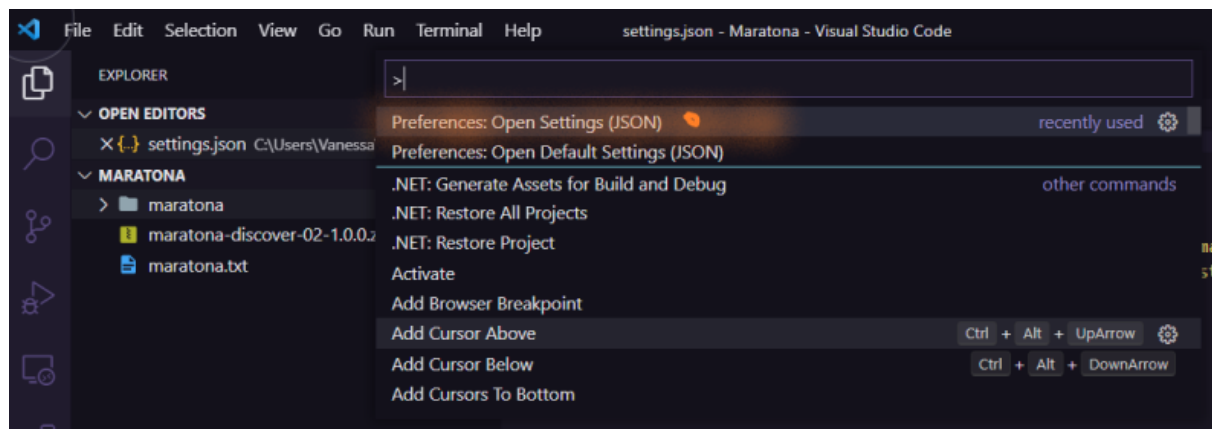
- Tabnine Autocomplete AI: JavaScript, Python, TypeScript, PHP, C/C++, HTML/CSS, Go, Java, Ruby, C#, Rust, SQL, Bash, Kotlin, Julia, Lua, OCaml, Perl, Haskell, and React.;
  - Clique em extensões;
  - Busque por Tabnine Autocomplete AI;
  - Instale.



- EJS language support (dá dicas de como usar o EJS):
  - Clique em extensões;
  - Busque por EJS language support;
  - Instale.

## Configurações

- CTRL+Shift+P;
- Selecione Preferences Open Settings (JSON);



- Altere o texto para:

```
//tamanho da font
"editor.fontSize": 16,
//barra rodapé
"workbench.statusBar.visible": false,
//barra lateral direita
"editor.minimap.enabled": false,
//barra lateral esquerda
"workbench.activityBar.visible": true,
//tema
"workbench.colorTheme": "Omni",
//ícones pastas e arquivos
"workbench.iconTheme": "material-icon-theme",
```

```
//Fonte
"editor.fontFamily": "Fira Code, Menlo, Monaco, 'Courier New', monospace",
//tamanho da font no terminal
"terminal.integrated.fontSize": 16,
//autocomplete
"tabnine.experimentalAutoImports": true,
//terminal
"terminal.integrated.shell.windows": "C:\\Program Files\\Git\\bin\\bash.exe",
```

## Atalhos

- Ctrl+K M: selecionar a linguagem, exemplo c#
- SHIFT + ALT + F: alinhar código
- CTRL + Shift + P e Selecione Preferences Open Settings (JSON): documento para configuração de preferências;
- SHIFT + B: abre e fecha a barra da lateral esquerda
- CTRL + Shift + X: abre e fecha a aba de extensões
- CTRL + Shift + E: abre a aba de componentes do projeto
- CTRL + F: pesquisa
- ALT + Shift + Seta pra baixo: copia a linha e cola pra baixo
- CTRL + C: para o terminal
- CTRL + L: limpa o terminal

## CRIANDO NOSSO PRÓPRIO SERVIDOR E AJUSTANDO A APLICAÇÃO

- Abrir o terminal;
- Certifique-se de que o terminal está apontando para o caminho correto da sua aplicação. Exemplo: Vanessa@VanessaNote MINGW64  
/e/CURSOS/Maratona/maratona
- NPM: permite utilizarmos coisas/bibliotecas/pacotes que alguém já construiu. Nesse caso vamos utilizar um pacote que alguém já construiu para criação de um servidor;
- npm init -y: iniciando o meu projeto. O -y serve para pularmos algumas perguntas que surgem durante a inicialização do projeto. Esse comando irá criar o arquivo package.json, cujo contém informações do nosso projeto, como: nome, versão, descrição, main(página principal), autor, tipo de licença, etc...;
- npm i express: irá instalar os pacotes para criarmos o nosso servidor. Esses pacotes ficarão na pasta node\_modules e nosso arquivo package-lock.json é atualizado com informações de dependências de do pacote express;
- criar uma pasta src;
- criar dentro da pasta src o arquivo server.js com o conteúdo abaixo:

```
//função require: estou chamando o pacote express da nossa pasta node_modules
const express = require("express")

//criando o meu servidor executando o pacote do express
const server = express()

//acessando a página index.
//Como ainda não criamos as rotas, precisamos setar a página index através do request,
response
```

```
//abaixo enviamos apenas um Oi para testar
```

```
server.get('/', (requeste, response) => {  
  return response.send('Oi')  
})
```

```
//Funcionalidade do express para simular o nosso servidor em uma determinada porta  
server.listen(3000, () => console.log('rodando'))
```

- node src/server.js: executar o nosso servidor;
- npm i nodemon -D: é uma dependência de desenvolvimento, ou seja, só funciona no ambiente de DEV. Sua função é ficar olhando o projeto e reinicia sempre que alguma coisa for alterada. Isso evita termos que executar o comando anterior para executar o servidor. Nosso arquivo package.json é atualizado com informações de dependências de do pacote nodemon;
- Altere o arquivo package.json para que o nodemon seja executado.:

ANTES	DEPOIS
<pre>{   "name": "maratona",   "version": "1.0.0",   "description": "Freelance Calculator",   "main": "index.js",   "scripts": {     "teste": "echo \ "Error: no test specified\ " &amp;&amp; exit 1"   },   "keywords": [],   "author": "",   "license": "MIT",   "dependencies": {     "ejs": "^3.1.6",     "express": "^4.17.1"   },   "devDependencies": {     "nodemon": "^2.0.7"   } }</pre>	<pre>{   "name": "maratona",   "version": "1.0.0",   "description": "Freelance Calculator",   "main": "src/server.js",   "scripts": {     "dev": "nodemon ."   },   "keywords": [],   "author": "",   "license": "MIT",   "dependencies": {     "ejs": "^3.1.6",     "express": "^4.17.1"   },   "devDependencies": {     "nodemon": "^2.0.7"   } }</pre>

*\*Se não alterarmos este arquivo teremos que ir com o prompt de comando/terminal até a pasta onde está o nodemon (node\_modules) para podermos executá-lo*

*\*\*Esse comando **"dev": "nodemon ."** serve para executar comandos que estão em nosso terminal. Por ele ter acesso a pasta node\_modules eu posso apontar pro que eu quero dentro dela através do comando **"main": "src/server.js"**, - > arquivo principal do meu servidor.*

- npm run dev: agora estamos executando nosso servidor através do nodemon e qualquer alteração que fizermos no projeto (nos arquivos com extensão js, mjs e json) será refletida na execução do servidor;

## CRIANDO NOSSO BANCO DE DADOS

- Abra o terminal;

- npm install sqlite3
- npm install sqlite
- Crie uma pasta chamada db dentro de src;
- Crie dois arquivos dentro da pasta db com os seguintes nomes e códigos:

config.js

```
//este arquivo é a ponte entre o projeto e o banco de dados

//importante o sqlite3
const sqlite3 = require('sqlite3')
//ao invés de importar tudo do sqlite importo apenas a função open - única que
vamos usar
const { open } = require('sqlite')

//abrindo a conexão com o banco de dados
//o open() por regra deve estar dentro de uma função, por isso estamos utilizando
ele da
//forma abaixo e não simplesmente exportando o objeto open:
//module.exports = open(){}
module.exports = () => open({
  //arquivo onde as informações ficarão armazenadas
  filename: './database.sqlite',
  //driver é quem processa e armazena no arquivo os dados
  driver: sqlite3.Database
});
```

init.js

```
//esse arquivo serve para colocar algumas configurações para o banco de dados
//ele roda uma única vez
//ele é quem cria o arquivo do banco de dados
//resumindo, ele cria as tabelas

//importando o arquivo config
const Database = require('./config')

//async / await = espera o database terminar de inicializar para depois fazermos a
//próxima coisa - isso deve ser usado aqui pq o javascript não fica olhando quando
//o database termina ou não de inicializar
//e toda vez que usarmos o await temos que usar o async, pois, este diz que tudo o
```



que está

//dentro de um await deve ser esperado seu processamento

```
const initDb = {
```

```
  async init(){
```

```
    //iniciando a conexão com o banco de dados - Database()
```

```
    //observe que estamos armazenando o resultado da conexão aberta para  
podermos seguir
```

```
    //para os próximos comandos
```

```
    const db = await Database()
```

```
    //observe que não estamos armazenando o resultado da execução para  
podermos seguir
```

```
    //para os próximos comandos, porque não é necessário
```

```
    //pegar o sql que será passado pra exec e executá-lo no banco de dados
```

```
    await db.exec(`CREATE TABLE profile(  
      id INTEGER PRIMARY KEY AUTOINCREMENT,  
      name TEXT,  
      avatar TEXT,  
      monthly_budget INT,  
      days_per_week INT,  
      hours_per_day INT,  
      vacation_per_year INT,  
      value_hour INT  
    )`);
```

```
    await db.exec(`CREATE TABLE jobs(  
      id INTEGER PRIMARY KEY AUTOINCREMENT,  
      name TEXT,  
      daily_hours INT,  
      total_hours INT,  
      created_at DATETIME  
    )`);
```

```
    await db.run(`  
    INSERT INTO profile(  
      name,
```

```
    avatar,  
    monthly_budget,  
    days_per_week,  
    hours_per_day,  
    vacation_per_year,  
    value_hour  
  ) VALUES (  
    "Ichikawa",  
    "https://avatars.githubusercontent.com/u/34397262?v=4",  
    3000,  
    5,  
    5,  
    4,  
    70  
  );` )
```

```
await db.run(`  
INSERT INTO jobs(  
  name,  
  daily_hours,  
  total_hours,  
  created_at  
)  
VALUES (  
  "Pizzeria Guloso",  
  2,  
  1,  
  1617514376018  
);` )
```

```
await db.run(`  
INSERT INTO jobs(  
  name,  
  daily_hours,  
  total_hours,  
  created_at  
)  
VALUES (  
  "OneTwo Projects",  
  3,  
  47,
```

```

        1617514376018
    );` )

    //fechando a conexão com o banco de dados
    await db.close()
  }
}

//executando o init()
initDb.init()

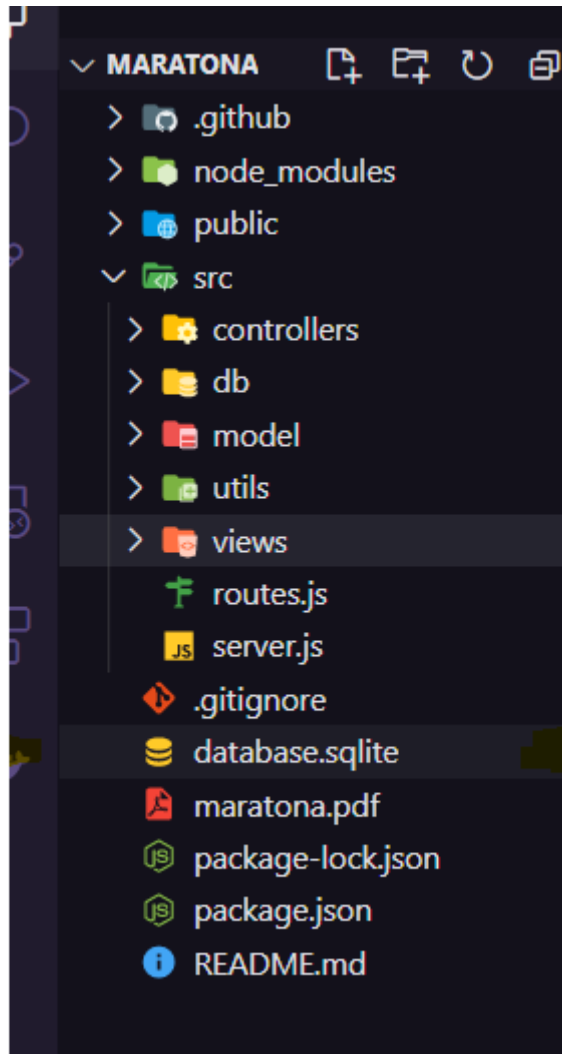
```

- Altere o arquivo package.json

DE	PARA
<pre> {   "name": "maratona",   "version": "1.0.0",   "description": "Freelance Calculator",   "main": "src/server.js",   "scripts": {     "dev": "nodemon .",   },   "keywords": [],   "author": "",   "license": "MIT",   "dependencies": {     "ejs": "^3.1.6",     "express": "^4.17.1",     "sqlite": "^4.0.21",     "sqlite3": "^5.0.2"   },   "devDependencies": {     "nodemon": "^2.0.7"   } } </pre>	<pre> {   "name": "maratona",   "version": "1.0.0",   "description": "Freelance Calculator",   "main": "src/server.js",   "scripts": {     "dev": "nodemon .",     "init-db": "node src/db/init.js"   },   "keywords": [],   "author": "",   "license": "MIT",   "dependencies": {     "ejs": "^3.1.6",     "express": "^4.17.1",     "sqlite": "^4.0.21",     "sqlite3": "^5.0.2"   },   "devDependencies": {     "nodemon": "^2.0.7"   } } </pre>

- Abrir o terminal;
- npm run init-db;

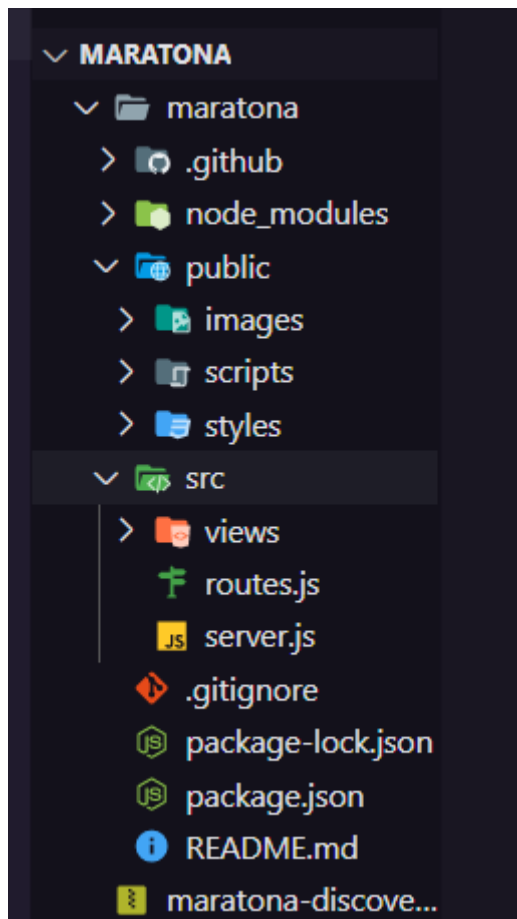
- Abra o Beekeeper;
- Selecione o tipo do Banco de dados, nesse caso, SQLite;
- Selecione o arquivo criado para o nosso banco de dados:



- Selecione o arquivo criado para o nosso banco de dados;
- Clique em Connect;
- Verifique se as tabelas foram criadas e se os dados inseridos inicialmente existe:

E:\CURSOS\Maratona\mar.		Query #1		jobs [all]	
Filter		id		equals	
ENTITIES 3		id		Enter Value	
> jobs		id integer		name text	
> profile		daily_hours int		total_hours int	
> sqlite_sequence		created_at datetime			
		1	Pizzaria Guloso	2	40
		2	OneTwo Projects	2	60
		5	Discover	9	0
					1617514376018
					1617514376018
					1617762809291

## ENTENDENDO A ESTRUTURA DO PROJETO



- .github:
  - Tudo o que tem . na frente do diretório se torna um diretório oculto;
  - Nele nós inserimos uma foto do projeto e a licença do projeto (MIT: use à vontade só dê os créditos);
  - Essa pasta só existe devido ao arquivo **README.md**, cujo ficará no GitHub para que ele forneça uma preview do nosso projeto.
- node\_modules:
  - Nesta pasta ficam todas as bibliotecas/ pacotes que estamos importando para dentro do nosso projeto, por exemplo, a biblioteca express.
- public:
  - Nesta pasta ficam todos os arquivos públicos.
- src:
  - É onde o nosso servidor irá ficar;
  - Dentro dessa pasta também inserimos tudo o que ficará alocado no servidor controllers, models, utils, views (MVC), routes e server (nosso servidor).
- .gitignore:
  - Também é um arquivo oculto;
  - Está escondendo uma pasta chamada node\_modules;
  - A pasta node\_modules vai ignorar os pontos da história do git (versionamento);
- package-lock.json: vai atualizando com os os pacotes que vamos instalando em nossa aplicação;

- package.json: contém informações do nosso projeto, como: nome, versão, descrição, main(página principal), autor, tipo de licença, dependências (pacotes/bibliotecas...).

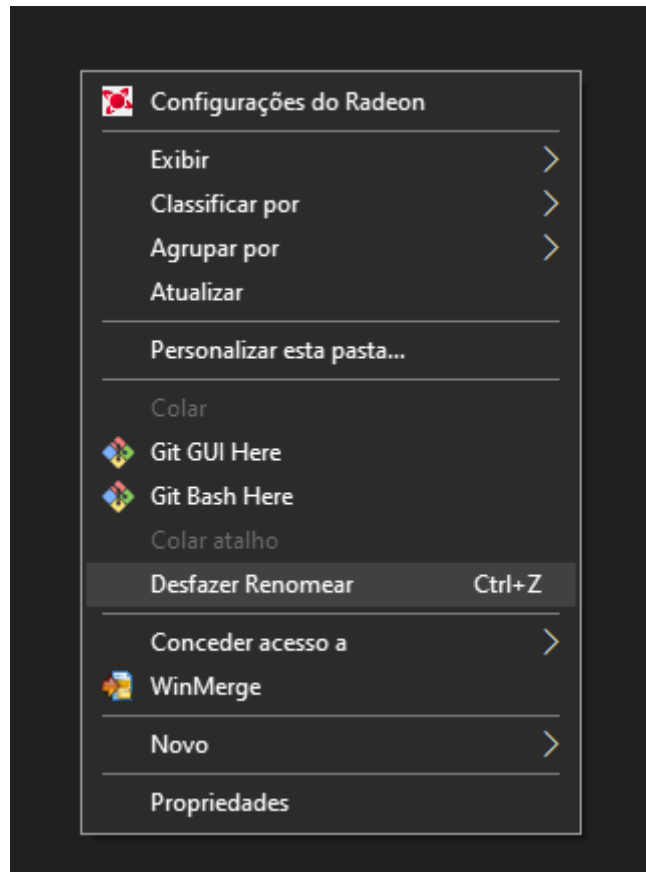
## ENTENDENDO A ESTRUTURA DO BANCO DE DADOS

### Banco de Dados

Profile	Job
id	id
name	name
avatar	daily-hours
monthly-budget	total-hours
days-per-week	created_at
hours-per-day	
vacation-per-year	
value-hour	

## ADICIONANDO NO GITHUB

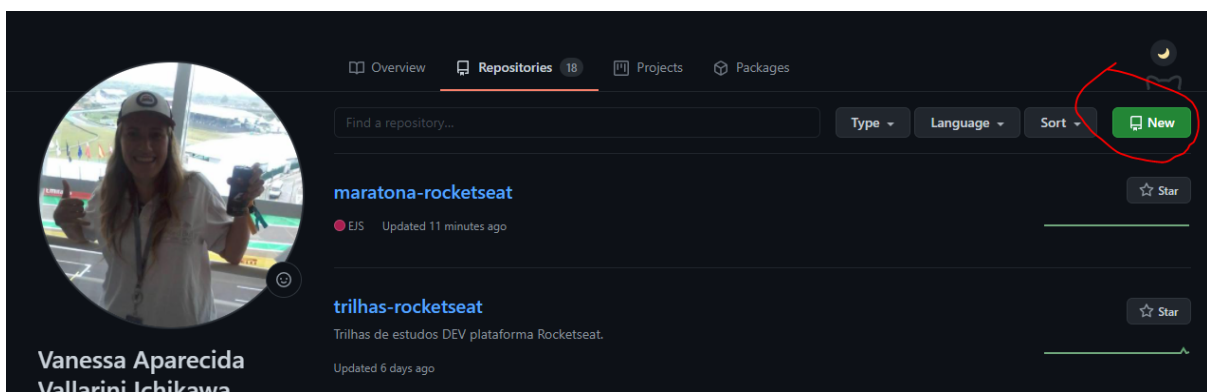
- Abra o git Bash no diretório do seu projeto, git bash here:



- Dê o comando para iniciar um repositório dentro do nosso projeto. Nesta etapa os arquivos estão no estado Working Directory :

```
$ git init
```


- Crie um diretório no GitHub com o nome de sua preferência, deixe público e clique em “Criar”:



# Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

 VanessaVallarini ▾

Repository name \*

/ nome ✓

Great repository names are short. nome is available. Need inspiration? How about **redesigned-giggle?**

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

## Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)


☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

**Create repository**



**Quick setup — if you've done this kind of thing before**

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/VanessaVallarini/nome.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# nome" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/VanessaVallarini/nome.git
git push -u origin main
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/VanessaVallarini/nome.git
git branch -M main
git push -u origin main
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

- De o seguinte comando para linkar o repositório da nossa máquina com o repositório criado no GitHub:

```
$ git remote add origin https://github.com/VanessaVallarini/maratona-rocketseat.git
```

*\*OBS: O link é o mesmo que irá aparecer após criar o seu repositório.*

- Dê o comando para mover os arquivos do estado Working Directory para o estado Stage Area (arquivo está preparado para ser comitado):

```
$ git add .
```

- Dê o comando para mover os arquivos do estado Stage Area para o estado Local Repository (arquivo está no repositório local e o ponto da história é criado e evitamos perder esse arquivo):

```
$ git commit
```

*\*OBS: O Git Bash irá pedir para que dê um nome para o arquivo, como recomendação para o primeiro commit do projeto utilize "initial commit".*

*\*\*OBS: Para sair da tela digite: ESC, :wq (escrevi e poder salvar).*

- Agora os últimos comandos:

```
$ git branch -M main
```

```
$ git push -u origin main
```

The screenshot shows a GitHub repository page for 'VanessaVallarini / maratona-rocketseat'. The repository is on the 'main' branch, has 1 branch and 0 tags. The commit history shows a single initial commit by VanessaVallarini 25 minutes ago. The file list includes .github, public, src, .gitignore, README.md, database.sqlite, maratona.pdf, package-lock.json, and package.json. The README.md file is open, showing a 'JobsCalc' logo and links for 'Tecnologias', 'Projeto', 'Layout', and 'Licença'. The right sidebar contains sections for 'About', 'Releases', 'Packages', and 'Languages'. The 'Languages' section shows a bar chart with EJS at 18.5%, JavaScript at 36.1%, and CSS at 25.4%.

VanessaVallarini / maratona-rocketseat

main 1 branch 0 tags

Go to file Add file Code

VanessaVallarini initial commit 5b7cbab 25 minutes ago 1 commit

.github	initial commit	25 minutes ago
public	initial commit	25 minutes ago
src	initial commit	25 minutes ago
.gitignore	initial commit	25 minutes ago
README.md	initial commit	25 minutes ago
database.sqlite	initial commit	25 minutes ago
maratona.pdf	initial commit	25 minutes ago
package-lock.json	initial commit	25 minutes ago
package.json	initial commit	25 minutes ago

README.md

**JobsCalc**

Tecnologias | Projeto | Layout | Licença

PRs welcome license MIT

**About**

No description, website, or topics provided.

Readme

**Releases**

No releases published  
[Create a new release](#)

**Packages**

No packages published  
[Publish your first package](#)

**Languages**

EJS	18.5%	JavaScript	36.1%
CSS	25.4%		