Configurando Ambiente de desenvolvimento Back-End TypeScripe com Node.js e VSCODE

Ferramentas utilizadas:

- **NODE.JS:** permite que você execute código JavaScript fora do navegador, ou seja,no servidor, no terminal, ou em scripts de automação.

- **TYPESCRIPT:** é um superset do JavaScript, ou seja, uma linguagem que estende o JavaScript com tipagem estática opcional e recursos avançados de linguagem, como interfaces, enums, generics, etc. Ele foi criado pela Microsoft e seu principal objetivo é trazer mais segurança, escalabilidade e produtividade para projetos JavaScript,especialmente em aplicações grandes ou com múltiplos desenvolvedores.

- **VSCODE:** O VS Code (Visual Studio Code) é um editor de código-fonte leve, gratuito e multiplataforma, criado pela Microsoft, amplamente utilizado por desenvolvedores para programar em várias linguagens, como JavaScript, TypeScript, Python, Java, C++, HTML,CSS, entre outras.

Instalar Node.Js

- Acesse: https://nodejs.org/en/download
- Baixe e instale a versão recomendada para seu sistema
- Exemplo de versão utilizada: v22.15.0

Após a instalação do Node.js escreva os seguintes comandos no seu terminal:

Caso não saiba abrir o terminal utilize:

- Win + R -> abre o Executar do Windows
- Digite: cmd e dê um Enter
- Pronto o seu terminal foi aberto!

node -v: para descobrir a versão do seu node.js
npm -v: para descobrir a versão do npm do node.js

Diferença entre NPM e NPX

NPM (Node Package Manager)

- **Função principal:** instalar **pacotes** (dependências) no projeto ou no sistema.
- É usado para **gerenciar dependências** e scripts no package.json.

Exemplos:

npm install typescript --save-dev # instala o TypeScript no projeto (como dependência de desenvolvimento)

npm install express # instala o Express como dependência

O npm **não executa diretamente** binários dos pacotes. Ele apenas os instala em node_modules/.bin.

NPX (Node Package Execute)

- **Função principal: executar pacotes** que estão instalados localmente (em node_modules) ou **executar sem instalar** (baixando temporariamente da internet).
- Ideal para **executar CLIs**, ferramentas ou **comandos de pacotes** sem ter que instalá-los globalmente.

Exemplos:

npx tsc --init # executa o compilador TypeScript sem precisar chamar diretamente o caminho do binário

npx create-react-app my-app # executa o gerador de projeto React sem instalar globalmente

Criar nossa Pasta de Projeto

Navegar até o diretório disco local C

Criar uma pasta - coloque o nome que desejar nesta pasta, a minha será: TypeScript
Dentro desta pasta deixo opcional criar uma subpasta chamada de: curso

Iniciando projeto com Node.Js:

- Dentro da Pasta onde você quer criar o projeto com node
- no meu caso : C://TypeScript/curso

use o comando:

npm init -y # isso cria o package.json, o arquivo que gerencia as dependências do projeto.

Instalar o TypeScript

Para instalar o TypeScript acesse esse site: https://www.typescriptlang.org/download/
e siga as instruções.

Dentro da Pasta do seu Projeto onde vamos usar o Node e o TypeScript use o comando:

(recomendado para projetos)
npm install typescript --save-dev -> Instale o TypeScript localmente

Obs.: --save-dev instala só para desenvolvimento.

Se quiser usar o TypeScript em qualquer projeto no seu sistema, instale globalmente:

npm install -g typescript

Ao fazer essa instalação uma pasta será criada que contém os módulos do projeto node com TypeScript, nosso famoso (node_modules)

Agora crie o arquivo de configuração do typescript, use:

npx tsc --init # Isso cria o tsconfig.json, com várias opções de compilação.

lembrando: usa-se npx para instalar localmente no seu projeto

Configurando o tsconfig.json, arquivo de configuração do TypeScript

Agora vamos configurar o arquivo de configuração do typescript:

Acesse esse site: https://github.com/microsoft/TypeScript/wiki/Node-Target-Mapping para ter configurações básicas do tsconfig.json

deixei configurado o "outDir": "./build" -> Basicamente é a pasta onde JavaScript gerado pelo nosso TypeScript compilado vai estar

Vamos criar uma Pasta chamada de src, ao qual irá comportar nossos arquivos.ts, que serão executados pelo Node.Js após conversão para javascript. Como o node.js não suporta o typescript, o typescript deve ser compilado e convertido em um código JavaScript.

Use o comando no terminal:

npm install typescript @types/node -D

npm install tsx -D -> Basicamente converte o código Typescript para JavaScript executa o código JavaScript convertido com o Node.js de forma automatizada

depois, posteriormente após a configuração do package.json, de um npm dev run

Configurando o package.json

Depois de rodar:

npm init -y

Você terá algo assim:

```
{
  "name": "meu-projeto",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT"
}
```

Vamos transformar isso em um setup completo.

Instale as dependências

Execute no terminal (no diretório do projeto):

npm install typescript tsx --save-dev

Agora o package.json vai ganhar uma seção chamada devDependencies.

Adicione os scripts: Agora edite seu package.json assim:

```
{
  "name": "meu-projeto",
  "version": "1.0.0",
  "description": "Projeto Node.js com TypeScript",
  "main": "build/index.js",
  "scripts": {
    "dev": "tsx watch src/index.ts",
    "build": "tsc",
    "start": "node build/index.js"
  },
  "author": "Seu Nome",
  "license": "MIT",
  "devDependencies": {
    "tsx": "^4.12.0",
    "typescript": "^5.3.3"
  }
}
```

O que cada script faz:

Script	Comando	O que faz
npm run dev	tsx watch src/index.ts	Executa o arquivo src/index.ts e recompila automaticamente em tempo real
npm run build	tsc	Compila todos os arquivos .ts em .js para a pasta build/
npm start	node build/index.js	Roda o código já compilado com o Node.js

Exemplo de estrutura do projeto

```
/meu-projeto
  /src
    index.ts
  /build
  tsconfig.json
  package.json
  node_modules/
```

Como usar

1. Escreva seu código em src/index.ts

Para rodar em tempo real: npm run dev

Para compilar: npm run build

Para rodar o build: npm start