Guia: Rodando no VS Code, Versionando com GitHub e Publicando na Vercel

Este guia detalha os passos para pegar o código React do nosso aplicativo de controle de entregas (que usa IndexedDB) e:

1. Configurá-lo como um projeto React local para rodar no VS Code.
2. Versionar o código usando Git e publicá-lo no GitHub.
3. Fazer o deploy (publicação online) do aplicativo usando a plataforma Vercel.

Parte 1: Configurando e Rodando o Projeto Localmente no VS Code

Para rodar um aplicativo React, você precisará do Node.js e de um gerenciador de pacotes (npm ou yarn) instalados no seu computador. Recomendo usar o **Vite** para criar e gerenciar seu projeto React, pois é uma ferramenta moderna e muito rápida.

1. Instale o Node.js e npm/yarn (se ainda não tiver):

- Baixe o Node.js em nodejs.org (recomenda-se a versão LTS). O npm é instalado junto com o Node.js.
- Se preferir o yarn, instale-o após o Node.js: `npm install --global yarn`.

2. Crie um Novo Projeto React com Vite:

Abra seu terminal (ou o terminal integrado do VS Code) e execute um dos seguintes comandos:

- **Usando npm:**

```
npm create vite@latest seu-delivery-app -- --template react
```

-
- **Usando yarn:**

```
yarn create vite seu-delivery-app --template react
```

-

Substitua `seu-delivery-app` pelo nome que desejar para a pasta do projeto. Siga as instruções que aparecerão no terminal.

Após a criação, navegue para a pasta do projeto:

```
cd seu-delivery-app
```

3. Instale as Dependências Necessárias:

O código do nosso aplicativo utiliza as bibliotecas lucide-react para ícones e recharts para gráficos. Instale-as:

- **Usando npm:**

```
npm install lucide-react recharts
```

- **Usando yarn:**

```
yarn add lucide-react recharts
```

4. Adicione o Código do Aplicativo ao Projeto:

- No seu projeto Vite recém-criado, vá para a pasta `src/`.
- **Substitua o conteúdo do arquivo** `src/App.jsx` (ou `src/App.js`) pelo código completo do artefato "Sistema de Controle de Entregas (IndexedDB Refatorado)" que geramos.
- Se houver um arquivo `src/App.css`, você pode limpá-lo ou remover suas importações do `App.jsx`, pois estamos usando Tailwind CSS.

5. Configure o Tailwind CSS:

O código do aplicativo usa classes do Tailwind CSS para estilização. Siga estes passos para configurá-lo no seu projeto Vite:

- Instale o Tailwind CSS e suas dependências de peer:
  Aqui usamos a versão específica que funcionou para você, para maior confiabilidade:

```
# Usando npm
npm install -D tailwindcss@3.4.17 postcss autoprefixer
# Usando yarn
# yarn add -D tailwindcss@3.4.17 postcss autoprefixer
```

- **Crie os arquivos de configuração do Tailwind CSS e PostCSS:**

```
npx tailwindcss init -p
```

- Se este comando `npx` ainda apresentar problemas, você pode tentar as alternativas discutidas anteriormente (como executar o script diretamente de `node_modules/.bin` ou via um script no `package.json`).

- Configure o tailwind.config.js:
  Abra o arquivo tailwind.config.js e modifique a propriedade content para que o Tailwind saiba onde procurar por suas classes:

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}", // Garante que ele olhe seus arquivos React
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

-
- Adicione as diretivas do Tailwind ao seu arquivo CSS principal:
  Abra o arquivo src/index.css (o Vite geralmente cria este arquivo). Substitua todo o conteúdo dele por:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

-
- Importe o arquivo CSS principal no seu main.jsx:
  Verifique se o arquivo src/index.css está sendo importado no seu arquivo src/main.jsx (ou src/main.js):

```
// src/main.jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css' // <--- Certifique-se que esta linha existe

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

-

6. Rode o Projeto Localmente:

Com tudo configurado, inicie o servidor de desenvolvimento:

- **Usando npm:**

```
npm run dev
```

- **Usando yarn:**

```
yarn dev
```

O terminal mostrará um endereço local (geralmente `http://localhost:5173` ou similar). Abra-o no seu navegador para ver e testar o aplicativo. Os dados serão armazenados no IndexedDB do seu navegador.

Parte 2: Subindo o Projeto para o GitHub

Versionar seu código com Git e hospedá-lo no GitHub é uma prática essencial.

1. Crie um Repositório no GitHub:

- Vá para GitHub.com.
- Crie um novo repositório. Dê um nome a ele (ex: `controle-entregas-app`).
- **Não** o inicialize com `README`, `.gitignore` ou licença neste momento, pois faremos isso localmente.
- Copie a URL do seu novo repositório (ex: `https://github.com/seu-usuario/controle-entregas-app.git`).

2. Inicialize o Git Localmente e Faça o Primeiro Push:

No terminal, dentro da pasta do seu projeto (seu-delivery-app):

- **Inicialize um repositório Git:**

```
git init -b main
```

- (O `-b main` define a branch principal como `main`, que é o padrão atual.)
- Crie um arquivo .gitignore:
  Na raiz do seu projeto, crie um arquivo chamado .gitignore. Este arquivo diz ao Git quais arquivos e pastas ignorar. Um bom .gitignore para projetos Vite/React é:

```
# Dependencies
/node_modules

# Vite build output
```

```
/dist
/dist-ssr

# Vite cache
.vite/

# Logs
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*

# Editor directories and files
.idea
.vscode/settings.json
# Mantenha .vscode/extensions.json se quiser sugerir extensões
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?

# Environment variables
.env
.env*.local
.env.*.local

# Mac files
.DS_Store
```

-
- **Adicione os arquivos ao Git:**

```
git add .
```

-
- **Faça o primeiro commit:**

```
git commit -m "Primeiro commit: Configuração inicial do app de controle de entregas"
```

-
- Conecte seu repositório local ao repositório do GitHub:
  Substitua <URL_DO_SEU_REPOSITORIO_GITHUB> pela URL que você copiou anteriormente:

```
git remote add origin <URL_DO_SEU_REPOSITORIO_GITHUB>
```

-
- **Envie (push) seus arquivos para o GitHub:**

`git push -u origin main`

-

Agora seu código está versionado e seguro no GitHub!

Parte 3: Publicando pela Vercel

A Vercel é uma plataforma excelente para fazer deploy de aplicações frontend modernas como projetos Vite/React. Ela se integra perfeitamente com o GitHub.

1. Crie uma Conta na Vercel:

- Se ainda não tiver, acesse vercel.com e crie uma conta. A maneira mais fácil é se inscrever usando sua conta do GitHub.

2. Importe seu Projeto do GitHub para a Vercel:

- No seu dashboard da Vercel, clique em "Add New..." e selecione "Project".
- A Vercel provavelmente pedirá para se conectar à sua conta do GitHub (se ainda não estiver). Autorize o acesso.
- Você verá uma lista dos seus repositórios do GitHub. Encontre o repositório do seu `controle-entregas-app` e clique em "Import".

3. Configure o Projeto na Vercel:

- **Project Name:** A Vercel sugerirá um nome, mas você pode alterá-lo.
- **Framework Preset:** A Vercel é muito boa em detectar automaticamente projetos Vite. Se ela identificar corretamente, ótimo. Caso contrário, selecione "Vite" na lista.
- **Root Directory:** Geralmente é `./` (a raiz do seu repositório). Mantenha assim, a menos que seu código esteja em uma subpasta específica.
- **Build and Output Settings:**
  - **Build Command:** A Vercel deve preencher isso automaticamente com `vite build` ou `npm run build` (que, para projetos Vite, geralmente executa `vite build`). Verifique o script `build` no seu arquivo `package.json`.
  - **Output Directory:** Para projetos Vite, o diretório de saída da build é `dist`. A Vercel também costuma detectar isso.
  - **Install Command:** Pode deixar o padrão (`npm install` ou `yarn install`, dependendo do seu projeto).
- **Environment Variables:** Para este projeto, como estamos usando IndexedDB (armazenamento local no navegador), você **não precisa** configurar nenhuma variável de ambiente específica na Vercel para o funcionamento básico do armazenamento.

4. Faça o Deploy:

- Clique no botão "Deploy".

- A Vercel irá buscar seu código do GitHub, instalar as dependências, executar o comando de build e, finalmente, publicar sua aplicação.
- O processo pode levar alguns minutos. Você verá os logs do build em tempo real.
- Quando terminar, a Vercel fornecerá um ou mais links públicos (ex: `seu-delivery-app.vercel.app`) onde sua aplicação estará acessível online.

Considerações Importantes para a Versão Publicada:

- **Armazenamento IndexedDB é Local:** Lembre-se que o IndexedDB armazena dados **localmente no navegador de cada usuário**. Isso significa que:
    - Cada pessoa que acessar seu site publicado na Vercel terá seu próprio banco de dados IndexedDB vazio inicialmente.
    - Os dados não são compartilhados entre diferentes usuários ou entre diferentes navegadores/dispositivos do mesmo usuário.
    - Se um usuário limpar os dados do navegador, os dados do aplicativo serão perdidos para ele.
- **Atualizações Contínuas (CI/CD):** Por padrão, a Vercel configura a integração contínua. Sempre que você fizer um `git push` para a branch principal (`main`) do seu repositório no GitHub, a Vercel automaticamente fará um novo build e deploy da versão mais recente do seu aplicativo.
- **Domínio Personalizado:** A Vercel permite que você configure um domínio personalizado para sua aplicação, se desejar.

Seguindo esses passos, você terá seu aplicativo de controle de entregas rodando localmente para desenvolvimento, com o código versionado no GitHub, e publicado online para que qualquer pessoa possa acessá-lo (com seus próprios dados locais).

Gui