Resolvendo o erro "Unable to locate file in Vite manifest" no Laravel

Este guia explica como resolver o erro `Unable to locate file in Vite manifest: resources/js/osAuxiliar_Veiculo.js` que ocorre frequentemente em ambientes de homologação e produção.

1. Entendendo a diferença entre Vite e asset()

Vite (@vite)

O que é? Vite é um bundler e servidor de desenvolvimento moderno usado pelo Laravel a partir da versão 9.

Como funciona:

- Em desenvolvimento: Funciona como um servidor para hot module replacement (HMR)
- Em produção: Compila e empacota os arquivos, gerando um manifesto (manifest.json)
- A diretiva `@vite` procura os arquivos compilados no manifesto

Exemplo de uso:

```blade
@vite(['resources/css/app.css', 'resources/js/osAuxiliar_Veiculo.js'])
```

Helper asset()

O que é? `asset()` é um helper do Laravel que gera URLs para assets públicos.

Como funciona:

- Simplesmente aponta para a pasta `public`
- Não realiza nenhum processamento ou bundling
- Funciona com arquivos estáticos que já estão na pasta `public`

Exemplo de uso:

```blade
<script src="{{ asset('js/osAuxiliar_Veiculo.js') }}"></script>
```

2. Por que ocorre o erro no ambiente de homologação/produção?

O erro `Unable to locate file in Vite manifest` ocorre por dois motivos principais:

2.1. Configuração incorreta do vite.config.js

O arquivo não está incluído no array `input` do arquivo `vite.config.js`. Cada arquivo que você referencia com `@vite` nas views deve estar explicitamente listado:

```javascript
// vite.config.js
export default defineConfig({
    plugins: [
        laravel({
            input: [
                'resources/css/app.css',
                'resources/js/app.js',
                'resources/js/osAuxiliar_Veiculo.js', // Este arquivo precisa estar listado aqu
            ],
            refresh: true,
        }),
    ],
});
```

2.2. Script não importado no app.js

Alternativamente, se você não importar o script em seu arquivo principal (`app.js`), ele não será incluído no processo de compilação:

```javascript
// resources/js/app.js
import './bootstrap';

// Importe seus scripts personalizados
import './osAuxiliar_Veiculo'; // Importar aqui inclui este script no bundle principal
```

Este problema ocorre porque:

- Em desenvolvimento: O Vite serve os arquivos diretamente, mesmo sem estarem no manifesto
- Em produção: O Laravel procura estritamente no manifesto gerado pelo `npm run build`

3. Soluções disponíveis

Você tem três opções para resolver o problema:

Opção 1: Importar os scripts no app.js (Recomendado)

1. Adicione importações para todos seus scripts personalizados no arquivo principal:

```javascript
// resources/js/app.js
import './bootstrap';

// Importe seus scripts personalizados
import './osAuxiliar_Veiculo';
import './fornecedores';
import './dashboard';
```

2. Nas suas views, use apenas:

```blade
@vite(['resources/css/app.css', 'resources/js/app.js'])
```

Vantagens:

- Manutenção simplificada (você só precisa gerenciar um ponto de entrada)
- Menos risco de erros de manifesto
- Bundle mais eficiente e otimizado

Desvantagens:

- Todas as páginas carregam todos os scripts (carregamento "tudo ou nada")
- Potencial aumento do tamanho do bundle final

Opção 2: Configurar corretamente o vite.config.js

1. Adicione todos os arquivos JS/CSS no array `input`:

```javascript
export default defineConfig({
    plugins: [
        laravel({
            input: [
                'resources/css/app.css',
                'resources/js/app.js',
                'resources/js/osAuxiliar_Veiculo.js',
                'resources/js/fornecedores.js',
                // Adicione todos os scripts que você usa com @vite
            ],
            refresh: true,
        }),
    ],
});
```

2. Execute `npm run build` durante cada deploy

Vantagens:

- Carregamento seletivo (cada página carrega apenas os scripts necessários)
- Maior controle sobre o que cada página carrega

Desvantagens:

- Manutenção mais trabalhosa (precisa atualizar o vite.config.js a cada novo script)
- Maior risco de esquecer de incluir arquivos

Opção 3: Substituir por chamadas asset()

1. Copie seus arquivos JS para a pasta public:

```bash
mkdir -p public/js
cp resources/js/osAuxiliar_Veiculo.js public/js/
```

2. Substitua as chamadas @vite nas views:

```blade
<!-- Antes -->
@vite(['resources/js/osAuxiliar_Veiculo.js'])

<!-- Depois -->
<script src="{{ asset('js/osAuxiliar_Veiculo.js') }}"></script>
```

Vantagens:

- Solução rápida para ambientes problemáticos
- Funciona de forma consistente em todos os ambientes

Desvantagens:

- Perde todos os benefícios do Vite (minificação, bundling, etc.)
- Exige gerenciamento manual de dependências
- Exige cópias manuais de arquivos

Conclusão

A solução mais recomendada é a Opção 1 (importar no app.js), pois oferece o melhor equilíbrio entre simplicidade e funcionalidades do Vite. Para scripts realmente independentes ou situações emergenciais, as outras opções podem ser consideradas.

Lembre-se de executar `npm run build` após qualquer alteração nos arquivos JS/CSS antes de fazer deploy para produção ou homologação.