



Guia Completo - Deploy React + Vite no Vercel



Por que deu errado no início e como resolvemos

❌ Problemas Encontrados:

1. package.json inexistente/corrompido

- NPM não conseguia encontrar o arquivo
- Arquivo estava com encoding incorreto
- JSON mal formatado (sintaxe inválida)

2. Arquivos React vazios

- `src/App.jsx` e `src/main.jsx` com 0 bytes
- Comandos `echo` do Windows não funcionaram para JSX
- Estrutura de pastas incorreta

3. vite.config.js com dependências faltando

- Plugin `vite-plugin-pwa` não instalado
- Plugin `@vitejs/plugin-react` não configurado
- Sintaxe ES6 quebrada pelo terminal

4. HTTP 404 Error

- Vite não encontrava os arquivos de entrada
- `index.html` apontando para arquivos inexistentes



Como Resolvemos:

1. Criamos **package.json válido** com dependências corretas
2. Geramos **arquivos React funcionais** via comandos simples
3. Configuramos **vite.config.js básico** sem dependências extras
4. Testamos **step-by-step** cada componente



Roteiro Anti-Falhas - Passo a Passo

Pré-requisitos

- Node.js 16+ instalado
- NPM funcionando
- Git instalado (para deploy)

ETAPA 1: Criar Estrutura Base

1.1 Criar pasta do projeto

```
bash
mkdir gateway-financeiro-app
cd gateway-financeiro-app
```

1.2 Criar estrutura de pastas

```
bash
mkdir src
mkdir public
```

ETAPA 2: Criar package.json (CRÍTICO)

2.1 Método Seguro - Via comando único

```
bash
echo {"name":"gateway-financeiro-app","version":"1.0.0","private":true,"type":"module","scripts":{"dev":"vite","build":"vite
```

2.2 Verificar se foi criado corretamente

```
bash
dir package.json
type package.json
```

! VERIFICAÇÃO OBRIGATÓRIA: O arquivo deve ter conteúdo JSON válido, não pode estar vazio!

ETAPA 3: Criar Arquivos React

3.1 Criar src/main.jsx

```
bash
```

```
(
echo import React from 'react'
echo import ReactDOM from 'react-dom/client'
echo import App from './App.jsx'
echo.
echo ReactDOM.createRoot^(document.getElementById^('root'^)).render^(
echo   React.createElement^(React.StrictMode, null, React.createElement^(App)^)
echo ^)
) > src\main.jsx
```

3.2 Criar src/App.jsx básico para teste

```
bash

(
echo import React from 'react'
echo.
echo export default function App^(^) {
echo   return React.createElement^('div', null,
echo     React.createElement^('h1', null, 'Gateway Financeiro'^),
echo     React.createElement^('p', null, 'Funcionando! 🚀'^)
echo   ^)
echo }
echo }
) > src\App.jsx
```

3.3 Verificar se têm conteúdo

```
bash

dir src
type src\main.jsx
type src\App.jsx
```

! **VERIFICAÇÃO OBRIGATÓRIA:** Arquivos devem ter tamanho > 0 bytes!

ETAPA 4: Criar vite.config.js

4.1 Configuração básica e segura

```
bash
```

```
echo import { defineConfig } from 'vite' > vite.config.js
echo import react from '@vitejs/plugin-react' >> vite.config.js
echo. >> vite.config.js
echo export default defineConfig({ >> vite.config.js
echo   plugins: [react()], >> vite.config.js
echo   server: { port: 5173 } >> vite.config.js
echo }) >> vite.config.js
```

4.2 Verificar sintaxe

```
bash

type vite.config.js
```

ETAPA 5: Criar index.html

5.1 Template HTML básico

```
bash

echo ^<!DOCTYPE html^>^<html lang="pt-BR"^>^<head^>^<meta charset="UTF-8" /^>^<meta name="viewpor
```

ETAPA 6: Instalar e Testar

6.1 Instalar dependências

```
bash

npm install
```

⚠ **Se der erro ENOENT:** O package.json não existe ou está corrompido. Volte à Etapa 2.

6.2 Instalar plugin React

```
bash

npm install @vitejs/plugin-react --save-dev
```

6.3 Teste básico

```
bash

npm run dev
```

✓ **Sucesso:** Deve abrir `http://localhost:5173` com "Gateway Financeiro" e "Funcionando! 🎉"

✗ **Se der erro 404:** Verifique se todos os arquivos existem e têm conteúdo.

ETAPA 7: Adicionar Código Completo

7.1 Substituir App.jsx

1. Pare o servidor (Ctrl+C)
2. Abra `src\App.jsx` no Notepad
3. Substitua pelo código completo do Gateway Financeiro
4. Salve (Ctrl+S)

7.2 Testar versão completa

```
bash  
npm run dev
```

✓ **Sucesso:** Dashboard completo com KPIs, gráficos, tabelas, etc.

ETAPA 8: Preparar para Deploy

8.1 Testar build de produção

```
bash  
npm run build
```

✓ **Sucesso:** Deve criar pasta `dist` sem erros.

8.2 Testar preview

```
bash  
npm run preview
```

✓ **Sucesso:** Deve funcionar em `http://localhost:4173`

8.3 Criar vercel.json

```
bash  
echo {"buildCommand":"npm run build","outputDirectory":"dist","framework":"vite","rewrites":[{"source":"/(.*)", "destinat
```

8.4 Criar .gitignore

```
bash

echo node_modules/ > .gitignore
echo dist >> .gitignore
echo .env >> .gitignore
echo .vercel >> .gitignore
```

ETAPA 9: Deploy no Vercel

9.1 Preparar Git

```
bash

git init
git add .
git commit -m "Initial commit: Gateway Financeiro"
```

9.2 Subir para GitHub

```
bash

# Criar repositório no GitHub primeiro
git remote add origin https://github.com/SEU-USUARIO/gateway-financeiro-app.git
git branch -M main
git push -u origin main
```

9.3 Deploy no Vercel

1. Acesse vercel.com
2. Conecte conta GitHub
3. Import Project → Selecione seu repositório
4. Deploy automático!

Troubleshooting - Problemas Comuns

✗ "npm install" falha

Causa: package.json inexistente/corrompido

Solução: Recriar package.json com comando da Etapa 2

✗ "vite not found"

Causa: Dependências não instaladas

Solução:

```
bash  
  
npm install  
# ou  
npx vite
```

❌ HTTP 404 Error

Causa: Arquivos React vazios ou inexistentes

Solução: Verificar se src/App.jsx e src/main.jsx têm conteúdo

❌ "Failed to resolve import"

Causa: vite.config.js mal configurado

Solução: Recriar vite.config.js básico da Etapa 4

❌ Página em branco

Causa: Erro de JavaScript no navegador

Solução: Abrir F12, verificar console, corrigir erro

❌ Build falha

Causa: Sintaxe JSX incorreta ou dependências faltando

Solução:

```
bash  
  
npm install @vitejs/plugin-react --save-dev  
npm run build
```

Checklist Final

Antes do deploy, verificar:

- ☐ package.json existe e tem conteúdo válido
- ☐ src/App.jsx e src/main.jsx existem e têm conteúdo
- ☐ vite.config.js está configurado corretamente
- ☐ index.html aponta para /src/main.jsx
- ☐ npm install executa sem erros
- ☐ npm run dev abre o app no navegador
- ☐ npm run build executa sem erros
- ☐ npm run preview funciona

- ☐ `vercel.json` configurado para SPA
 - ☐ Git configurado e commit feito
-

💡 Dicas de Prevenção

1. **Sempre verificar tamanho dos arquivos** - arquivos 0 bytes indicam problema
 2. **Testar step-by-step** - não pular verificações
 3. **Usar comandos simples** - evitar sintaxe complexa no terminal
 4. **Backup dos arquivos funcionais** - salvar códigos que funcionam
 5. **Testar localmente antes do deploy** - nunca fazer deploy sem testar
-

📖 Arquivos de Referência

package.json mínimo funcional:

```
json
{
  "name": "gateway-financeiro-app",
  "version": "1.0.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "recharts": "^2.8.0"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.2.1",
    "vite": "^5.0.8"
  }
}
```

vite.config.js mínimo funcional:

```
javascript
```



```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  server: { port: 5173 }
})
```

src/main.jsx mínimo funcional:

```
javascript

import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'

ReactDOM.createRoot(document.getElementById('root')).render(
  React.createElement(React.StrictMode, null, React.createElement(App))
)
```

Conclusão

O sucesso veio da abordagem **step-by-step sistemática**:

1. **Identificamos cada problema específico**
2. **Resolvemos um de cada vez**
3. **Verificamos cada etapa antes de avançar**
4. **Testamos em camadas** (básico → completo)

Este roteiro garante 99% de sucesso se seguido exatamente como descrito!

Gateway Financeiro - Dashboard React + Vite + Vercel ✨