Dockerfile v8.12 - Fix DevDependencies

🎯 Problema Identificado

Error en v8.11

```
Failed to compile.

Error: Cannot find module 'tailwindcss'
Require stack:
- /app/node_modules/next/dist/build/webpack/config/blocks/css/plugins.js
```

Causa raíz: Tailwind CSS está en `devDependencies`, pero npm NO lo estaba instalando.

🔍 Análisis del Problema

Por Qué NO se Instalaban DevDependencies

Configuración errónea en v8.11:

```dockerfile
FROM node:18-alpine AS base

# ❌ PROBLEMA: NODE_ENV=production en stage base
ENV NODE_ENV=production
ENV NEXT_TELEMETRY_DISABLED=1
ENV PORT=3000
ENV HOSTNAME=0.0.0.0

WORKDIR /app

# Stage deps hereda NODE_ENV=production del stage base
FROM base AS deps

COPY app/package.json ./

# ❌ Con NODE_ENV=production, npm NO instala devDependencies
RUN npm install --legacy-peer-deps
```

Flujo del Problema

```
1. Stage 'base'
   ↓ ENV NODE_ENV=production

2. Stage 'deps' (hereda de 'base')
   ↓ NODE_ENV=production está activo

3. npm install
   ↓ Detecta NODE_ENV=production
   ↓ NO instala devDependencies

4. devDependencies faltantes:
   ❌ tailwindcss
   ❌ postcss
   ❌ autoprefixer
   ❌ @types/* (tipos de TypeScript)
   ❌ eslint

5. Build de Next.js falla
   ❌ Cannot find module 'tailwindcss'
```

✅ Solución en v8.12

Configuración Correcta

```dockerfile
FROM node:18-alpine AS base

# ✅ NO establecer NODE_ENV=production aquí
# Solo variables que NO afectan el install
ENV NEXT_TELEMETRY_DISABLED=1
ENV PORT=3000
ENV HOSTNAME=0.0.0.0

WORKDIR /app

# Stage deps (sin NODE_ENV=production)
FROM base AS deps

COPY app/package.json ./

# ✅ npm instala TODAS las dependencias
RUN echo "=== INSTALANDO DEPENDENCIAS ===" && \
    echo "NODE_ENV: ${NODE_ENV:-not set}" && \
    npm install --legacy-peer-deps && \
    echo "✅ Dependencias instaladas (incluyendo devDependencies)"
```

NODE_ENV en Stage Runner (Correcto)

```
# Stage runner (imagen final)
FROM base AS runner

WORKDIR /app

# ✅ AQUÍ sí establecemos NODE_ENV=production
ENV NODE_ENV=production
ENV NEXT_TELEMETRY_DISABLED=1
ENV PORT=3000

# ... resto del runtime
```

📊 Comparación v8.11 vs v8.12

Variables de Entorno por Stage

Stage	v8.11	v8.12
base	`NODE_ENV=production` ❌	`NODE_ENV=` (no set) ✅
deps	Hereda `production` ❌	No tiene NODE_ENV ✅
builder	Hereda `production` ❌	No tiene NODE_ENV ✅
runner	`NODE_ENV=production` ✅	`NODE_ENV=production` ✅

Resultado de npm install

Versión	devDependencies	Resultado
v8.11	❌ NO instaladas	Build falla
v8.12	✅ Instaladas	Build exitoso

🔧 Dependencias Afectadas

DevDependencies Críticas para Build

```
{
  "devDependencies": {
    "tailwindcss": "3.3.3",           // ← Error principal
    "postcss": "8.4.30",              // ← Requerido por Next.js
    "autoprefixer": "10.4.15",        // ← Requerido por Tailwind
    "@types/node": "20.6.2",          // ← TypeScript
    "@types/react": "18.2.22",        // ← TypeScript
    "@types/react-dom": "18.2.7",     // ← TypeScript
    "typescript": "5.2.2",            // ← Compilador
    "eslint": "9.24.0",               // ← Linting
    "eslint-config-next": "15.3.0"    // ← Next.js linting
  }
}
```

Todas estas dependencias son NECESARIAS para el build de Next.js.

💡 Por Qué Esto Sucede

Comportamiento de npm install

npm tiene diferentes modos según NODE_ENV:

NODE_ENV no establecido o = “development”

```
npm install
# Instala:
# ✅ dependencies
# ✅ devDependencies
# ✅ peerDependencies
```

NODE_ENV = “production”

```
npm install
# Instala:
# ✅ dependencies
# ❌ devDependencies (omitidas automáticamente)
# ✅ peerDependencies
```

Esto es equivalente a:

```
npm install --production  # Omite devDependencies
```

🎯 Por Qué la Solución Funciona

Estrategia de Multi-Stage Build Correcta

```
Stage 1: deps
├─ NODE_ENV: no establecido
├─ npm install → Instala TODO
└─ Resultado: node_modules COMPLETO (deps + devDeps)

Stage 2: builder
├─ Copia: node_modules COMPLETO
├─ npm run build → Usa Tailwind, PostCSS, TypeScript
└─ Resultado: .next/ build exitoso

Stage 3: runner (imagen final)
├─ NODE_ENV: production ✅
├─ Copia: Solo archivos necesarios para runtime
└─ Resultado: Imagen optimizada solo con runtime deps
```

Beneficios de la Estrategia

1. **Stage deps/builder:** Instala TODO lo necesario para build
2. **Stage runner:** Solo incluye lo necesario para runtime
3. **Imagen final:** Optimizada (~200MB) pero build completo

📋 Qué Hay de Nuevo en v8.12

Cambios Técnicos

```
# Stage base
FROM node:18-alpine AS base

-ENV NODE_ENV=production
+# NODE_ENV no establecido (permite instalar devDependencies)
 ENV NEXT_TELEMETRY_DISABLED=1
 ENV PORT=3000
 ENV HOSTNAME=0.0.0.0

# Stage deps
FROM base AS deps

 RUN echo "=== INSTALANDO DEPENDENCIAS ===" && \
+    echo "NODE_ENV: ${NODE_ENV:-not set}" && \
     npm install --legacy-peer-deps && \
+    echo "✅ Dependencias instaladas (incluyendo devDependencies)"
```

Stage runner (sin cambios)

```
# Aquí sí queremos NODE_ENV=production
ENV NODE_ENV=production
```

🚀 Resultado Esperado

Build Exitoso con v8.12

```
#25 [builder 15/16] RUN npx prisma generate
#25 ✔ Generated Prisma Client (v6.7.0)

#26 [builder 16/16] RUN npm run build
#26 Creating an optimized production build...
#26 ✓ Compiled successfully
#26 ✓ Linting and checking validity of types
#26 ✓ Collecting page data
#26 ✓ Generating static pages (0/0)
#26 ✓ Finalizing page optimization
#26
#26 Route (app)                                Size     First Load JS
#26 ┌ ○ /                                     1.2 kB          120 kB
#26 ├ ○ /admin/analytics                       2.3 kB          125 kB
#26 ├ ○ /admin/audit                           1.8 kB          122 kB
#26 └ ...
#26
#26 ✅ Build de Next.js completado exitosamente
```

📊 Checklist de Verificación

- [x] ✅ Problema identificado: `Cannot find module 'tailwindcss'`
- [x] ✅ Causa raíz: `NODE_ENV=production` en stage base
- [x] ✅ Solución: Remover `NODE_ENV=production` de stage base
- [x] ✅ Mantener `NODE_ENV=production` en stage runner
- [x] ✅ Verificar que componentes existen
- [x] ✅ Commit y push a GitHub
- [ ] → **Rebuild en EasyPanel** (tu turno)
- [ ] → **Verificar build exitoso**

🎉 Conclusión

El Problema Era Simple Pero Sutil

❌ **v8.11:**

```
NODE_ENV=production (global)
→ npm no instala devDependencies
→ Falta Tailwind
→ Build falla
```

✅ **v8.12:**

```
NODE_ENV no establecido (en build stages)
→ npm instala TODO (deps + devDeps)
→ Tailwind disponible
→ Build exitoso
```

Lección Aprendida

En Dockerfiles multi-stage:

- ✅ NO establecer `NODE_ENV=production` en stages de build
- ✅ Solo establecerlo en el stage final (runner)
- ✅ Así se instalan devDeps para build pero no van a producción

Próximo Paso

Rebuild con v8.12:

1. Ve a EasyPanel
2. Trigger rebuild
3. El build debería completar exitosamente ahora
4. La app estará lista para usar

Versión: 8.12
Fecha: 2025-10-08 03:35 GMT
Estado: 🔧 DEVDEPENDENCIES FIX

Fix crítico:

- NODE_ENV=production removido de stage base
- DevDependencies se instalan correctamente
- Build de Next.js puede completarse

Resultado esperado:

✅ Build exitoso
✅ App funcionando
✅ Todos los componentes compilados

¡Rebuild y confirma el build exitoso! 🚀