Dockerfile v8.13 - Fix Runtime Error

🎯 Nuevo Problema Identificado

Progreso Hasta Ahora

```
✅ v8.12: DevDependencies instaladas
✅ v8.12: Build de Docker completa exitosamente
✅ v8.12: Imagen Docker se crea
❌ v8.13: Runtime error al arrancar contenedor
```

Error de Runtime (v8.12)

```
▲ Next.js 14.2.28
  - Local:        http://localhost:3000

 ✓ Starting...
Error: Could not find a production build in the '.next' directory.
Try building your app with 'next build' before starting the production server.
https://nextjs.org/docs/messages/production-start-no-build-id
```

Esto significa:
- ✅ El build de Docker **pasó exitosamente**
- ✅ La imagen se **creó correctamente**
- ✅ El contenedor **arranca**
- ❌ El servidor de Next.js **no encuentra los archivos de build**

🔍 Análisis del Problema

¿Qué es el Standalone Output?

Cuando configuras `output: 'standalone'` en `next.config.js`, Next.js genera una versión autocontenida:

```
.next/
├── BUILD_ID
├── static/              # Assets estáticos compilados
├── standalone/          # Versión autocontenida
│   ├── .next/           # Build optimizado para runtime
│   ├── node_modules/    # Solo dependencias necesarias
│   ├── package.json
│   └── server.js        # Servidor standalone de Next.js
```

Configuración Actual

next.config.js:

```
const nextConfig = {
  output: 'standalone',
  experimental: {
    outputFileTracingRoot: path.join(__dirname, '../'),
  },
  // ...
};
```

Dockerfile v8.12 (runner stage):

```
# Copiar archivos del standalone build
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./.next/static
COPY --from=builder /app/public ./public
COPY --from=builder /app/prisma ./prisma

# Comando de inicio
CMD ["node", "server.js"]
```

Posibles Causas

1. **Standalone no se genera correctamente**
   - `next build` falla silenciosamente
   - Configuración de `next.config.js` incorrecta

2. **server.js no está donde esperamos**
   - La estructura del standalone es diferente
   - La copia no captura server.js

3. **Archivos .next/ no se copian correctamente**
   - El standalone esperaba .next/ en el runner
   - La copia solo incluye .next/static

4. **Prisma Client falta en runtime**
   - El servidor arranca pero falla al acceder DB
   - Error se manifiesta como “build not found”

✅ Solución en v8.13

Cambio 1: Verificar Standalone Output Después del Build

```
# Build de Next.js
RUN npm run build && \
    # ... verificaciones existentes ... && \
    echo "=== VERIFICANDO STANDALONE OUTPUT ===" && \
    echo "Contenido de .next/:" && ls -la .next/ && \
    echo "" && \
    echo "Contenido de .next/standalone/:" && ls -la .next/standalone/ && \
    echo "" && \
    if [ -f .next/standalone/server.js ]; then \
        echo "✅ server.js encontrado en standalone"; \
    else \
        echo "❌ server.js NO encontrado en standalone"; \
        exit 1; \
    fi
```

¿Qué hace?
- Lista TODO el contenido de `.next/`
- Lista TODO el contenido de `.next/standalone/`
- Verifica que `server.js` existe
- Falla el build si server.js no existe

¿Por qué?
Nos dirá EXACTAMENTE qué archivos se generan y dónde están.

Cambio 2: Copiar Prisma Client Explícitamente

```
# Copiar archivos del standalone build
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./.next/static
COPY --from=builder /app/public ./public

# Copiar prisma para migraciones si es necesario
COPY --from=builder /app/prisma ./prisma
COPY --from=builder /app/node_modules/.prisma ./node_modules/.prisma
COPY --from=builder /app/node_modules/@prisma/client ./node_modules/@prisma/client
```

¿Qué hace?
- Copia explícitamente `.prisma` (Prisma binaries)
- Copia explícitamente `@prisma/client` (Prisma Client)

¿Por qué?
El standalone output podría NO incluir estos archivos automáticamente.

Cambio 3: Verificar Estructura en Runner

```
# Verificar estructura de archivos (como root antes de cambiar usuario)
RUN echo "=== VERIFICANDO ESTRUCTURA RUNNER ===" && \
    echo "Archivos en /app:" && ls -la /app && \
    echo "" && \
    echo "¿Existe server.js?" && \
    if [ -f /app/server.js ]; then \
        echo "✅ server.js encontrado"; \
    else \
        echo "❌ server.js NO encontrado"; \
        echo "Listado completo:" && \
        find /app -name "server.js" || echo "No se encontró server.js"; \
        exit 1; \
    fi && \
    echo "" && \
    echo "Contenido de .next/:" && \
    ls -la /app/.next 2>/dev/null || echo "❌ .next/ no existe" && \
    echo "" && \
    echo "✅ Estructura verificada"
```

¿Qué hace?
- Lista archivos en `/app` del runner
- Busca `server.js` en toda la imagen si no está en `/app`
- Lista contenido de `.next/` si existe
- Falla el build si server.js no existe

¿Por qué?
Nos dirá EXACTAMENTE qué archivos llegaron al runner y dónde están.

📊 Diagnóstico Esperado

Escenario 1: Standalone NO se genera

Output del build:

```
=== VERIFICANDO STANDALONE OUTPUT ===
Contenido de .next/:
drwxr-xr-x  BUILD_ID
drwxr-xr-x  static/
❌ No existe .next/standalone/
```

Solución:
- Ajustar `next.config.js`
- Verificar que `output: 'standalone'` está correcto
- Verificar compatibilidad con App Router

Escenario 2: server.js en Lugar Incorrecto

Output del build:

```
=== VERIFICANDO STANDALONE OUTPUT ===
✅ .next/standalone/ existe
❌ server.js NO encontrado en standalone
```

Solución:
- Buscar dónde está server.js
- Ajustar el COPY para copiar desde la ubicación correcta

Escenario 3: Archivos No Se Copian Correctamente

Output del runner:

```
=== VERIFICANDO ESTRUCTURA RUNNER ===
Archivos en /app:
drwxr-xr-x  node_modules/
drwxr-xr-x  public/
drwxr-xr-x  prisma/
❌ server.js NO encontrado
```

Solución:
- La copia de `.next/standalone` no incluye server.js
- Ajustar el comando COPY

Escenario 4: Todo OK pero .next/ Falta

Output del runner:

```
=== VERIFICANDO ESTRUCTURA RUNNER ===
✅ server.js encontrado
❌ .next/ no existe
```

Solución:
- El standalone copia server.js pero no .next/
- Necesitamos copiar .next/ completo o ajustar estructura

🎯 Estrategia de Debugging

Paso 1: Rebuild con v8.13

```
# En EasyPanel
Trigger Rebuild
```

Paso 2: Observar Output del Build

Buscar en logs:

```
=== VERIFICANDO STANDALONE OUTPUT ===
```

Copiar TODO el output desde aquí hasta:

```
✅ server.js encontrado en standalone
```

O:

```
❌ server.js NO encontrado
```

Paso 3: Observar Output del Runner

Buscar en logs:

```
=== VERIFICANDO ESTRUCTURA RUNNER ===
```

Copiar TODO el output hasta:

```
✅ Estructura verificada
```

Paso 4: Determinar el Problema

Según el output, sabremos exactamente:
1. ¿Se genera el standalone?
2. ¿Dónde está server.js?
3. ¿Qué archivos hay en .next/?
4. ¿Qué archivos llegan al runner?

Paso 5: Aplicar el Fix Específico

Una vez identificado el problema exacto, aplicaremos el fix correspondiente.

📋 Posibles Fixes (según diagnóstico)

Fix A: Ajustar next.config.js

Si el standalone no se genera:

```
const nextConfig = {
  output: 'standalone',
  distDir: '.next',
  // Remover experimental.outputFileTracingRoot si causa problemas
};
```

Fix B: Ajustar Copia de Archivos

Si server.js está en un lugar diferente:

```
# Copiar desde la ubicación real
COPY --from=builder /app/.next/standalone/ ./
# O tal vez:
COPY --from=builder /app/.next/standalone/app/ ./
```

Fix C: Copiar .next/ Completo

Si .next/ falta en runner:

```
COPY --from=builder /app/.next ./.next
# En lugar de solo .next/static
```

Fix D: Usar Estructura Diferente

Si la estructura standalone es diferente:

```
# Copiar todo el standalone a un subdirectorio
COPY --from=builder /app/.next/standalone ./app
CMD ["node", "app/server.js"]
```

🚀 Resultado Esperado

Con v8.13, Sabremos:

```
1. ✅ ¿Se genera .next/standalone/?
2. ✅ ¿Está server.js en .next/standalone/?
3. ✅ ¿Qué archivos contiene .next/standalone/?
4. ✅ ¿Qué archivos llegan a /app en runner?
5. ✅ ¿Está server.js accesible en /app?
6. ✅ ¿Existe .next/ en runner?
```

Entonces Podremos:

```
→ Identificar el problema exacto
→ Aplicar el fix específico
→ Crear Dockerfile v8.14 funcional
→ Desplegar la app exitosamente
```

📝 Próximos Pasos

1. Rebuild con v8.13

```
# En EasyPanel
1. Ir al proyecto
2. Click "Rebuild"
3. Esperar a que termine
```

2. Copiar Output Completo

Secciones críticas:

```
=== VERIFICANDO STANDALONE OUTPUT ===
... copiar todo ...

=== VERIFICANDO ESTRUCTURA RUNNER ===
... copiar todo ...
```

3. Enviar Output

Copia y pega el output completo de estas dos secciones.

4. Diagnosticar

Con el output, identificaremos el problema exacto.

5. Aplicar Fix

Crearemos v8.14 con el fix específico.

🎉 Conclusión

Evolución del Debugging

```
v8.0-8.11  → Prisma y configuración base
v8.12      → DevDependencies instaladas ✅
v8.13      → Diagnóstico de runtime ← Estamos aquí
v8.14      → Fix de runtime (próximo)
```

Estrategia

En lugar de adivinar qué está mal, **agregamos verificaciones exhaustivas** para que el build nos diga EXACTAMENTE:

- ✅ Qué archivos se generan
- ✅ Dónde están
- ✅ Qué se copia al runner
- ✅ Qué falta

Una vez que sepamos qué está mal, el fix será directo y certero.

Versión: 8.13
Fecha: 2025-10-08 04:00 GMT
Estado: 🔍 DIAGNÓSTICO DE RUNTIME

Objetivo:
Identificar exactamente por qué el server.js
no encuentra el build de producción.

Resultado esperado:
Output detallado que nos permita crear
el fix específico en v8.14.

¡Rebuild con v8.13 y comparte el output de las verificaciones! 🔍