

# Dockerfile v8.8 - Captura Mejorada de Errores

## Problema Actual

El build de Next.js está fallando pero **no vemos los logs del error**:

```
ERROR: process did not complete successfully: exit code: 1
```

Sin información sobre **QUÉ** está fallando.

## Solución en v8.8

### Captura Completa de Errores

```
RUN echo "=== BUILD NEXT.JS ===" && \
  npm run build 2>&1 | tee /tmp/build-output.log; \
  BUILD_EXIT_CODE=${PIPESTATUS[0]}; \
  echo "Build exit code: $BUILD_EXIT_CODE"; \
  if [ $BUILD_EXIT_CODE -ne 0 ]; then \
    echo "❌ BUILD FALLÓ - Exit code: $BUILD_EXIT_CODE"; \
    echo "=== ÚLTIMAS 50 LÍNEAS DEL LOG ==="; \
    tail -50 /tmp/build-output.log; \
    echo "=== ARCHIVOS EN /app ==="; \
    ls -la; \
    exit 1; \
  fi
```

### Qué Hace Diferente

#### v8.7 (problema):



```
npm run build && ... || (error block)
```

- El OR (||) puede no capturar todos los errores
- No guarda el output del build
- Difícil ver qué falló

#### v8.8 (solución):

```
npm run build 2>&1 | tee /tmp/build-output.log;
BUILD_EXIT_CODE=${PIPESTATUS[0]};
if [ $BUILD_EXIT_CODE -ne 0 ]; then
  tail -50 /tmp/build-output.log;
fi
```

- ☒ Captura STDOUT y STDERR
- ☒ Guarda en archivo
- ☒ Captura exit code real

-  Muestra últimas 50 líneas si falla
-  Lista archivos para debug

---

## Logs Esperados

---

### Si hay Error de TypeScript

```

=== BUILD NEXT.JS ===
Creating an optimized production build...

Build exit code: 1
✗ BUILD FALLÓ - Exit code: 1
=== ÚLTIMAS 50 LÍNEAS DEL LOG ===

Type error: Cannot find module '@prisma/client'

   > 1 | import { PrismaClient } from '@prisma/client'
       |                                ^
   2 |

at app/lib/db.ts:1:1

=== ARCHIVOS EN /app ===
-rw-r--r-- package.json
drwxr-xr-x node_modules/
drwxr-xr-x prisma/

```

### Si hay Error de Webpack

```

=== ÚLTIMAS 50 LÍNEAS DEL LOG ===

webpack compiled with 1 error

./app/lib/db.ts
Module not found: Can't resolve '@prisma/client'
in /app/app/lib

```

### Si hay Error de Dependencias

```

=== ÚLTIMAS 50 LÍNEAS DEL LOG ===

Error: Cannot find module 'some-package'
Require stack:
- /app/node_modules/...

```

### Si hay Error de Next.js Config

```

=== ÚLTIMAS 50 LÍNEAS DEL LOG ===

Error: Invalid next.config.js
output must be one of: 'standalone', 'export'

```

---

## Con Estos Logs Podremos

1. **Ver el error exacto** → TypeScript, webpack, config, etc.
2. **Identificar el módulo problemático** → @prisma/client, otro package, etc.
3. **Ver el stack trace completo** → Dónde ocurre el error
4. **Aplicar fix específico** → Según el error encontrado

## Posibles Errores y Soluciones

### Error 1: @prisma/client No Encontrado

Log:

```
Module not found: Can't resolve '@prisma/client'
```

**Causa:** Prisma Client no se generó o no está en node\_modules

**Solución:**

- Verificar que `npx prisma generate` se ejecutó exitosamente
- Revisar que node\_modules/@prisma/client existe

### Error 2: Error de TypeScript

Log:

```
Type error: Property 'xyz' does not exist
```

**Causa:** Error de tipos en el código

**Solución:**

- Corregir el tipo en el archivo indicado
- O agregar `typescript.ignoreBuildErrors: true` temporalmente

### Error 3: Variable de Entorno Requerida

Log:

```
Error: DATABASE_URL is required
```

**Causa:** Next.js requiere la variable aunque tengamos SKIP\_ENV\_VALIDATION

**Solución:**

- Ya tenemos DATABASE\_URL definida como ENV
- Verificar que esté disponible durante el build

### Error 4: Dependencia Faltante

Log:

```
Cannot find module 'package-name'
```





**Causa:** Package no instalado

**Solución:**

- Agregar a package.json
- Ejecutar npm install

## Información de Debug Incluida

Cuando el build falle, v8.8 mostrará:

1.  **Exit code del npm run build**
2.  **Últimas 50 líneas del output** (incluye el error)
3.  **Lista de archivos en /app** (para verificar estructura)
4.  **Contenido de .next/** (si existe parcialmente)

## Próximos Pasos

### 1. Rebuild con v8.8

**Commit:** Próximo




**Versión:** 8.8 (captura de errores mejorada)

### 2. Copiar TODOS los Logs

**IMPORTANTE:** Necesito ver:

```
=== BUILD NEXT.JS ===
[... todo el output ...]
Build exit code: X
❌ BUILD FALLÓ - Exit code: X
=== ÚLTIMAS 50 LÍNEAS DEL LOG ===
[... las últimas 50 líneas ...]
=== ARCHIVOS EN /app ===
[... lista de archivos ...]
```

### 3. Con Esos Logs:

-  Identificaré el error exacto
-  Aplicaré el fix específico
-  Resolveremos el problema definitivamente

## Por Qué Este Enfoque

### Problema con v8.7

```
npm run build && ... || (echo error; exit 1)
```





**Limitaciones:**

- El `&&` y `||` pueden no capturar todos los escenarios
- No guarda el output para análisis
- Difícil debuggear

**Solución en v8.8**

```
npm run build 2>&1 | tee /tmp/build-output.log;
BUILD_EXIT_CODE=${PIPESTATUS[0]};
if [ $BUILD_EXIT_CODE -ne 0 ]; then
    tail -50 /tmp/build-output.log;
fi
```


**Ventajas:**

-  Captura exit code real ( PIPESTATUS )
-  Guarda output completo en archivo
-  Muestra contexto relevante si falla
-  Más robusto y confiable

**Resumen**

**v8.8 no arregla el error del build**, sino que **nos permitirá VER el error** para poder arreglarlo.

**Proceso:**

1. v8.8 → Captura el error completo 
2. Analizar logs → Identificar causa raíz
3. v8.9 → Aplicar fix específico
4. Build exitoso 🎉

**Versión:** 8.8

**Fecha:** 2025-10-08 02:30 GMT

**Estado:** 🔍 CAPTURA DE ERRORES MEJORADA

**Cambio principal:**

- Mejora captura de errores de build
- Muestra últimas 50 líneas del log
- Captura exit code con PIPESTATUS
- Lista archivos para debug

**Objetivo:** Ver el error real para poder arreglarlo.

**Próximo paso:** Rebuild y compartir TODOS los logs del error para diagnóstico preciso.