



# FIX CRÍTICO: Build de Next.js sin Standalone

**Fecha:** 2025-10-11

**Problema:** Error "Could not find a production build in the '.next' directory"

## ✗ Problema Identificado

El contenedor de Docker no encontraba el build de producción de Next.js:

```
Error: Could not find a production build in the '.next' directory.
Try building your app with 'next build' before starting the production server.
```

## Causa Raíz

El `next.config.js` tenía:

```
output: process.env.NEXT_OUTPUT_MODE,
```

Si `NEXT_OUTPUT_MODE` se configuraba como `"standalone"`, Next.js generaba una estructura diferente:

- `.next/standalone/` con `server.js`
- Pero el Dockerfile copiaba `.next/` normalmente
- El servidor no encontraba los archivos necesarios

## ✓ Solución Implementada

### 1. Dockerfile Actualizado

**Cambios clave:**

1. **Forzar build normal (no standalone):**

```
# CRITICAL: Do NOT use standalone output
ENV NEXT_OUTPUT_MODE=""
```

1. **Verificar build en etapa builder:**

```
RUN echo "🔍 Verifying build output..." && \
  if [ -f ".next/BUILD_ID" ]; then \
    echo "✅ Build ID found: $(cat .next/BUILD_ID)"; \
  else \
    echo "❌ BUILD_ID not found - build may have failed!"; \
    exit 1; \
fi
```

### 1. Verificar build en imagen final:

```
RUN echo "🔍 Verifying production files..." && \
ls -la /app/.next/ && \
if [ -f "/app/.next/BUILD_ID" ]; then \
  echo "✅ Production build verified: $(cat /app/.next/BUILD_ID)"; \
else \
  echo "❌ BUILD_ID missing in production image!"; \
  exit 1; \
fi
```

### 1. Instalar bash en la imagen base:

```
RUN apk add --no-cache libc6-compat openssl bash
```

## Estructura del Build Normal

Con `output` sin definir o vacío, Next.js genera:



Y el servidor se ejecuta con:

```
npm start      # o node_modules/.bin/next start
```

## Diferencia con Standalone

### Modo Normal (Actual)

- Build completo en `.next/`
- Requiere `node_modules/` completo
- Se ejecuta con `npm start` o `next start`

- **Más compatible y predecible**

## Modo Standalone (Problemático)

- Build en `.next/standalone/`
  - Solo incluye dependencias necesarias
  - Se ejecuta con `node server.js`
  - Requiere copiar estructura especial
- 

## Cómo Redeploy en Coolify

### 1. Hacer Push a GitHub

```
cd /home/ubuntu/muebleria_la_economica
git add Dockerfile DOCKER-BUILD-NO-STANDALONE-FIX.md
git commit -m "fix: force normal Next.js build (no standalone)"
git push origin main
```

### 2. Redeploy en Coolify

- Ve a tu aplicación en Coolify
- Haz clic en “**Redeploy**”
- El nuevo Dockerfile se descargará y compilará
- Verifica los logs durante el build:
- “Build ID found: [id]”
- “Production build verified: [id]”

### 3. Verificar la Aplicación

Una vez desplegado, verifica:

```
# 1. Check container logs
docker logs [container-name]

# 2. Check if server is responding
curl http://localhost:3000/api/health

# 3. Check from outside
curl https://app.mueblerialaeconomica.com
```

## Verificaciones del Dockerfile

El nuevo Dockerfile incluye verificaciones automáticas:

### 1. Durante el build:

- Verifica que `BUILD_ID` existe
- Si falla, muestra el contenido de `.next/`
- Detiene el build si hay error

## 2. En la imagen final:

- Verifica que `BUILD_ID` se copió correctamente
- Lista el contenido de `/app/.next/`
- Detiene el build si falta el archivo

## 3. Durante el inicio:

- El script `start.sh` ya verifica la conexión a la base de datos
  - Genera el cliente Prisma si es necesario
  - Inicia el servidor con `npm start`
- 

## Resultado Esperado

Después de este fix:

1.  El build de Next.js se completará correctamente
  2.  El archivo `BUILD_ID` existirá en `/app/.next/`
  3.  El servidor encontrará todos los archivos necesarios
  4.  La aplicación iniciará sin errores
  5.  Coolify/Traefik podrá enrutar el tráfico correctamente
- 

## Notas Importantes

### Por qué NO usar Standalone en este caso

1. **Complejidad innecesaria:** La app ya funciona bien con build normal
2. **Problemas de estructura:** Requiere copiar archivos de múltiples ubicaciones
3. **Compatibilidad:** Prisma y otros módulos funcionan mejor con `node_modules` completo
4. **Debugging:** Más fácil diagnosticar problemas con estructura estándar

### Cuándo considerar Standalone

- Cuando necesites reducir el tamaño de la imagen dramáticamente
  - Para despliegues en serverless/edge functions
  - Cuando tengas experiencia con la estructura standalone de Next.js
- 

## Troubleshooting

### Si el error persiste

#### 1. Verificar variables de entorno en Coolify:

- NO debe haber `NEXT_OUTPUT_MODE=standalone`
- Verifica en Settings → Environment Variables

#### 2. Limpiar caché de Docker en Coolify:

- Settings → Build Options → Clear Build Cache
- Luego Redeploy

**3. Verificar logs del build:**

- Ve a Deployments → [último deployment] → Logs
- Busca “Build ID found” o errores

**4. SSH al contenedor:**

```
docker exec -it [container-name] sh  
ls -la /app/.next/  
cat /app/.next/BUILD_ID
```



## Referencias

- [Next.js Output File Tracing](https://nextjs.org/docs/app/api-reference/next-config-js/output) (<https://nextjs.org/docs/app/api-reference/next-config-js/output>)
- [Docker Multi-Stage Builds](https://docs.docker.com/build/building/multi-stage/) (<https://docs.docker.com/build/building/multi-stage/>)
- [Next.js Production Deployment](https://nextjs.org/docs/deployment) (<https://nextjs.org/docs/deployment>)

---

**Autor:** DeepAgent

**Status:** Listo para Deploy