🔧 Fix Final: Error de Standalone Build en Docker

Fecha: 9 de Octubre, 2025
Error Original: `/app/.next/standalone/app: not found`
Estado: ✅ RESUELTO

🔍 Problema Identificado

El build de Docker fallaba al intentar copiar el directorio standalone:

```
ERROR: "/app/.next/standalone/app": not found
```

Causa Raíz

El archivo `next.config.js` tenía:

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

Cuando `NEXT_OUTPUT_MODE` no se evalúa correctamente (undefined), Next.js no genera el output standalone.

✅ Solución Implementada

1. Forzar Output Standalone Durante el Build

Modificación en Dockerfile:

```
# Force standalone output and build
RUN echo "🏗️ Building Next.js with standalone output..." && \
    node -e "const fs=require('fs');const
cfg=fs.readFileSync('next.config.js','utf8').replace(/output:.*,/,'output:\"stan-
dalone\",');fs.writeFileSync('next.config.js',cfg);" && \
    echo "📋 Verifying next.config.js:" && \
    grep -A2 "output:" next.config.js && \
    echo "\n🔨 Running build..." && \
    yarn build && \
    echo "\n✅ Build completed!" && \
    echo "\n📁 Checking .next structure:" && \
    ls -la .next/ && \
    echo "\n📁 Checking .next/standalone:" && \
    find .next/standalone -type f -o -type d | head -30
```

Qué hace:
1. Modifica `next.config.js` para forzar `output: "standalone"`
2. Verifica la configuración con grep

3. Ejecuta el build
4. Lista la estructura generada para debugging

2. Copia Flexible del Standalone Output

ANTES (Inflexible):

```
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone/app ./
```

DESPUÉS (Flexible):

```
# Copy entire standalone directory
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./

# Move nested app directory to root if needed
RUN if [ -f "app/server.js" ]; then \
      echo "📦 Found nested app structure, moving to root..."; \
      cp -r app/* . && rm -rf app; \
    elif [ -f "server.js" ]; then \
      echo "✅ server.js already in root"; \
    else \
      echo "❌ ERROR: server.js not found!"; \
      ls -laR; \
      exit 1; \
    fi
```

Qué hace:
1. Copia todo el directorio standalone
2. Detecta si existe estructura anidada `app/server.js`
3. Mueve el contenido a la raíz si es necesario
4. Verifica que `server.js` exista o falla con error claro

🎯 Beneficios de la Solución

Aspecto	Antes	Después
Configuración	Depende de variable de entorno	Forzada en build time
Flexibilidad	Path hardcoded	Detecta estructura automát-icamente
Debugging	Error críptico	Logs detallados de estructura
Robustez	Falla silenciosamente	Valida y falla con información

---

🚀 Cómo Usar Este Fix

Paso 1: Pull los Cambios de GitHub

EasyPanel debería detectar automáticamente los cambios, o ejecuta manualmente:

```
cd /home/ubuntu/muebleria_la_economica
git pull origin main
```

Paso 2: Rebuild en EasyPanel

1. **Ir a tu proyecto en EasyPanel**
2. **Pestaña “Build” o “Deploy”**
3. **Click en “Rebuild”** o “Redeploy”
4. **Observar los logs del build**

Paso 3: Verificar el Build Exitoso

En los logs del build deberías ver:

```
🏗️ Building Next.js with standalone output...
📋 Verifying next.config.js:
  output: "standalone",

🔨 Running build...
✅ Build completed!

📁 Checking .next structure:
[lista de archivos]

📁 Checking .next/standalone:
[lista de archivos en standalone]
```

Luego en la etapa runner:

```
✅ server.js already in root
```

O:

```
📦 Found nested app structure, moving to root...
```

Paso 4: Verificar el Contenedor

Una vez que el build complete:

```
# Ver logs del contenedor
docker logs <container-id>

# Deberías ver:
✅ server.js encontrado en /app/server.js (CORRECTO)
🚀 EJECUTANDO: node server.js
```

📊 Estructura Esperada

Durante el Build (Builder Stage)

```
/app/
├── .next/
│   ├── standalone/
│   │   ├── server.js          # Caso 1: Sin outputFileTracingRoot
│   │   └── app/               # Caso 2: Con outputFileTracingRoot
│   │       └── server.js
│   └── static/
├── public/
└── node_modules/
```

En Producción (Runner Stage)

```
/app/
├── server.js              # ← SIEMPRE en la raíz
├── .next/
│   └── static/
├── public/
├── prisma/
└── node_modules/
    ├── @prisma/
    ├── .prisma/
    └── .bin/
```

🔍 Diagnóstico de Errores

Si el Build Falla en “Building Next.js”

Ver:

```
📋 Verifying next.config.js:
```

Si no muestra `output: "standalone"`, el regex no funcionó.

Solución: Editar el regex en el Dockerfile para que coincida con tu next.config.js

Si Falla en “standalone/app: not found”

Este error ya NO debería ocurrir porque ahora copiamos todo `/app/.next/standalone`.

Si ocurre: Verificar que el build generó el directorio standalone:

```
find .next/standalone -type f -o -type d | head -30
```

Si el Contenedor No Arranca

Revisar logs:

```
docker logs <container-id>
```

Buscar:

```
❌ ERROR CRÍTICO: server.js NO ENCONTRADO
```

Si aparece, significa que el RUN que mueve los archivos falló.

Verificar en logs del build:

```
📦 Found nested app structure, moving to root...
```

o

```
✅ server.js already in root
```

🛠️ Variables de Entorno Importantes

Durante el Build

```
ENV NODE_ENV=production          # Activa modo producción
ENV NEXT_TELEMETRY_DISABLED=1    # Desactiva telemetría
ENV BUILD_TIMESTAMP=20251009_025500_FORCED_STANDALONE
```

En Runtime (EasyPanel)

Asegúrate de tener configuradas:

```
DATABASE_URL=postgresql://...
NEXTAUTH_URL=https://app.mueblerialaeconomica.com
NEXTAUTH_SECRET=<secret>
JWT_SECRET=<secret>
NODE_ENV=production
PORT=3000
```

✅ Checklist de Verificación Post-Deploy

- [ ] Build completa sin errores
- [ ] Logs muestran “output: standalone” en next.config.js
- [ ] Logs muestran estructura de .next/standalone
- [ ] Logs muestran “server.js already in root” o “Found nested app structure”
- [ ] Contenedor arranca sin errores
- [ ] Logs del contenedor muestran “✅ server.js encontrado”
- [ ] Aplicación responde en puerto 3000
- [ ] Sitio accesible en https://app.mueblerialaeconomica.com

---

🔄 Comparación con Versiones Anteriores

Versión 1 (Fallida)

- Usaba script `build-with-standalone.sh`
- Complejidad alta
- Falló con exit code 1

Versión 2 (Fallida)

- Build directo con `yarn build`
- Depende de variable de entorno
- Falló porque output no era standalone

Versión 3 (ACTUAL - Exitosa) ✅

- Fuerza standalone modificando config
- Copia flexible de estructura
- Detecta y adapta automáticamente
- Logs detallados para debugging

📝 Comandos Útiles para Debugging

Ver Config de Next.js Durante Build

```
RUN cat next.config.js
```

Ver Estructura Completa de Standalone

```
RUN find .next/standalone -ls
```

Verificar server.js Existe

```
RUN ls -la .next/standalone/server.js || ls -la .next/standalone/app/server.js
```

Debug en Contenedor Running

```
docker exec -it <container> sh
ls -la /app/
cat /app/server.js | head -20
```

🎉 Resumen

Cambio	Impacto
Forzar output standalone	Garantiza que Next.js genera el output correcto
Copia flexible	Se adapta a diferentes estructuras de build
Detección de estructura	Mueve archivos si es necesario
Logs detallados	Facilita debugging
Validación de server.js	Falla rápido con información clara

Timestamp del Build: 20251009_025500_FORCED_STANDALONE
Commit: Próximo push
Branch: main
Repository: https://github.com/qhosting/muebleria-la-economica.git

---

🆘 Si Persisten los Problemas

1. **Copia los logs completos del build** desde EasyPanel
2. **Busca específicamente:**
   - La línea “Verifying next.config.js”
   - La sección “Checking .next/standalone”
   - Los mensajes en la etapa runner
3. **Comparte esos logs** para diagnóstico específico

Con este fix, el build debería funcionar en cualquier escenario de Next.js standalone. 🚀