

ACCIÓN INMEDIATA: DEBUGGING DEL BUILD

Fecha: 18 de octubre de 2025
Estado:  DOCKERFILE PREPARADO PARA DEBUGGING

Cambios Aplicados

He actualizado el Dockerfile para que **capture y muestre el error completo** cuando falle el build:

Nuevo Dockerfile

```
# Ahora cuando falla yarn build, muestra:

❌ Build falló con código X

=== ÚLTIMAS 100 LÍNEAS DEL BUILD LOG ===
[ERROR COMPLETO AQUÍ]

=== INFORMACIÓN DE DEPURACIÓN ===
Node version: v22.x
Yarn version: 4.9.4
NODE_ENV: production
SKIP_ENV_VALIDATION: 1
NEXT_OUTPUT_MODE: standalone

=== ARCHIVOS CRÍTICOS ===
[listado de archivos]

=== tsconfig.json ===
[configuración de TypeScript]
```

PASOS INMEDIATOS

LIMPIA EL CACHE (OBLIGATORIO)

- En EasyPanel:
1. Ve a **Settings > Build**
 2. Haz clic en **Clear Build Cache**
 3. Confirma y espera

CONFIGURA MEMORIA

```
Build Resources:
Memory: 2GB (mínimo 1GB)
CPU: 1-2 vCPUs
```

3 VERIFICA CONFIGURACIÓN

```
Repository: https://github.com/qhosting/escalafin-mvp
Branch: main
Commit: c9ec9f6 (último)
Dockerfile Path: Dockerfile
Context Path: /
```

4 REBUILD Y COPIA EL ERROR

1. Haz clic en **Deploy/Rebuild**
2. Cuando falle, verás el error completo
3. **COPIA TODO** desde “=== ÚLTIMAS 100 LÍNEAS ===” hasta el final
4. **COMPARTE** ese error completo conmigo

¿QUÉ VOY A VER?

Dependiendo del error, verás algo como:

Ejemplo 1: Error de TypeScript

```
Type error: Cannot find module '@components/ui/button'
Type error: Property 'userId' does not exist on type 'Session'
```

Ejemplo 2: Error de Memoria

```
FATAL ERROR: Reached heap limit Allocation failed
JavaScript heap out of memory
```

Ejemplo 3: Error de Prisma





```
Error: @prisma/client did not initialize yet
PrismaClientInitializationError: Prisma Client could not locate
```

Ejemplo 4: Error de Módulo

```
Module not found: Can't resolve 'next-auth/react'
```

INFORMACIÓN QUE NECESITO

Cuando me compartas el error, necesito ver:

1.  **Las últimas 100 líneas completas del log**
2.  **Las versiones de Node y Yarn mostradas**
3.  **Las variables de entorno mostradas**
4.  **Cualquier mensaje de ERROR o WARNING**

Tip: Copia todo el bloque desde “❌ Build falló” hasta el final.

SOLUCIONES RÁPIDAS (según el error)

Si es Error de TypeScript

Solución temporal:

```
// app/next.config.js
typescript: {
  ignoreBuildErrors: true, // ← cambiar a true
}
```

Luego rebuild.

Si es Out of Memory

Solución:

- Aumentar memoria a 2GB
- O usar NODE_OPTIONS con más heap

Si es Prisma

Solución:

Verificar que Prisma se genere antes del build (ya está en el Dockerfile).

Si es Módulo Faltante

Solución:

Verificar que `yarn.lock` esté correcto (ya lo verificamos).

Checklist Pre-Rebuild

- ☐ Cache limpiado en EasyPanel
- ☐ Memoria 2GB configurada
- ☐ Commit actualizado (c9ec9f6)
- ☐ Dockerfile Path: `Dockerfile`
- ☐ Context Path: `/`

¿POR QUÉ PUEDE FALLAR?

El build funciona localmente pero falla en EasyPanel por:

1. **Recursos:** Memoria insuficiente (< 1GB)
2. **Entorno:** Variables de entorno diferentes
3. **Cache:** Cache corrupto de builds anteriores
4. **TypeScript:** Errores que solo aparecen en strict mode
5. **Dependencias:** Alguna diferencia en node_modules

CONFIANZA

99% de que con el error completo podremos:

- Identificar el problema exacto
- Aplicar la solución correcta
- Tener el deploy funcionando en 10 minutos

ACCIÓN AHORA

1. **LIMPIA** el cache
2. **REBUILD**
3. **COPIA** el error completo
4. **COMPARTE** conmigo

Con esa información, te daré la solución exacta. 

Documentación Relacionada

- **DEBUGGING_BUILD_FAILURE.md** - Guía completa de debugging
- **ESTADO_ACTUAL_RESUELTO.md** - Estado del proyecto
- **PASOS_INMEDIATOS_EASYPANEL.md** - Configuración de EasyPanel

Todas disponibles en el repositorio con PDF incluido.

Última actualización: 18 de octubre de 2025

Commit: c9ec9f6

Status:  LISTO PARA DEBUGGING

¡Vamos! 