🔍 DEBUGGING: Build Failure en EasyPanel

🎯 Situación Actual

Error:

```
yarn build - exit code: 1
```

Estado:
- ✅ Build funciona localmente
- ✅ yarn.lock es archivo regular (no symlink)
- ❌ Falla en EasyPanel

🔍 Nuevo Dockerfile con Debugging

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

```
# Ahora captura y muestra:
- ✅ Últimas 100 líneas del build log
- ✅ Versiones de Node y Yarn
- ✅ Variables de entorno
- ✅ Archivos críticos
- ✅ tsconfig.json
```

🚀 PASOS PARA DEBUGGING

1️⃣ Pull y Rebuild

En EasyPanel:

1. **Limpia el cache** (crítico):
   - Settings > Build > Clear Build Cache

2. **Pull latest commit**:
   - Verifica que esté en commit `[actual]`

3. **Rebuild** y observa los logs

2️⃣ Busca en los Logs

Cuando falle, verás:

```
❌ Build falló con código 1

=== ÚLTIMAS 100 LÍNEAS DEL BUILD LOG ===
[aquí verás el error COMPLETO]
```

Busca líneas con:
- `Error:`

- `Type error:`
- `Cannot find module`
- `FATAL ERROR`
- `Allocation failed`

3️⃣ Posibles Errores y Soluciones

🔴 Error de TypeScript

Síntoma:

```
Type error: Cannot find module '@/...'
Type error: Property 'X' does not exist on type 'Y'
```

Causa: Diferencias en el entorno de build

Solución temporal:
Modifica `next.config.js`:

```
typescript: {
  ignoreBuildErrors: true, // ← cambiar a true temporalmente
}
```

🔴 Out of Memory

Síntoma:

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

Solución:
- Aumenta memoria a **2GB mínimo**
- Si no es posible, reduce el tamaño del build:
`javascript`
`// next.config.js`
`experimental: {`
`optimizeCss: true,`
`optimizePackageImports: ['@radix-ui/react-icons'],`
`}`

🔴 Prisma Error

Síntoma:

```
Error: @prisma/client did not initialize yet
Cannot find Prisma Client
```

Solución:
Verifica que `npx prisma generate` se ejecute ANTES de `yarn build`

🔴 Missing Environment Variable

Síntoma:

```
Error: DATABASE_URL is not defined
Error: NEXTAUTH_URL is not defined
```

Solución:
Aunque tenemos `SKIP_ENV_VALIDATION=1`, algunas variables son requeridas en build-time:

```
# En EasyPanel, configura estas variables de BUILD:
SKIP_ENV_VALIDATION=1
DATABASE_URL=postgresql://dummy:dummy@localhost:5432/dummy
```

🔴 Module Not Found

Síntoma:

```
Module not found: Can't resolve '@/components/...'
Module not found: Can't resolve 'next-auth'
```

Solución:
1. Verifica que `node_modules` se copie correctamente
2. Verifica que `yarn.lock` exista
3. Limpia cache y rebuild

🎯 CONFIGURACIÓN RECOMENDADA EN EASYPANEL

Build Settings

```
Dockerfile Path: Dockerfile
Context Path: /
Build Cache: LIMPIAR antes de rebuild
```

Build Arguments (si es necesario)

```
# Solo si el build falla por variables de entorno
SKIP_ENV_VALIDATION=1
DATABASE_URL=postgresql://dummy@localhost/dummy
```

Resources

```
Memory: 2GB (mínimo 1GB)
CPU: 1-2 vCPUs
Timeout: 600 segundos (10 minutos)
```

📊 Checklist de Verificación

Antes de rebuild, confirma:

- [ ] Cache limpiado en EasyPanel
- [ ] Commit actualizado (pull latest)
- [ ] Memoria configurada (2GB)

- [ ] Dockerfile Path: `Dockerfile`
- [ ] Context Path: `/`
- [ ] No hay variables de build extrañas

🔍 Análisis del Error

Una vez que tengas el error completo de los logs:

1. **Cópialo completo** (últimas 100 líneas)
2. **Identifica el tipo** (TypeScript, Memory, Prisma, etc.)
3. **Aplica la solución** correspondiente de arriba
4. **Comparte el error** conmigo si no está en esta lista

💡 Alternativa: Ignorar Errores de Build Temporalmente

Si el build falla por errores de TypeScript que no son críticos:

```
// app/next.config.js
module.exports = {
  // ...
  typescript: {
    ignoreBuildErrors: true, // ← Temporal
  },
  eslint: {
    ignoreDuringBuilds: true, // ← Ya está
  },
};
```

⚠️ **Importante:** Esto es solo para debugging. Una vez que funcione el deploy, debemos corregir los errores de TypeScript reales.

🎯 Próximos Pasos

1. **Haz rebuild** con el nuevo Dockerfile
2. **Copia el error completo** de los logs
3. **Identifica el tipo** de error
4. **Aplica la solución**
5. Si persiste, **comparte el error** conmigo

Confianza: Una vez que veamos el error completo, podremos solucionarlo en 5 minutos.

El nuevo Dockerfile está diseñado específicamente para mostrar toda la información que necesitamos para debugging. 🔍