

ESTRATEGIA DE DEBUG SISTEMÁTICO PARA EASYPANEL

PROBLEMA ACTUAL


Basándonos en las capturas de pantalla subidas:

1. **Error de validación en EasyPanel:** Campo “Ruta de compilación” está vacío (Required)
2. Múltiples intentos fallidos de deploy
3. No está claro qué parte específica falla (backend, frontend, o configuración)

SOLUCIÓN PASO A PASO




FASE 1: CONFIGURACIÓN CORRECTA EN EASYPANEL

1.1 Configuración del Repositorio GitHub

```
Propietario: ghosting
Repositorio: escalafin-mvp
Rama: main
Ruta de compilación: /app  IMPORTANTE: No dejar vacío
```

1.2 Método de Compilación

Opción Recomendada: Dockerfile

-  Más control sobre el proceso de build
-  Consistente con desarrollo local
-  Permite debugging granular

1.3 Variables de Entorno Críticas (EasyPanel)

```
 Database
DATABASE_URL=postgresql://password@postgres:5432/escalafin?schema=public

 NextAuth
NEXTAUTH_URL=https://tu-dominio.com
NEXTAUTH_SECRET=<generar-secreto-seguro>

 AWS S3
AWS_BUCKET_NAME=<tu-bucket>
AWS_FOLDER_PREFIX=escalafin/
AWS_REGION=us-east-1

 Openpay
OPENPAY_MERCHANT_ID=<tu-merchant-id>
OPENPAY_PRIVATE_KEY=<tu-private-key>
OPENPAY_PUBLIC_KEY=<tu-public-key>

 Build
NODE_ENV=production
NEXT_OUTPUT_MODE=standalone
SKIP_ENV_VALIDATION=1
NEXT_TELEMETRY_DISABLED=1
```

FASE 2: DOCKERFILE PASO A PASO (DEBUGGING)

Vamos a crear Dockerfiles incrementales para detectar exactamente dónde falla.

Estrategia de 3 Dockerfiles:

1. **Dockerfile.step1-backend** → Solo pruebas de Prisma/DB
2. **Dockerfile.step2-frontend** → Solo build de Next.js
3. **Dockerfile.step3-full** → Build completo integrado

FASE 3: PROCESO DE TESTING SISTEMÁTICO

Test 1: Verificar Dependencias

```
cd /home/ubuntu/escalafin_mvp/app
npm install --legacy-peer-deps
```

✅ **Debe pasar:** Sin errores de dependencias

Test 2: Verificar Prisma

```
npx prisma generate
npx prisma validate
```

✅ **Debe pasar:** Client generado correctamente

Test 3: Verificar Build Local

```
npm run build
```

✅ **Debe pasar:** Build completo sin errores

Test 4: Verificar Docker Local

```
docker build -t escalafin-test -f Dockerfile.step1-backend .
docker build -t escalafin-test -f Dockerfile.step2-frontend .
docker build -t escalafin-test -f Dockerfile.step3-full .
```

✅ **Debe pasar:** Cada step exitoso antes de continuar

Test 5: Deploy en EasyPanel

Solo después de que todos los tests anteriores pasen.

CHECKLIST DE VERIFICACIÓN

Pre-Deploy

- ☐ Variables de entorno configuradas en EasyPanel
- ☐ Database PostgreSQL creada y accesible
- ☐ Bucket S3 configurado y credenciales válidas
- ☐ Credenciales Openpay válidas
- ☐ Ruta de compilación configurada: `/app`
- ☐ Método de compilación: Dockerfile

Durante el Build

- ☐ Logs de npm install exitosos
- ☐ Logs de prisma generate exitosos
- ☐ Logs de next build exitosos
- ☐ Archivos `.next/standalone` generados

Post-Deploy

- ☐ Contenedor inicia correctamente
- ☐ Healthcheck responde OK
- ☐ Página de login accesible
- ☐ No hay errores en logs del contenedor

DEBUGGING POR CAPAS

Si falla en npm install:

Problema: Dependencias incompatibles
 Solución: Usar `--legacy-peer-deps` y verificar `package-lock.json`

Si falla en prisma generate:

Problema: Schema inválido o DATABASE_URL incorrecta
 Solución: Verificar `prisma/schema.prisma` y variable DATABASE_URL

Si falla en next build:

Problema: Errores de TypeScript o variables de entorno faltantes
 Solución:




1. Revisar errores específicos en logs
2. Agregar variables faltantes
3. Verificar `SKIP_ENV_VALIDATION=1`

Si falla al iniciar contenedor:




Problema: `server.js` no encontrado o permisos incorrectos
 Solución: Verificar que `.next/standalone` fue generado correctamente

RECOMENDACIÓN FINAL

NO intentar deploy en producción hasta que:

1.  Todos los tests locales pasen
2.  Docker build local funcione 100%
3.  Hayas identificado la capa exacta que falla

Usar esta estrategia incremental evita:

-  Pérdida de tiempo en builds fallidos
-  Confusión sobre qué parte específica falla
-  Deployments parciales que rompen producción

PRÓXIMOS PASOS INMEDIATOS

1. **AHORA:** Crear los 3 Dockerfiles incrementales
2. **LUEGO:** Testear cada uno localmente
3. **FINALMENTE:** Deploy en EasyPanel solo cuando todo pase

¿Quieres que proceda a crear los Dockerfiles paso a paso?