

SOLUCIÓN: Error de Validación en EasyPanel

Basado en tus capturas de pantalla

Problema detectado: “Error de validación” - Falta configurar “Ruta de compilación”

ANÁLISIS DE TUS CAPTURAS

Imagen 1: Error de Validación (dok.jpg)

```
❌ Error de validación
Propietario: qhosting ✅
Repositorio: escalafin-mvp ✅
Rama: main ✅
Ruta de compilación: [VACÍO] ❌ ➡ ESTE ES EL PROBLEMA
```

Diagnóstico: El campo “Ruta de compilación” está vacío y es obligatorio.

Imagen 2: Opciones de Compilación (dok2.jpg)

Muestra 3 opciones:

- Dockerfile ← **Esta es la que necesitas**
- Buildpacks
- Nixpacks

Imagen 3: Configuración Correcta (esca.png)

```
✅ Configuración correcta:
Propietario: qhosting
Repositorio: escalafin-mvp
Rama: main
Ruta de compilación: / ✓ ← DEBE SER UNA BARRA "/"
```

Imagen 4: Coolify Dashboard (escal.jpg)

Muestra que tienes acceso a Coolify con “My first project”.

SOLUCIÓN INMEDIATA

Paso 1: Completar Configuración en EasyPanel

Según tu captura `dok.jpg`, necesitas completar este campo:

Ruta de compilación: `/`

Simplemente escribe una **barra diagonal** (`/`) en el campo que está vacío.

Paso 2: Seleccionar Método de Compilación

En la captura `dok2.jpg`, selecciona:

- ☒ **Dockerfile** (primera opción)
- ☐ Buildpacks
- ☐ Nixpacks

Paso 3: Archivo Dockerfile

El sistema debe detectar automáticamente el `Dockerfile` en la raíz del repositorio.

Contenido del Dockerfile (ya está en tu repo):

```
# Node 22 + Yarn 4.9.4
# Multi-stage build optimizado
# Standalone mode configurado
# Todas las correcciones aplicadas
```

Paso 4: Guardar Configuración

Después de completar:

```
Propietario: qhosting
Repositorio: escalafin-mvp
Rama: main
Ruta de compilación: /
Compilación: Dockerfile
```


Click en el botón **“Guardar”** (botón verde en tu captura).



CONFIGURACIÓN COMPLETA PASO A PASO

Tab: Subir (Source)

Propietario * qhosting	← Ya lo tienes
Repositorio * escalafin-mvp	← Ya lo tienes
Rama * main	← Ya lo tienes
Ruta de compilación * /	← AGREGAR ESTO

 `github.com / qhosting / escalafin-mvp`
Esta debe ser una rama válida en su repositorio

Tab: Compilación

- ☒ Dockerfile
Usa el comando "docker build" (docs)
- ☐ Buildpacks
Elija sus buildpacks deseados
- ☐ Nixpacks
Nueva forma de crear aplicaciones desde Railway (documentación)

Archivo:

Dockerfile

← Dejar por defecto

Esto es útil si tiene un monorepo

VARIABLES DE ENTORNO

Después de guardar la configuración, ir a **Settings** o **Variables de Entorno** y agregar:

Variables Mínimas Requeridas

```

DATABASE_URL=postgresql://role_edced812a:kzVbCZPfcYphJlIF5Y6qRXa7bRTR86gy@db-ed-
ced812a.db002.hosteddb.reai.io:5432/edced812a

NEXTAUTH_SECRET=tu-secret-aleatorio-muy-largo-y-seguro

NEXTAUTH_URL=https://tu-dominio.com
# (o http://tu-ip:puerto si no tienes dominio aún)

NODE_ENV=production

PORT=3000

HOSTNAME=0.0.0.0

```

Variables Opcionales (Openpay)

```

OPENPAY_MERCHANT_ID=tu-merchant-id
OPENPAY_PRIVATE_KEY=tu-private-key
OPENPAY_PUBLIC_KEY=tu-public-key
OPENPAY_BASE_URL=https://api.openpay.mx/v1
# o https://sandbox-api.openpay.mx/v1 para testing

```

Variables de Storage (Opcional)

Si usas storage local:

```

STORAGE_TYPE=local
LOCAL_UPLOAD_DIR=/app/uploads
LOCAL_BASE_URL=/api/files/serve
LOCAL_MAX_FILE_SIZE=10

```

Si usas AWS S3:

```
STORAGE_TYPE=s3
AWS_BUCKET_NAME=tu-bucket
AWS_REGION=us-east-1
AWS_ACCESS_KEY_ID=tu-access-key
AWS_SECRET_ACCESS_KEY=tu-secret-key
```



PROCESO DE DEPLOY

1. Después de Guardar Configuración

- El sistema debería iniciar el build automáticamente
- O click en botón “Deploy” / “Rebuild”

2. Monitorear Build

Deberías ver:

```
→ Cloning repository...
→ Building with Dockerfile...
→ [1/3] Installing dependencies...
→ [2/3] Building application...
→ [3/3] Starting runtime...
→ Deployment successful ✓
```

3. Verificar Logs de Runtime

Buscar:

```
▲ Next.js 14.2.28
- Local: http://0.0.0.0:3000
✓ Ready in XXXms
```

4. Configurar Dominio/Puerto

En **Settings** → **Network** o **Ports**:

```
Container Port: 3000
Public Port: 80 o 443 (según tu preferencia)
Domain: tu-dominio.com (opcional)
```

5. Acceder a la Aplicación

```
https://tu-dominio.com
# o
http://tu-ip-del-servidor
```

✖ ERRORES COMUNES Y SOLUCIONES

Error: “Ruta de compilación requerida”

Solución: Agregar `/` en el campo “Ruta de compilación”

Error: “No se encontró Dockerfile”

Solución:

- Verificar que “Dockerfile” esté seleccionado en “Compilación”
- Verificar que el archivo se llame exactamente `Dockerfile` (sin extensión)
- Verificar que esté en la raíz del repositorio (no en subcarpeta)

Error: “Build failed”

Solución:

1. Limpiar build cache
2. Verificar que el commit sea el más reciente (3989923)
3. Revisar logs de build para ver el error específico

Error: “Cannot find module ‘.prisma/client’”

Solución: Este error ya está resuelto en el commit actual (3989923). Asegúrate de usar el código más reciente.

Error: “Port 3000 already in use”

Solución:

- Verificar que no haya otro contenedor usando el puerto
- Cambiar el mapeo de puertos en la configuración



VERIFICACIÓN POST-DEPLOY

1. Health Check

```
curl https://tu-dominio.com/api/health

# Debe responder:
{"status": "ok"}
```

2. Página de Login

Abrir en navegador:

```
https://tu-dominio.com/auth/login
```

Debe mostrar:

- ☒ Formulario de login
- ☒ Estilos cargados correctamente
- ☒ Sin errores en consola del navegador

3. Credenciales de Prueba

Email: admin@escalafin.com
Password: admin123



RESUMEN DE CAMBIOS NECESARIOS

Tu configuración actual (según capturas):

```
Propietario: qhosting ✓
Repositorio: escalafin-mvp ✓
Rama: main ✓
- Ruta de compilación: [vacío] ✗
+ Ruta de compilación: / ✓

Compilación:
+ Dockerfile seleccionado ✓
```

Solo necesitas agregar la barra / y seleccionar Dockerfile.



CHECKLIST FINAL

Antes de hacer deploy, verifica:

- [] ☒ Propietario: qhosting
- [] ☒ Repositorio: escalafin-mvp
- [] ☒ Rama: main
- [] ☒ Ruta de compilación: /
- [] ☒ Compilación: Dockerfile seleccionado
- [] ☒ Variables de entorno configuradas
- [] ☒ Puerto 3000 expuesto
- [] ⚡ Click en "Guardar" y "Deploy"



SOPORTE

Si después de seguir estos pasos sigues teniendo el error:

1. Captura de pantalla de la configuración completa
2. Logs de build (si llega a iniciar)
3. Logs de runtime (si el contenedor inicia)
4. Mensaje de error exacto que aparece



DOCUMENTACIÓN RELACIONADA

- `REPORTE_VERIFICACION_LOCAL.md` - Validación del código
 - `COMANDOS_TEST_LOCAL_DOCKER.md` - Testing con Docker
 - `DIAGNOSTICO_RUNTIME_EASYPANEL.md` - Troubleshooting completo
 - `VARIABLES_ENTORNO_COMPLETAS.md` - Todas las variables disponibles
-

Estado:  Solución identificada - Solo falta agregar `/` en "Ruta de compilación"

Próximo paso: Completar configuración y hacer deploy

Código:  Listo en GitHub (commit 3989923)