

# Comandos de Validación - Guía Rápida

---

## Objetivo

---

Evitar que errores de configuración (como rutas absolutas en Prisma) vuelvan a romper el build en Docker.

---



## Comandos Principales

---

### 1. Validación Rápida (Recomendado)

```
cd app
./validate.sh
```

#### Ejecuta:

-  Validación de schema.prisma
-  Verificación de TypeScript






**Cuándo usarlo:** Antes de cada commit que modifique:

- prisma/schema.prisma
  - Archivos TypeScript con tipos de Prisma
  - Migraciones de base de datos
- 

### 2. Validación Solo Prisma

```
cd app
node scripts/validate-prisma-schema.js
```

#### Verifica:

-  No rutas absolutas en `output`
  -  No configuración `output` (debe usar default)
  -  Configuración correcta de `binaryTargets`
  -  Provider correcto ( `prisma-client-js` )
  -  DATABASE\_URL usando `env()`
- 

### 3. Validación Completa (Pre-Deploy)

```
cd app
bash scripts/pre-commit-check.sh
```

#### Ejecuta:

-  Validación de schema.prisma

- ☒ Verificación de TypeScript
- ☒ Verificación de Dockerfile
- ☒ Verificación de variables de entorno

**Cuándo usarlo:** Antes de hacer push a GitHub para deploy en Coolify

---

## 4. Build de Prueba Local

```
cd app
npm run build
```

**Cuándo usarlo:** Después de cambios importantes, para asegurar que el build funciona antes de deployar

---

## Flujo de Trabajo Seguro

---

### Opción A: Validación Manual (Recomendado)

```
# 1. Hacer cambios en el código
# ...

# 2. Validar antes de commit
cd app
./validate.sh

# 3. Si pasa, hacer commit
cd ..
git add -A
git commit -m "Tu mensaje"

# 4. Validación completa antes de push
cd app
bash scripts/pre-commit-check.sh

# 5. Si pasa, hacer push
cd ..
git push origin main
```

### Opción B: Con Git Hooks (Automático)

Si instalas Husky, las validaciones se ejecutarán automáticamente:

```
cd app

# Instalar Husky (una sola vez)
npm install --save-dev husky
npx husky-init
npx husky install

# Configurar el hook (una sola vez)
echo '#!/bin/sh' > .husky/pre-commit
cd app && ./validate.sh > .husky/pre-commit

chmod +x .husky/pre-commit
```

Después de esto, cada vez que hagas `git commit`, se ejecutará automáticamente la validación.

## Si la Validación Falla

### Error: “output path con ruta absoluta detectado”

#### Solución:

```
# Editar schema.prisma
nano prisma/schema.prisma

# Buscar esta línea y ELIMINARLA:
# output = "/home/ubuntu/..."

# El generador debe verse así:
# generator client {
#   provider = "prisma-client-js"
#   binaryTargets = ["native", "linux-musl-arm64-openssl-3.0.x"]
# }

# Regenerar Prisma Client
npx prisma generate

# Verificar de nuevo
node scripts/validate-prisma-schema.js
```

### Error: TypeScript

```
# Ver todos los errores
npx tsc --noEmit

# Corregir los errores reportados
# ...

# Verificar de nuevo
npx tsc --noEmit
```



## Integración con CI/CD

Si usas GitHub Actions o similar, agrega esto a tu workflow:

```

- name: Validate Prisma Schema
  run: |
    cd app
    node scripts/validate-prisma-schema.js

- name: TypeScript Check
  run: |
    cd app
    npx tsc --noEmit

- name: Build Test
  run: |
    cd app
    npm run build

```



## Reglas de Oro

1. **✓ SIEMPRE** ejecuta `./validate.sh` antes de commit
2. **✓ SIEMPRE** ejecuta `npm run build` localmente antes de push importante
3. **✗ NUNCA** modifiques `schema.prisma` sin validar después
4. **✗ NUNCA** uses rutas absolutas en `schema.prisma`
5. **✓ SIEMPRE** regenera Prisma Client después de cambiar schema: `npx prisma generate`



## Archivos de Validación Creados

```

app/
├── scripts/
│   ├── validate-prisma-schema.js  # Validación específica de Prisma
│   ├── pre-commit-check.sh        # Validación completa pre-commit
│   └── validate.sh                # Script de validación rápida
├── .husky/
│   └── pre-commit                 # Git hook (opcional)
├── REGLAS-PRISMA-DOCKER.md        # Documentación completa de reglas
└── COMANDOS-VALIDACION.md         # Esta guía

```



## Soporte

Si encuentras problemas:

1. Lee `REGLAS-PRISMA-DOCKER.md` para reglas detalladas
2. Ejecuta `node scripts/validate-prisma-schema.js` para diagnóstico
3. Revisa los logs del build en Coolify

4. Verifica que tu schema.prisma siga la configuración correcta

---

**Fecha de Creación:** 13 de octubre, 2025

**Versión:** 1.0