🛡️ SISTEMA PREVENTIVO DE VALIDACIÓN - Implementado

🎯 Objetivo

Prevenir errores recurrentes de configuración que causan fallos en el build de Docker, especialmente relacionados con el schema.prisma de Prisma.

❓ ¿Por Qué Se Implementó?

El usuario reportó que el error de "ruta absoluta en schema.prisma" ya se había resuelto anteriormente, pero volvió a aparecer. Esto indica que necesitamos **medidas preventivas automáticas** para que no vuelva a suceder.

🛠️ Componentes del Sistema

1️⃣ Script de Validación de Prisma

Archivo: `app/scripts/validate-prisma-schema.js`

Función:
- Detecta rutas absolutas en la configuración `output`
- Verifica que no exista configuración `output` (debe usar default)
- Valida que existan `binaryTargets` para Docker
- Verifica provider correcto
- Valida uso de `env("DATABASE_URL")`

Uso:

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

Salida cuando hay error:

```
❌ CRÍTICO: output path con ruta absoluta detectado (/home/ubuntu/...)
   Esto causará fallos en Docker. Elimina la línea "output" del generador.

💡 SOLUCIÓN RÁPIDA:
Edita prisma/schema.prisma y asegúrate que el generador se vea así:

generator client {
    provider = "prisma-client-js"
    binaryTargets = ["native", "linux-musl-arm64-openssl-3.0.x"]
    // ✅ NO incluir línea "output"
}
```

2️⃣ Script de Validación Rápida

Archivo: `app/validate.sh`

Función:
- Ejecuta validación de Prisma
- Ejecuta verificación de TypeScript
- Validación rápida antes de commits

Uso:

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

3️⃣ Script de Validación Completa

Archivo: `app/scripts/pre-commit-check.sh`

Función:
- Validación de schema.prisma
- Verificación de TypeScript
- Verificación de Dockerfile
- Verificación de variables de entorno

Uso:

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

4️⃣ Git Hook Pre-Commit (Opcional)

Archivo: `app/.husky/pre-commit`

Función:
- Se ejecuta **automáticamente** antes de cada commit
- Bloquea commits si las validaciones fallan
- Requiere Husky instalado

Instalación (opcional):

```
cd app
npm install --save-dev husky
npx husky-init
npx husky install
```

Nota: Si no instalas Husky, simplemente usa los scripts manualmente antes de commit.

📚 Documentación Creada

1. REGLAS-PRISMA-DOCKER.md

Documentación completa de:
- ❌ Qué NO hacer
- ✅ Qué SIEMPRE hacer
- 🔧 Scripts disponibles
- 🚀 Flujo de trabajo seguro
- 📋 Checklist pre-commit
- 🆘 Troubleshooting

2. COMANDOS-VALIDACION.md

Guía rápida de:
- Comandos principales
- Cuándo usar cada comando
- Flujo de trabajo manual vs automático
- Solución de errores comunes

🚀 Flujo de Trabajo Recomendado

Antes de Cada Commit

```
# 1. Validar rápidamente
cd app
./validate.sh

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

Antes de Push a Producción

```
# 1. Validación completa
cd app
bash scripts/pre-commit-check.sh

# 2. Build de prueba
npm run build

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

✅ Reglas de Oro

Para Prisma Schema

1. ✅ **NUNCA** incluir línea `output` en el generador

2. ✅ **SIEMPRE** usar ubicación predeterminada de Prisma
3. ✅ **SIEMPRE** incluir `binaryTargets` para Docker
4. ✅ **SIEMPRE** ejecutar validación después de modificar schema

Para Desarrollo

1. ✅ **SIEMPRE** ejecutar `./validate.sh` antes de commit
2. ✅ **SIEMPRE** ejecutar `npm run build` antes de push importante
3. ✅ **SIEMPRE** regenerar Prisma Client después de cambiar schema
4. ❌ **NUNCA** hacer push sin validar localmente

🔍 Cómo Funciona la Validación

Detección de Rutas Absolutas

```
// Detecta patrones como:
// output = "/home/ubuntu/..."
if (schemaContent.match(/output\s*=\s*["']\/home\/ubuntu/)) {
    errors.push('❌ CRÍTICO: output path con ruta absoluta detectado');
}
```

Detección de Output Path

```
// Detecta cualquier configuración output
if (schemaContent.match(/^\s*output\s*=/m)) {
    warnings.push('⚠️  ADVERTENCIA: output path detectado');
}
```

🎓 Beneficios del Sistema

1. Prevención Temprana

- Detecta errores **antes** de que lleguen a Docker
- Feedback inmediato al desarrollador
- Ahorra tiempo de debugging

2. Documentación Integrada

- Explica QUÉ está mal
- Muestra CÓMO corregirlo
- Proporciona solución inmediata

3. Automatización Opcional

- Puede funcionar manualmente
- Puede automatizarse con Git hooks
- Flexible según preferencias

4. Educación Continua

- Cada vez que falla, explica el problema
- Desarrolladores aprenden las reglas
- Menos errores con el tiempo

📊 Ejemplo de Uso

Escenario: Modificar Schema Prisma

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

# 2. Validar cambios
cd app
./validate.sh

# Salida:
# 🔍 Ejecutando validaciones completas...
#
# 🔍 Validando schema.prisma...
# ✅ Validación completada exitosamente!
#
# 📋 Configuración actual del generador:
# generator client {
#     provider = "prisma-client-js"
#     binaryTargets = ["native", "linux-musl-arm64-openssl-3.0.x"]
# }
#
# 🔷 Verificando TypeScript...
# ✅ Todas las validaciones completadas

# 3. Regenerar Prisma Client
npx prisma generate

# 4. Hacer commit
cd ..
git add -A
git commit -m "Update: Schema Prisma modificado"

# 5. Validación pre-deploy
cd app
bash scripts/pre-commit-check.sh

# 6. Push
cd ..
git push origin main
```

🔧 Mantenimiento del Sistema

Actualizar Reglas de Validación

Si necesitas agregar nuevas reglas:

1. Editar `app/scripts/validate-prisma-schema.js`
2. Agregar nueva verificación en la función `validatePrismaSchema()`
3. Probar con: `node scripts/validate-prisma-schema.js`
4. Actualizar documentación si es necesario

Ejemplo de Nueva Regla

```
// REGLA 7: Verificar que no use sqlite en producción
if (schemaContent.match(/provider\s*=\s*["']sqlite["']/)) {
    errors.push('❌ CRÍTICO: SQLite no debe usarse en producción');
}
```

📈 Estadísticas del Sistema

Archivos Creados

- 3 scripts de validación
- 1 hook de Git
- 2 documentos PDF
- 2 documentos Markdown

Líneas de Código

- ~200 líneas de JavaScript (validación)
- ~50 líneas de Bash (scripts)
- ~500 líneas de documentación

Validaciones Implementadas

- ✅ 6 reglas críticas para schema.prisma
- ✅ Verificación de TypeScript
- ✅ Verificación de Dockerfile
- ✅ Verificación de variables de entorno

🎯 Estado Actual

- ✅ Sistema implementado y probado
- ✅ Validación de Prisma funcionando correctamente
- ✅ Documentación completa creada
- ✅ Scripts con permisos de ejecución
- ✅ Schema.prisma corregido
- ✅ Cambios pusheados a GitHub (commit: b88dd39)

🚀 Próximos Pasos

Para el Usuario

1. **Familiarizarse con los comandos:**
   ```
   bash
      cd app
      ./validate.sh          # Uso diario
   ```

2. **Integrar en workflow:**
   - Ejecutar validación antes de cada commit
   - Validación completa antes de deploy

3. **Opcional - Instalar Husky:**
   ```
   bash
      cd app
      npm install --save-dev husky
      npx husky install
   ```

Para Coolify Deploy

1. Hacer redeploy en Coolify
2. El build debería completarse exitosamente
3. Verificar que el sitio funcione correctamente

📖 Recursos Adicionales

- **REGLAS-PRISMA-DOCKER.md** - Reglas detalladas
- **COMANDOS-VALIDACION.md** - Guía de comandos
- **REGLAS-PRISMA-DOCKER.pdf** - Versión imprimible
- **COMANDOS-VALIDACION.pdf** - Versión imprimible

💡 Consejos Finales

1. **Haz de la validación un hábito:**
   - Ejecuta `./validate.sh` frecuentemente
   - Es rápido (< 5 segundos)
   - Previene dolores de cabeza futuros

2. **Lee la documentación una vez:**
   - `REGLAS-PRISMA-DOCKER.md` tiene toda la info
   - Te ahorrará tiempo a largo plazo
   - Evitarás errores comunes

3. **Comparte con el equipo:**
   - Si trabajas en equipo, comparte las reglas
   - Todos deben usar el mismo flujo de trabajo
   - Consistencia = menos errores

Fecha de Implementación: 13 de octubre, 2025
Versión: 1.0
Commit: b88dd39
Estado: ✅ ACTIVO Y FUNCIONANDO