



Registro de Errores Resueltos

Historial completo de todos los errores encontrados y corregidos durante el deployment.

Error #1: yarn.lock Symlink Roto

Fecha: 2025-11-17

Commit: df36a47

Síntoma

```
ERROR: failed to calculate checksum of ref: "/app/yarn.lock": not found
```

Causa

- app/yarn.lock era un symlink a /opt/hostedapp/node/root/app/yarn.lock
- El archivo no existía en el contexto de Docker build

Solución

1. Detectar symlinks: [-L "app/yarn.lock"]
2. Eliminar symlink: rm app/yarn.lock
3. Recrear como archivo real: cd app && yarn install
4. Resultado: archivo real de 448KB

Prevención

- Check #1 en pre-deploy-check.sh
- Auto-fix: convierte symlinks automáticamente

Error #2: Alpine Linux 3.21 Repository Error

Fecha: 2025-11-17

Commit: c984e27

Síntoma

```
ERROR: unable to select packages:  
openssl-dev (no such package):  
    required by: world[openssl-dev]
```

Causa

- Alpine 3.21 tiene problemas con los repositorios de paquetes
- Algunos paquetes esenciales como openssl-dev no están disponibles

Solución

1. Cambiar base image en Dockerfile
 - Antes: FROM node:18-alpine3.21
 - Despues: FROM node:18-alpine3.19
2. Alpine 3.19 es stable y tiene todos los paquetes

Prevención

- Check #3.1 en pre-deploy-check.sh
- Detecta Alpine != 3.19

Error #3: Prisma Client - Enums No Exportados

Fecha: 2025-11-17

Commits: 7fa783b , e5d42c2

Síntoma

```
error TS2305: Module '@prisma/client' has no exported member 'UserRole'
error TS2305: Module '@prisma/client' has no exported member 'StatusCuenta'
error TS2305: Module '@prisma/client' has no exported member 'Periodicidad'
error TS2305: Module '@prisma/client' has no exported member 'TipoPago'
error TS2305: Module '@prisma/client' has no exported member 'MotivoMotarario'
```

Causa

- Prisma Client no se generó correctamente en Docker build
- Los enums del schema.prisma no estaban disponibles en tipos TypeScript

Solución

1. Asegurar que `prisma generate` se ejecute en build:

```
dockerfile
RUN ./node_modules/.bin/prisma generate
```

2. Validar que enums existan después de generar:

```
dockerfile
RUN node -e "const { UserRole, StatusCuenta, Periodicidad, TipoPago, MotivoMotarario } =
require('@prisma/client'); console.log('✓ Enums OK');"
```

3. Usar path directo a prisma CLI (no npx)

Prevención

- Check #2: Valida 5 enums en schema.prisma
- Check #3.4: Valida `prisma generate` en Dockerfile
- Check #3.5: Valida test de enums con `node -e`

Error #4: npm vs yarn Inconsistencia

Fecha: 2025-11-17

Commit: 1408449

Síntoma

```
npm ERR! Fix the upstream dependency conflict, or retry
npm ERR! this command with --force, or --legacy-peer-deps
```

Causa

- Proyecto usa `yarn.lock` pero `Dockerfile` usaba `npm install`
- Versiones de dependencias inconsistentes entre `npm` y `yarn`

Solución

1. Cambiar en `Dockerfile`:

- Antes: `RUN npm install --legacy-peer-deps`
- Despues: `RUN yarn install --frozen-lockfile`

2. `--frozen-lockfile` asegura versiones exactas del `yarn.lock`

Prevención

- Check #3.2: Detecta `yarn install` en `Dockerfile`
- Check #3.3: Valida `-frozen-lockfile` flag

Error #5: Scripts Sin Permisos de Ejecución

Fecha: 2025-11-17

Commits: varios

Síntoma

```
/bin/sh: ./start.sh: Permission denied
```

Causa

- Scripts bash no tenían permisos de ejecución (`+x`)
- Git no preserva permisos en algunos casos

Solución

1. Pre-deploy check detecta y corrige:

```
bash
[ -x "start.sh" ] || chmod +x start.sh
```

2. Scripts afectados:

- `start.sh`
- `seed-admin.sh`
- `backup-manual.sh`
- `restore-backup.sh`

Prevención

- Check #6 en pre-deploy-check.sh
- Auto-fix: ejecuta chmod +x automáticamente

Error #6: npx prisma Path Issues

Fecha: 2025-11-17

Commit: 7fa783b

Síntoma

```
Error: Cannot find module '@prisma/client'
Require stack:
- /app/...
```

Causa

- `npx prisma generate` puede no encontrar el CLI correcto
- Path resolution issues en entorno Docker

Solución

1. Usar path directo al binario:

```
```dockerfile
Antes
RUN npx prisma generate

Despues
RUN ./node_modules/.bin/prisma generate
```

```

1. Garantiza que usa la versión instalada localmente

Prevención

- Check #3.6 en pre-deploy-check.sh
- Detecta npx prisma y sugiere path directo

Error #7: Reinstalación de yarn en Alpine

Fecha: 2025-11-17

Commit: d872ef2

Síntoma

```
npm error EEXIST: file already exists
npm error File exists: /usr/local/bin/yarn
npm error Remove the existing file and try again, or run npm
npm error with --force to overwrite files recklessly.
```

Causa

- Dockerfile intentaba instalar yarn: `RUN npm install -g yarn`
- yarn ya viene preinstalado en node:18-alpine3.19
- npm intenta sobrescribir el binario existente

Solución

1. Eliminar línea del Dockerfile:

```
```dockerfile
Antes
RUN npm install -g yarn

Después
yarn ya viene preinstalado en node:18-alpine3.19
No es necesario instalar yarn nuevamente
```

```

1. Alpine node images incluyen yarn por defecto

Prevención

- Check #3.7 en pre-deploy-check.sh
- Detecta `npm install -g yarn` y rechaza con error

Resumen Estadístico

| Error | Tipo | Severidad | Auto-Fix | Estado |
|-------|-----------------|------------|----------|------------|
| #1 | Symlink | 🔴 Critical | ✅ Sí | ✅ Resuelto |
| #2 | Alpine version | 🔴 Critical | ✗ No | ✅ Resuelto |
| #3 | TypeScript | 🔴 Critical | ✗ No | ✅ Resuelto |
| #4 | Package manager | 🟡 Medium | ✗ No | ✅ Resuelto |
| #5 | Permisos | 🟡 Medium | ✅ Sí | ✅ Resuelto |
| #6 | CLI path | 🟡 Medium | ✗ No | ✅ Resuelto |
| #7 | Yarn EEXIST | 🔴 Critical | ✗ No | ✅ Resuelto |

Métricas

- **Total errores:** 7
- **Críticos:** 4
- **Medium:** 3
- **Auto-fix:** 2/7 (29%)
- **Manual fix:** 5/7 (71%)

Timeline

Nov 17, 2025

- 14:30 - Error #1: yarn.lock symlink
- 14:45 - Error #2: Alpine 3.21
- 15:00 - Error #3: Prisma enums
- 15:15 - Error #4: npm vs yarn
- 15:20 - Error #5: Script permisos
- 15:30 - Error #6: npx prisma
- 15:48 - Error #7: yarn EEXIST

Lecciones Aprendidas

1. Symlinks en Git

- ✗ **Problema:** Git puede commitear symlinks que no funcionan en Docker
✓ **Solución:** Pre-deploy check detecta y convierte a archivos reales

2. Alpine Version Stability

- ✗ **Problema:** Usar versión edge (3.21) causa package errors
✓ **Solución:** Usar versión LTS estable (3.19)

3. Prisma Client Generation

- ✗ **Problema:** Asumir que Prisma Client existe después de npm install
✓ **Solución:** Ejecutar explícitamente prisma generate + validar enums

4. Package Manager Consistency

- ✗ **Problema:** Mezclar npm y yarn en mismo proyecto
✓ **Solución:** Elegir uno y usarlo consistentemente (yarn + yarn.lock)

5. Permisos en Git

- ✗ **Problema:** Git no siempre preserva permisos de ejecución
✓ **Solución:** Auto-fix en pre-deploy check

6. npx vs Direct Path

- ✗ **Problema:** npx puede tener issues de path resolution
✓ **Solución:** Usar ./node_modules/.bin/ directamente

7. Preinstalaciones en Base Images

- ✗ **Problema:** Asumir que todo debe instalarse manualmente
✓ **Solución:** Verificar qué incluye la base image (yarn en node:alpine)

Herramientas de Prevención

1. pre-deploy-check.sh

- 18 verificaciones automáticas
- 2 auto-fixes

- Detecta los 7 errores históricos

2. Git Hook pre-push

- Ejecuta pre-deploy-check.sh antes de push
- Cancela push si hay errores
- Instalación: `bash install-git-hooks.sh`

3. Documentación

- PRE-DEPLOY-CHECKLIST.md - Guía de uso
 - PRE-DEPLOY-VERIFICATION-MAP.md - Mapa técnico
 - ERRORES-RESUELTOS.md - Este archivo
-

Última actualización: 2025-11-17

Versión: 1.0.0

Errores documentados: 7

Cobertura de prevención: 100%