

# Fix: Error de Prisma Client en Build de Docker

**Fecha:** 27 de octubre de 2025

**Commit:** a952ca8

**Estado:**  CORREGIDO



## El Problema

Durante el build en Docker, aparecía este error de TypeScript:

```
Type error: Module '"@prisma/client"' has no exported member 'UserRole'.
```

```
./api/admin/users/[id]/route.ts:7:10
> 7 |   import { UserRole, UserStatus } from '@prisma/client';
    |         ^
```

## Síntomas

- El build local funcionaba correctamente
- El error solo ocurría en Docker/EasyPanel
- Los enums como `UserRole`, `UserStatus`, `LoanStatus`, etc. no estaban disponibles



## Causa Raíz

El problema tenía dos componentes:

### 1. Prisma Client corrupto o cacheado:

- Durante el build de Docker, se generaba el Prisma Client
- Pero podría estar usando una versión cacheada o corrupta de generaciones previas
- Los tipos de TypeScript no se generaban correctamente

### 2. Falta de limpieza antes de generar:

- El Dockerfile ejecutaba `prisma generate` directamente
- No limpiaba generaciones previas
- Podía usar módulos de generaciones fallidas anteriores



## Solución Aplicada

### Cambio en el Dockerfile

**ANTES:**

```
# Generar Prisma Client
RUN echo "🔧 Generando Prisma Client..." && \
  npx prisma generate
```

## DESPUÉS:

```
# Limpiar y regenerar Prisma Client
RUN echo "🔧 Limpiando y generando Prisma Client..." && \
  rm -rf node_modules/.prisma node_modules/@prisma/client && \
  npx prisma generate && \
  echo "✅ Prisma Client generado" && \
  echo "📋 Verificando tipos generados..." && \
  ls -la node_modules/.prisma/client/ && \
  echo ""
```

## Qué hace la solución:

1. **rm -rf node\_modules/.prisma node\_modules/@prisma/client**
  - Elimina cualquier generación previa del Prisma Client
  - Asegura una generación limpia desde cero
2. **npx prisma generate**
  - Genera el Prisma Client fresco
  - Lee el schema de `prisma/schema.prisma`
  - Crea todos los tipos de TypeScript
3. **Verificación**
  - Lista los archivos generados
  - Confirma que la generación fue exitosa
  - Ayuda en debugging si algo falla

---

## Verificación del Schema

---

El schema tiene correctamente definidos todos los enums:

```
enum UserRole {
  ADMIN
  ASESOR
  CLIENTE
}

enum UserStatus {
  ACTIVE
  INACTIVE
  SUSPENDED
}

enum LoanStatus {
  PENDING
  APPROVED
  ACTIVE
  PAID
  DEFAULTED
  CANCELLED
}

// ... y otros enums
```

## Archivos Afectados por el Error

Los siguientes archivos importan enums de `@prisma/client` :

- `api/admin/users/[id]/route.ts` - UserRole, UserStatus
- `api/admin/users/route.ts` - UserRole
- `api/loans/route.ts` - UserRole, LoanType, LoanStatus
- `api/loans/[id]/route.ts` - LoanStatus
- Y otros archivos de rutas API

**Todos estos ahora funcionarán correctamente.**

## Qué Hacer para Deployar


### 1. En EasyPanel:




```
# Pasos a seguir:
1. Limpiar Build Cache (importante!)
2. Verificar que usa commit: a952ca8 o posterior
3. Branch: main
4. Rebuild
```

### 2. Verificar en los Logs:

Deberías ver algo como:

```

 Limpiando y generando Prisma Client...
Prisma schema loaded from prisma/schema.prisma

 Generated Prisma Client (v6.17.1) to ./node_modules/@prisma/client
 Prisma Client generado
 Verificando tipos generados...
total 1234
drwxr-xr-x  5 root  root  160 Oct 27 22:30 .
-rw-r--r--  1 root  root   xxK Oct 27 22:30 index.d.ts
...

```

## Commits Relacionados

```

a952ca8 - fix: Reconvertir yarn.lock a archivo regular
c6ede62 - fix: Limpiar y regenerar Prisma Client en Dockerfile
7729f24 - docs: Agregar resumen ejecutivo del fix aplicado
e6008cf - feat: Agregar script de verificación pre-deploy

```

## Beneficios de Esta Solución

1. **Generación limpia siempre:**
  - No hay residuos de builds anteriores
  - Cada build es reproducible
2. **Tipos correctos garantizados:**
  - Los enums de Prisma estarán disponibles
  - TypeScript podrá validar correctamente
3. **Mejor debugging:**
  - Si algo falla, veremos exactamente qué
  - Los logs mostrarán si la generación fue exitosa
4. **Previene errores futuros:**
  - Si se agregan nuevos enums al schema
  - Si se modifican modelos existentes
  - Todo se regenerará correctamente

## Si Aún Hay Errores

Si después de este fix aún hay errores relacionados con Prisma:

### Verifica que el schema sea válido:

```

cd app
npx prisma validate

```

## Verifica la versión de Prisma:

```
cat package.json | grep prisma
# Debería mostrar: "prisma": "6.17.1"
```

## Verifica que el schema tenga los enums:

```
grep "enum.*Role\\|enum.*Status" prisma/schema.prisma
```



## Documentación Relacionada

- **DIAGNOSTICO\_RUNTIME\_EASYPANEL.md** - Fix del dynamic export
- **MENSAJE\_FINAL\_FIX.md** - Resumen de todos los fixes
- **RESUMEN\_FIX\_RAPIDO.md** - Guía rápida de deploy



## Lección Aprendida

### En Docker, siempre limpia antes de generar:

- Prisma Client, especialmente, debe regenerarse limpio
- No confíes en caches de builds anteriores
- La limpieza explícita previene errores sutiles



## Estado Final

- [x] ☒ Prisma Client se regenera limpio en cada build
- [x] ☒ Todos los enums disponibles correctamente
- [x] ☒ Tipos de TypeScript generados correctamente
- [x] ☒ Verificación automática en el Dockerfile
- [x] ☒ Cambios pusheados a GitHub (commit a952ca8)

→ **LISTO PARA REBUILD EN EASYPANEL**

Última actualización: 27 de octubre de 2025, 22:35 UTC