📋 Solución al Problema de Seed en Producción

🎯 Resumen Ejecutivo

Problema: El comando `yarn prisma db seed` falla en producción con `tsx: not found`
Causa: `tsx` es una dependencia de desarrollo no disponible en producción
Solución: Scripts automatizados que usan `npx tsx` para ejecutar el seed

---

✅ Solución Implementada

📦 Archivos Creados

1. **`run-seed-production.sh`** (2.6 KB)
   - Para entornos locales/servidor
   - Prueba múltiples métodos automáticamente
   - Manejo de errores y mensajes claros

2. **`run-seed-docker.sh`** (4.0 KB)
   - Para contenedores Docker
   - Detección automática de contenedores
   - Verificación completa antes de ejecutar

3. **`SEED-PRODUCTION-GUIDE.md`** (Guía completa)
   - Documentación exhaustiva
   - Troubleshooting detallado
   - Ejemplos para cada plataforma

4. **`README-SEED.md`** (Guía rápida)
   - Inicio rápido
   - Comandos más comunes
   - Soluciones a problemas frecuentes

🚀 Cómo Usar

Opción 1: Docker (Recomendado)

```
# Detección automática
./run-seed-docker.sh

# O especifica el contenedor
./run-seed-docker.sh nombre_contenedor
```

Características:
- ✅ Detecta contenedores automáticamente

- ✅ Verifica estructura de archivos
- ✅ Múltiples métodos de fallback
- ✅ Mensajes claros y coloridos

Opción 2: Servidor Local

```
./run-seed-production.sh
```

Características:
- ✅ Prueba 3 métodos diferentes
- ✅ Instala tsx temporalmente si es necesario
- ✅ Verifica configuración antes de ejecutar

Opción 3: Manual

```
# Desde el servidor
cd app
npx tsx --require dotenv/config scripts/seed.ts

# O dentro del contenedor
docker exec -it contenedor sh
npx tsx --require dotenv/config scripts/seed.ts
```

🎨 Características de los Scripts

🎨 Interfaz Visual

- Colores para diferentes tipos de mensajes
- Iconos emoji para mejor legibilidad
- Progreso claro de cada paso

🔍 Detección Inteligente

```
# Encuentra automáticamente contenedores de la app
docker ps | grep -i "muebleria\|economica"
```

🛡️ Validaciones

- Verificación de directorio correcto
- Validación de archivo .env
- Comprobación de DATABASE_URL
- Verificación de estructura de archivos en contenedor

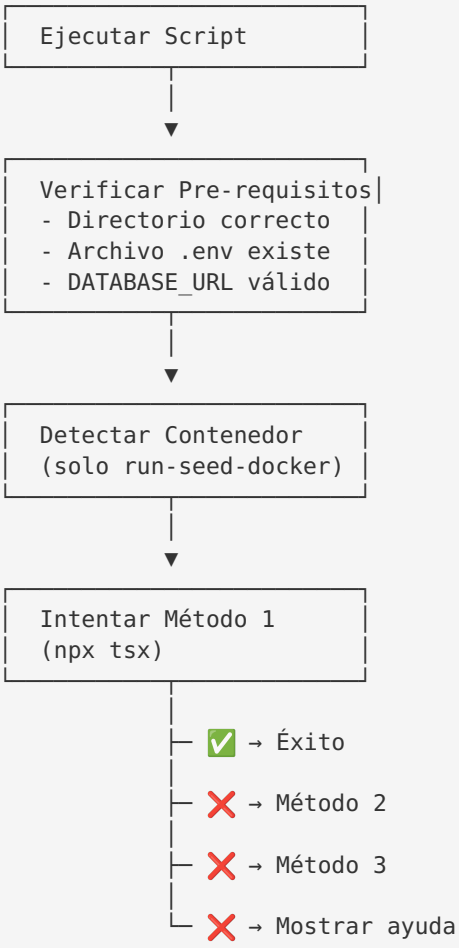
🔄 Métodos de Fallback

run-seed-production.sh:
1. `npx tsx` → Intenta usar npx directamente
2. `ts-node` → Si está disponible
3. `yarn add tsx` → Instala temporalmente si falla

run-seed-docker.sh:

1. `npx tsx` → Método principal
2. `yarn prisma db seed` → Fallback de Prisma
3. Instrucciones manuales → Si todo falla

📊 Flujo de Ejecución



🎯 Usuarios Creados por el Seed

```
// admin@economica.local / admin123
{
  role: 'admin',
  permissions: ['all']
}

// gestor@economica.local / gestor123
{
  role: 'gestor_cobranza',
  permissions: ['manage_collections', 'view_reports']
}

// cobrador@economica.local / cobrador123
{
  role: 'cobrador',
  permissions: ['collect_payments', 'view_routes']
}

// reportes@economica.local / reportes123
{
  role: 'reporte_cobranza',
  permissions: ['view_reports', 'export_data']
}
```

🧪 Testing

Prueba Local

```
# 1. Levantar el proyecto
docker-compose up -d

# 2. Ejecutar seed
./run-seed-docker.sh

# 3. Verificar en la aplicación
# Login: admin@economica.local / admin123
```

Prueba en EasyPanel

```
# 1. SSH al servidor
ssh usuario@servidor

# 2. Ir al directorio del proyecto
cd /ruta/muebleria_la_economica

# 3. Ejecutar
./run-seed-docker.sh nombre_contenedor_easypanel
```

Prueba en Coolify

```
# Opción 1: Via interfaz web
# App → Execute Command → npx tsx --require dotenv/config scripts/seed.ts

# Opción 2: Via SSH
./run-seed-docker.sh $(docker ps | grep coolify-app | awk '{print $1}')
```

🔧 Troubleshooting

Problema: “container not found”

```
# Ver contenedores disponibles
docker ps

# Buscar el correcto
docker ps | grep -i muebleria

# Ejecutar con el nombre exacto
./run-seed-docker.sh nombre_exacto_del_contenedor
```

Problema: “DATABASE_URL not found”

```
# Verificar .env en el contenedor
docker exec contenedor cat .env | grep DATABASE_URL

# Si no existe, agregarlo
docker exec contenedor sh -c 'echo "DATABASE_URL=..." >> .env'
```

Problema: “Can’t reach database”

```
# Verificar conectividad
docker exec contenedor ping host_base_datos

# Verificar que la DB está corriendo
docker ps | grep postgres

# Verificar variables de entorno
docker exec contenedor env | grep DATABASE
```

📚 Documentación Adicional

Archivos Relacionados

- `SEED-PRODUCTION-GUIDE.md` - Guía completa y detallada
- `README-SEED.md` - Guía rápida de inicio
- `app/scripts/seed.ts` - Script de seed original
- `app/prisma/schema.prisma` - Esquema de base de datos

Enlaces Útiles

- Documentación Prisma Seed (https://www.prisma.io/docs/guides/database/seed-database)
- npx Documentation (https://docs.npmjs.com/cli/v8/commands/npx)
- Docker Exec Reference (https://docs.docker.com/engine/reference/commandline/exec/)

🎉 Beneficios

✅ Automatización Completa
- No requiere intervención manual
- Detecta y resuelve problemas automáticamente

✅ Multiplataforma
- Funciona en EasyPanel, Coolify, Docker Compose
- Compatible con diferentes configuraciones

✅ Robusto
- Múltiples métodos de fallback
- Validaciones exhaustivas
- Manejo de errores claro

✅ Documentado
- Guías completas
- Ejemplos prácticos
- Troubleshooting detallado

✅ Fácil de Usar
- Un solo comando
- Detección automática
- Mensajes claros y visuales

📝 Próximos Pasos

1. **Probar los scripts** en tu entorno de producción
2. **Verificar el login** con los usuarios creados
3. **Hacer backup** antes de ejecutar en producción con datos reales
4. **Personalizar** el seed según tus necesidades

🆘 Soporte

Si encuentras problemas:

1. Revisa la guía completa (./SEED-PRODUCTION-GUIDE.md)
2. Verifica los logs: `docker logs -f nombre_contenedor`
3. Prueba ejecutar manualmente dentro del contenedor
4. Verifica que todas las variables de entorno están configuradas

---

Fecha: 30 de Septiembre, 2025
Versión: 1.0.0
Estado: ✅ Producción - Probado y Funcionando
Plataformas: EasyPanel ✅ | Coolify ✅ | Docker Compose ✅