

INSTRUCCIONES: Ejecutar Seed en Producción

INICIO RÁPIDO (3 pasos)

1 SSH a tu servidor

```
ssh usuario@tu-servidor.com  
cd /ruta/a/muebleria_la_economica
```

2 Ejecutar el script

```
# Para Docker (EasyPanel, Coolify, etc.)  
./run-seed-docker.sh  
  
# 0 especifica el contenedor  
./run-seed-docker.sh nombre_del_contenedor
```

3 ¡Listo! Verifica el login

- URL: <https://app.mueblerialaeconomica.com>
- Usuario: `admin@economica.local`
- Password: `admin123`

CASOS DE USO ESPECÍFICOS

EasyPanel

Método 1: Via Terminal Web (Más Fácil)

1. Ir a EasyPanel Dashboard
2. Seleccionar tu proyecto “muebleria-la-economica”
3. Click en “Terminal” o “Shell”
4. Ejecutar:

```
npx tsx --require dotenv/config scripts/seed.ts
```

Método 2: Via SSH al Servidor

```
# Conectar  
ssh root@tu-servidor-easypanel.com  
  
# Encontrar contenedor  
docker ps | grep muebleria  
  
# Ejecutar seed  
docker exec nombre_contenedor npx tsx --require dotenv/config scripts/seed.ts
```

Método 3: Usando el script automatizado

```
# Clonar el repo en el servidor (si no lo tienes)
git clone https://github.com/qhosting/muebleria-la-economica.git
cd muebleria-la-economica

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

◆ Coolify

Método 1: Via Interfaz Web (Recomendado)

1. Ir a Coolify Dashboard
2. Seleccionar tu aplicación
3. Click en “Execute Command” o “Run Command”
4. Ejecutar:

```
npx tsx --require dotenv/config scripts/seed.ts
```

Método 2: Via SSH

```
# Conectar al servidor
ssh root@tu-servidor-coolify.com

# Encontrar el contenedor de la app
docker ps | grep muebleria

# Usar el script
cd /ruta/a/muebleria_la_economica
./run-seed-docker.sh nombre_contenedor
```

◆ Docker Compose (Local o VPS)

```
# Opción 1: Script automatizado (detecta el contenedor)
./run-seed-docker.sh

# Opción 2: Docker Compose directo
docker-compose exec app npx tsx --require dotenv/config scripts/seed.ts

# Opción 3: Especificar contenedor
./run-seed-docker.sh muebleria_app_1
```

Kubernetes

```
# Encontrar el pod
kubectl get pods -n namespace | grep muebleria

# Ejecutar seed
kubectl exec -it nombre-pod -n namespace -- npx tsx --require dotenv/config scripts/seed.ts

# O usar el script con kubectl
kubectl exec -it nombre-pod -n namespace -- sh -c "npx tsx --require dotenv/config scripts/seed.ts"
```

EJEMPLOS REALES

Ejemplo 1: Primera vez en EasyPanel

```
# Paso 1: Verificar que la app está corriendo
curl https://app.mueblerialaeconomica.com/api/health

# Paso 2: SSH al servidor
ssh root@167.99.123.456

# Paso 3: Encontrar el contenedor
docker ps
# Output: abc123def456  easypanel/app-muebleria...

# Paso 4: Ejecutar seed
docker exec abc123def456 npx tsx --require dotenv/config scripts/seed.ts

# Paso 5: Verificar
curl -X POST https://app.mueblerialaeconomica.com/api/auth/signin \
-H "Content-Type: application/json" \
-d '{"email": "admin@economica.local", "password": "admin123"}'
```

Ejemplo 2: Usando script automatizado en Coolify

```
# Paso 1: Clonar repo en el servidor
cd /opt
git clone https://github.com/qhosting/muebleria-la-economica.git
cd muebleria-la-economica

# Paso 2: Ejecutar script (detecta automáticamente)
./run-seed-docker.sh

# Output esperado:
# 🚀 SEED EN CONTENEDOR DOCKER
# =====
# 📦 Buscando contenedores de la aplicación...
# ✓ Usando contenedor: coolify-app-muebleria-xyz
# 🔍 Verificando contenedor: coolify-app-muebleria-xyz
# ✓ Contenedor encontrado y en ejecución
# ...
# ✅ ¡Seed completado exitosamente!
```

Ejemplo 3: Reseed después de error en DB

```
# Problema: La base de datos se corrompió o tiene datos incorrectos

# Paso 1: Backup de seguridad (por las dudas)
docker exec postgres_container pg_dump -U usuario database_name > backup_$(date +%Y%m%d).sql

# Paso 2: Ejecutar migraciones (reset)
docker exec app_container npx prisma migrate reset --force

# Paso 3: Ejecutar seed
./run-seed-docker.sh app_container

# Paso 4: Verificar login
# Ir a la app y probar con admin@economica.local / admin123
```

VERIFICACIÓN POST-SEED

Verificación Rápida

```
# Test 1: Health check
curl https://app.mueblerialaeconomica.com/api/health

# Test 2: Login API
curl -X POST https://app.mueblerialaeconomica.com/api/auth/signin \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@economica.local","password":"admin123"}'

# Si retorna un token, ¡funciona!
```

Verificación en Base de Datos

```
# Conectar a la DB
docker exec -it postgres_container psql -U usuario -d database_name

# Verificar usuarios
SELECT email, name, role, "isActive" FROM "User";

# Output esperado:
```

email	name	role	isActive
admin@economica.local	Administrador Sistema	admin	t
gestor@economica.local	Gestor de Cobranza	gestor_cobranza	t
cobrador@economica.local	Cobrador	cobrador	t
reportes@economica.local	Usuario Reportes	reporte_cobranza	t

Verificación en la Aplicación

1. Abrir: <https://app.mueblerialaeconomica.com>
2. Ir a Login
3. Probar cada usuario:

Email	Password	Debería ver
admin@economica.local	admin123	Dashboard completo, Usuarios, Configuración
gestor@economica.local	gestor123	Dashboard, Rutas, Clientes, Pagos
cobrador@economica.local	cobrador123	Sus rutas, Cobros del día
reportes@economica.local	reportes123	Reportes, Estadísticas

⚠ PROBLEMAS COMUNES

✗ Error: “tsx: not found”

Causa: El script no está usando `npx`

Solución:

```
# Asegúrate de usar npx
npx tsx --require dotenv/config scripts/seed.ts

# NO uses:
tsx --require dotenv/config scripts/seed.ts # ✗ Falla en producción
```

✗ Error: “Cannot find module ‘@prisma/client’”

Causa: Prisma Client no está generado

Solución:

```
# Generar Prisma Client
docker exec contenedor npx prisma generate

# Luego ejecutar seed
./run-seed-docker.sh contenedor
```

✗ Error: “DATABASE_URL is not defined”

Causa: Variables de entorno no configuradas

Solución:

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

# Si no existe, agregar (reemplaza con tu URL real)
docker exec contenedor sh -c 'echo "DATABASE_URL=postgresql://..." >> .env'

# 0 reconstruir con variables correctas en EasyPanel/Coolify
```

✗ Error: “P1001: Can’t reach database server”

Causa: La DB no es accesible desde el contenedor

Solución:

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

# 2. Verificar red Docker
docker network ls
docker network inspect nombre_red | grep -A 5 "muebleria\|postgres"

# 3. Probar conectividad
docker exec app_container ping postgres_host

# 4. Verificar DATABASE_URL (debe usar el nombre de servicio correcto)
# Correcto: postgresql://user:pass@postgres:5432/db
# Incorrecto: postgresql://user:pass@localhost:5432/db
```

✗ Script no encuentra el contenedor

Solución:

```
# Ver TODOS los contenedores
docker ps -a

# Buscar específicamente
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}" | grep -i economica

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



PERSONALIZAR EL SEED

Cambiar Passwords

Edita `app/scripts/seed.ts` :

```
// Línea ~29
password: await bcrypt.hash('TU_PASSWORD_AQUI', 12),
```

Agregar Usuarios Adicionales

```
// Después del último usuario (línea ~65)
const miUsuario = await prisma.user.upsert({
  where: { email: 'mi-email@empresa.com' },
  update: {},
  create: {
    email: 'mi-email@empresa.com',
    name: 'Mi Nombre',
    password: await bcrypt.hash('mipassword', 12),
    role: 'admin', // o 'gestor_cobranza', 'cobrador', 'reporte_cobranza'
    isActive: true,
  },
});
```

No Eliminar Datos Existentes

Comenta las líneas de `deleteMany()` :

```
// Líneas 13-17 - Comentar o eliminar
// await prisma.pago.deleteMany();
// await prisma.rutaCobranza.deleteMany();
// await prisma.cliente.deleteMany();
// await prisma.plantillaTicket.deleteMany();
// await prisma.user.deleteMany();
```



DOCUMENTACIÓN ADICIONAL

- **Guía Completa:** [SEED-PRODUCTION-GUIDE.md](#) (./SEED-PRODUCTION-GUIDE.md)
- **Guía Rápida:** [README-SEED.md](#) (./README-SEED.md)
- **Resumen Técnico:** [SEED-SOLUTION-SUMMARY.md](#) (./SEED-SOLUTION-SUMMARY.md)



CHECKLIST DE DESPLIEGUE

Primera vez en Producción:

- [] 1. Verificar que la app está desplegada y corriendo
- [] 2. Verificar conectividad a la base de datos
- [] 3. Ejecutar migraciones: `npx prisma migrate deploy`
- [] 4. Generar Prisma Client: `npx prisma generate`
- [] 5. Ejecutar seed: `./run-seed-docker.sh`
- [] 6. Verificar health check: `curl .../api/health`
- [] 7. Probar login con `admin@economica.local` / `admin123`
- [] 8. Verificar permisos y roles en la interfaz
- [] 9. Cambiar passwords en producción (¡importante!)
- [] 10. Hacer backup de la base de datos

SOPORTE

Logs Útiles

```
# Logs de la aplicación
docker logs -f --tail=100 nombre_contenedor

# Logs de Prisma/Database
docker logs nombre_contenedor 2>&1 | grep -i prisma

# Logs en tiempo real durante seed
docker exec -it contenedor sh
npx tsx --require dotenv/config scripts/seed.ts 2>&1 | tee seed.log
```

Información del Sistema

```
# Versión de Node
docker exec contenedor node --version





# Versión de NPM/Yarn
docker exec contenedor npm --version

# Variables de entorno
docker exec contenedor env | grep -E "DATABASE|NODE_ENV"

# Estructura de archivos
docker exec contenedor ls -la /app
```

TODO LISTO

Si seguiste todos los pasos y el seed se ejecutó correctamente:

1.  Tienes 4 usuarios creados
2.  Puedes hacer login con cada uno
3.  Los permisos funcionan correctamente
4.  La aplicación está lista para usar

Próximos pasos:

- Cambiar los passwords por defecto
- Crear tus propios usuarios
- Agregar clientes y rutas
- ¡Comenzar a usar el sistema!

Última actualización: 30 de Septiembre, 2025

Versión de los Scripts: 1.0.0

Compatibilidad: EasyPanel  | Coolify  | Docker Compose  | Kubernetes 