

APRENDIZAJES-EXPRESS: ISS-04

# Seeders con Faker & Runner Orquestador

Generación de datos ficticios idempotentes por fuera del flujo HTTP

# Poblar la BD con datos realistas e idempotentes sin pasar por la capa HTTP

## 1. ¿Qué es?

Un mecanismo desacoplado del servidor Express para sembrar registros masivos en la BD usando la librería `@faker-js/faker`.

## 2. ¿Por qué existe?

Permite probar consultas, paginaciones y rendimiento con datos realistas (nombres, emails, passwords hasheados) sin insertarlos manualmente.

## 3. ¿Por qué fuera de HTTP?

Evita sobrecargar controladores/servicios y no depende de endpoints ni autenticación; interactúa directamente con los modelos de Sequelize.

## El Seeder Runner actúa independientemente del ciclo de vida de Express

### FLUJO TRADICIONAL HTTP (API)

Client Request → Routes → Controller → Service →  
Repository → Model → BD

- Requiere servidor corriendo (npm run dev)
- Pasa por validaciones HTTP y middlewares
- Lento para siembras masivas de datos

### FLUJO DIRECTO DE SEEDER (CLI)

npm run db:seed → SeedersRunner → clients.seeder.ts →  
ClientModel → BD

- Ejecución CLI en proceso aislado
- Conexión temporal a Sequelize y cierre automático
- Inserta directamente mediante arreglos y bulkCreate/insert

# Resolución de conteos en 3 niveles de prioridad: CLI → ENV → DEFAULT

| Prioridad / Nivel | Mecanismo de Configuración                                     | Ejemplo de Comando                 |
|-------------------|----------------------------------------------------------------|------------------------------------|
| 1. Máxima (CLI)   | Argumentos pasados directamente por consola (--clients=N)      | npm run db:seed -- --clients=20    |
| 2. Media (ENV)    | Variables de entorno configuradas en .env o CLI (SEED_CLIENTS) | SEED_CLIENTS=5 npm run db:seed     |
| 3. Base (Default) | Valores por defecto declarados en src/database/seeder/count.ts | npm run db:seed (Usa DEFAULT = 10) |

## Estructura del Seeder: Limpieza previa, Faker y Hasheo seguro

```
export const seedClients = async (count: number) => {
  // 1. Limpieza/Truncado (Idempotencia)
  await Client.destroy({ truncate: true, cascade:
    true });

  // 2. Hasheo previo de contraseña compartida
  const hashedPassword = await bcrypt.hash('Client123!',
    12);

  // 3. Bucle Faker
  for (let i = 0; i < count; i++) {
    await Client.create({
      first_name: faker.person.firstName(),
      last_name: faker.person.lastName(),
      email: faker.internet.email().toLowerCase(),
      password: hashedPassword,
      status: 'active'
    });
  }
};
```

### ASPECTOS CLAVE DEL CÓDIGO

- Idempotencia Garantizada: Borra los registros anteriores antes de sembrar nuevos.
- Optimización de Hash: Hashea la contraseña una sola vez fuera del bucle para mejorar la velocidad.
- Generación Realista: Usa `faker.person` y `faker.internet` para crear mails y nombres verosímiles.
- Respeto de Hooks: Al usar `Client.create` o `model.bulkCreate`, mantiene la integridad de los datos.

## El Runner ejecuta la secuencia completa de seeders y cierra la conexión

---

### 1. Conexión BD

Inicializa `sequelize.authenticate()` y valida credenciales.

### 2. Parseo Conteos

Llama a `resolveSeedCounts()` resolviendo argumentos CLI y variables de entorno.

### 3. Ejecución

Invoca `seedClients(counts.clients)` y los futuros seeders de las otras features.

### 4. Cierre Limpio

Imprime reporte en consola y ejecuta `process.exit(0)` o `sequelize.close()`.

## Criterios de Cierre GATE para aprobar la siembra de datos

### CHECKLIST DE VERIFICACIÓN (DEFINITION OF DONE)

- ✓ Script `npm run db:seed` configurado en `package.json` apuntando a `src/database/seeder/index.ts`.
- ✓ Compilador TypeScript en verde: `npx tsc --noEmit` finaliza sin errores.
- ✓ Idempotencia comprobada: Ejecutar el seeder repetidamente no genera duplicados ni rompe claves primarias.
- ✓ Flexibilidad de conteo: `npm run db:seed -- --clients=20` siembra exactamente 20 clientes en la BD.
- ✓ Verificación en BD: Consulta SQL `SELECT COUNT(*) FROM clients` confirma la cantidad sembrada.

**¡ISS-04 COMPLETADO CON ÉXITO!**

## **Siguiente Hito: ISS-05 — Swagger / OpenAPI**

Documentación interactiva OpenAPI 3.0 de la API Client en la etiqueta SIN AUTH.