

# **Manual de Usuario**

TDF HQ — CRM, Agendamiento, Paquetes e Invoicing (v1)

TDF Records / TDF Estudio

25 de septiembre de 2025

# Índice general

0.1. Introducción y alcance . . . . .	2
0.2. Requisitos del sistema . . . . .	2
0.3. Instalación rápida . . . . .	2
0.3.1. macOS (Homebrew) . . . . .	2
0.3.2. Ubuntu/Debian . . . . .	2
0.4. Configuración . . . . .	3
0.4.1. Variables de entorno . . . . .	3
0.4.2. Base de datos . . . . .	3
0.5. Compilación y ejecución . . . . .	3
0.6. Datos semilla (desarrollo) . . . . .	3
0.7. Uso por áreas . . . . .	4
0.7.1. CRM (Party/Personas) . . . . .	4
0.7.2. Agendamiento (Bookings) . . . . .	4
0.7.3. Paquetes de clases . . . . .	4
0.7.4. Facturación . . . . .	5
0.8. Referencia rápida de endpoints . . . . .	5
0.9. Roles y permisos recomendados (cuando se active Auth) . . . . .	5
0.10. Respaldo y restauración (DB) . . . . .	6
0.11. Solución de problemas . . . . .	6
0.12. Roadmap sugerido . . . . .	6
0.13. Glosario . . . . .	6
Apéndice A — Ejemplos listos para pegar . . . . .	7

## 0.1. Introducción y alcance

**TDF HQ** centraliza las operaciones del estudio y sello:

- **CRM:** artistas, clientes, profesores, ingenieros.
- **Agendamiento:** reservas básicas (sesiones, ensayos, clases).
- **Paquetes de clases:** productos y compras.
- **Facturación:** creación de facturas en borrador.

Esta versión (v1) expone un **backend API** en Haskell (Servant + Persistent + PostgreSQL) con migraciones automáticas y datos semilla para pruebas.

*No incluido aún:* autenticación/roles en API, detección de conflictos de recursos, deducción automática de créditos por asistencia, PDF de factura y SRI, Google Calendar. El modelo de datos ya lo contempla para siguientes versiones.

## 0.2. Requisitos del sistema

- PostgreSQL 14–16 (recomendado 16).
- Haskell Stack (Stack descarga GHC automáticamente).
- macOS o Linux (Windows funciona con WSL/instalador).
- Conexión a internet para la primera compilación.

## 0.3. Instalación rápida

### 0.3.1. macOS (Homebrew)

```
brew install haskell-stack postgresql@16
brew services start postgresql@16
echo 'export PATH="/opt/homebrew/opt/postgresql@16/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

### 0.3.2. Ubuntu/Debian

```
sudo apt update
sudo apt install haskell-stack postgresql postgresql-contrib
sudo service postgresql start
```

## 0.4. Configuración

### 0.4.1. Variables de entorno

Desde la carpeta del proyecto:

```
set -a; source config/default.env; set +a
```

Valores por defecto:

```
DB_HOST=127.0.0.1
DB_PORT=5432
DB_USER=postgres
DB_PASS=postgres
DB_NAME=tdf_hq
APP_PORT=8080
```

### 0.4.2. Base de datos

```
createdb tdf_hq || psql -d postgres -c "CREATE DATABASE tdf_hq;"
psql -d postgres -c "ALTER USER postgres WITH PASSWORD 'postgres';"
```

## 0.5. Compilación y ejecución

```
stack build
stack run
```

Salida esperada:

```
Running DB migrations...
Starting server on port 8080
```

Prueba de salud:

```
curl http://localhost:8080/health
# {"status":"ok","db":"ok"}
```

## 0.6. Datos semilla (desarrollo)

**Sólo** para entorno de desarrollo (crea artistas, profesores, salas, paquete Guitar 24h y equipo inicial):

```
curl -X POST http://localhost:8080/admin/seed
```

## 0.7. Uso por áreas

### 0.7.1. CRM (Party/Personas)

#### Crear persona

```
curl -X POST http://localhost:8080/parties \  
-H "Content-Type: application/json" \  
-d '{"cDisplayName":"Juano Ledesma","cIsOrg":false,  
    "cPrimaryEmail":"juano@tdf.com"}'
```

#### Listar personas

```
curl http://localhost:8080/parties
```

#### Consultar/Actualizar

```
curl http://localhost:8080/parties/1  
curl -X PUT http://localhost:8080/parties/1 \  
-H "Content-Type: application/json" \  
-d '{"uInstagram":"@juano"}'
```

#### Asignar rol (valores: Artist, Teacher, Engineer, Customer, etc.)

```
curl -X POST http://localhost:8080/parties/1/roles \  
-H "Content-Type: application/json" \  
-d '"Artist"'
```

### 0.7.2. Agendamiento (Bookings)

#### Crear reserva

```
curl -X POST http://localhost:8080/bookings \  
-H "Content-Type: application/json" \  
-d '{"cbTitle":"Rehearsal Arkabuz",  
    "cbStartsAt":"2025-10-01T19:00:00Z",  
    "cbEndsAt":"2025-10-01T21:00:00Z",  
    "cbStatus":"Confirmed"}'
```

#### Listar reservas

```
curl http://localhost:8080/bookings
```

### 0.7.3. Paquetes de clases

#### Ver productos

```
curl http://localhost:8080/packages/products
```

**Registrar compra**

```
curl -X POST http://localhost:8080/packages/purchases \
  -H "Content-Type: application/json" \
  -d '{"buyerId":1,"productId":1}'
```

**0.7.4. Facturación****Crear factura (borrador)**

```
curl -X POST http://localhost:8080/invoices \
  -H "Content-Type: application/json" \
  -d '{"ciCustomerId":1,"ciSubtotalCents":150000,
      "ciTaxCents":18000,"ciTotalCents":168000}'
```

**Listar facturas**

```
curl http://localhost:8080/invoices
```

**0.8. Referencia rápida de endpoints**

Endpoint	Descripción	Método
/health	Salud del servidor	GET
/parties	Lista/creación de personas (Party)	GET, POST
/parties/:id	Obtener/actualizar una persona	GET, PUT
/parties/:id/roles	Asignar un rol a la persona	POST
/bookings	Listar/crear reservas	GET, POST
/packages/products	Ver productos de paquete	GET
/packages/purchases	Registrar compra de paquete	POST
/invoices	Listar/crear facturas (borrador)	GET, POST
/admin/seed	Semilla de datos (dev)	POST

**0.9. Roles y permisos recomendados (cuando se active Auth)**

- **Admin/Manager:** acceso total.
- **Reception:** CRM básico, ventas de paquetes, reservas.
- **Teacher/Engineer:** ver agenda propia, actualizar asistencia (próximo).
- **Accounting:** facturas, pagos, notas de crédito, reportes.

## 0.10. Respaldo y restauración (DB)

### Backup

```
pg_dump -h 127.0.0.1 -U postgres -d tdf_hq -F c \
-f tdf_hq_$(date +%F).dump
```

### Restore

```
pg_restore -h 127.0.0.1 -U postgres -d tdf_hq \
--clean --create tdf_hq_YYYY-MM-DD.dump
```

## 0.11. Solución de problemas

- **stack: command not found:** instalar Stack y añadir `$HOME/.local/bin` al PATH.
- **createdb: command not found:** instalar Postgres y añadir bin al PATH.
- **Plan de Stack no resuelto (persistent-postgresql):** relajar versión en `.cabal` a `≥2.13` o añadir extra-deps sugeridos en `stack.yaml`.
- **Error libpq: invalid connection option "pool":** remover `pool=10` del connection string; el pool lo maneja Haskell.
- **Warning libpq.dylib built for newer macOS:** es sólo advertencia.

## 0.12. Roadmap sugerido

1. Autenticación Google OAuth + RBAC.
2. Conflictos de recursos (salas/profesores/equipo) + buffers.
3. Ledger de paquetes (descuento al marcar asistencia).
4. Integración Google Calendar (dos vías).
5. Facturas PDF con marca TDF + numeración (SRI).
6. Inventario: check-in/out con QR y mantenimiento.
7. Reportes: revenue por servicio, utilización, aging, expiración de paquetes.

## 0.13. Glosario

### Party

Persona/organización con roles (Artist, Teacher, Engineer, Customer, Vendor).

### Booking

Reserva en calendario (ensayo, sesión, clase).

**PackageProduct / Purchase**

Producto de horas/clases y su compra.

**Invoice**

Factura con totales (líneas/PDF en próximas versiones).

**Resource**

Recurso físico o humano (sala, profesor, ingeniero, equipo).

## Apéndice A — Ejemplos listos para pegar

**Crear cliente**

```
curl -X POST http://localhost:8080/parties \  
-H "Content-Type: application/json" \  
-d '{"cDisplayName":"El Bloque (Manager)","cIsOrg":true,  
    "cPrimaryEmail":"manager@elbloque.com"}'
```

**Asignar rol Customer**

```
curl -X POST http://localhost:8080/parties/2/roles \  
-H "Content-Type: application/json" \  
-d '"Customer"'
```

**Booking de grabación**

```
curl -X POST http://localhost:8080/bookings \  
-H "Content-Type: application/json" \  
-d '{"cbTitle":"Recording - Live Room",  
    "cbStartsAt":"2025-10-02T15:00:00Z",  
    "cbEndsAt":"2025-10-02T18:00:00Z",  
    "cbStatus":"Confirmed"}'
```

**Vender paquete Guitar 24h**

```
curl http://localhost:8080/packages/products  
curl -X POST http://localhost:8080/packages/purchases \  
-H "Content-Type: application/json" \  
-d '{"buyerId":1,"productId":1}'
```

**Crear factura**

```
curl -X POST http://localhost:8080/invoices \  
-H "Content-Type: application/json" \  
-d '{"ciCustomerId":1,"ciSubtotalCents":50000,  
    "ciTaxCents":6000,"ciTotalCents":56000}'
```