Dossier Técnico API RESTful - LazyLedger

César López, Alexis Jara, César Ramos

17/10/2025

1 Introducción

Propósito: Especificación técnica de la API RESTful de LazyLedger para integración y mantenimiento.

Objetivo: Desarrollar un MVP backend que permita gestionar y analizar operaciones financieras colaborativamente mediante interfaces conversacionales (chat) con respuestas contextualizadas.

Alcance Actual:

- Implementado: Autenticación JWT completa con registro y login
- Implementado: CRUD completo de Clientes con perfiles de usuario
- Implementado: Gestión de membresías en Ledgers colaborativos (invitaciones, roles, expulsiones)
- Implementado: Arquitectura limpia DDD con separación de capas
- Implementado: Principios RESTful, paginación y validaciones de negocio
- Pendiente: Módulos de Ledger, Transacciones, Deudas y Objetivos (estructurados)

Tecnologías: Spring Boot, Spring JPA, SpringDoc/Swagger, Docker, Clean Architecture.

2 Configuración General

URL Base: `http://localhost:8090/`
Formato: JSON exclusivamente
Autenticación: JWT Bearer Token (implementado)
Autorización: Basada en roles y membresía de ledger
Paginación: `page` (0-based), `size` (máx. 100)

2.1 Principios RESTful Implementados

Principio	Implementación en LazyLedger
Recursos por URIs	`/clientes`, `/miembros-ledger` - recursos claramente identificados
Métodos HTTP	GET (lectura), POST (crear), PUT (actualizar), DELETE (eliminar)
Stateless	No mantiene estado de sesión; cada request es independiente
Representaciones	JSON consistente para requests y responses
Códigos HTTP	200 (OK), 201 (Created), 204 (No Content), 400 (Bad Request), 403 (Forbidden), 404 (Not Found), 409 (Conflict)
HATEOAS	Implementación parcial con enlaces de navegación en responses

3 Modelos de Datos

3.1 Cliente

ClienteDTO (respuesta):

```
{
  "id": "01HXXXXXXXXXXXXXXXXXXXXX",
  "nombre": "Juan",
  "apellido": "Perez",
  "email": "juan.perez@example.com",
  "tipo": "PERSONA",
  "telefono": "+593987654321"
}
```

4 Endpoints Principales

4.1 Autenticación

Método	`Endpoint`	**Descripción**	**Estado**
POST	`/api/auth/signup`	Registro de nuevo usuario	Implementado
POST	`/api/auth/signin`	Autenticación de usuario	Implementado

4.2 Clientes

Método	`Endpoint`	**Descripción**	**Estado**
GET	`/api/clientes`	Lista paginada de clientes	Implementado
GET	`/api/clientes/{id}`	Cliente por ID	Implementado
GET	`/api/clientes/perfil`	Perfil del usuario autenticado	Pendiente
PUT	`/api/clientes/{id}`	Actualizar cliente	Implementado
PUT	`/api/clientes/perfil`	Actualizar perfil propio	Pendiente
DELETE	`/api/clientes/{id}`	Eliminar cliente	Implementado

Nota: Los clientes se crean automáticamente durante el registro de usuario en `/api/auth/signup`.

Para actualizar datos adicionales del perfil, usar `PUT /api/clientes/perfil`.

4.3 Ledgers

Método	`Endpoint`	**Descripción**	**Estado**
GET	`/api/ledgers`	Lista de mis ledgers	Pendiente
GET	`/api/ledgers/{id}`	Ledger específico	Pendiente
POST	`/api/ledgers`	Crear nuevo ledger	Pendiente
PUT	`/api/ledgers/{id}`	Actualizar ledger	Pendiente
PUT	`/api/ledgers/{id}/estado`	Cambiar estado del ledger	Pendiente
DELETE	`/api/ledgers/{id}`	Eliminar ledger	Pendiente

4.4 Transacciones

Método	Endpoint	Descripción	Estado
GET	/api/ledgers/{ledgerId}/transacciones	Lista de transacciones	Pendiente
GET	/api/ledgers/{ledgerId}/transacciones/{id}	Transacción específica	Pendiente
POST	/api/ledgers/{ledgerId}/transacciones	Crear transacción	Pendiente
PUT	/api/ledgers/{ledgerId}/transacciones/{id}	Actualizar transacción	Pendiente
DELETE	/api/ledgers/{ledgerId}/transacciones/{id}	Eliminar transacción	Pendiente
GET	/api/ledgers/{ledgerId}/transacciones/resumen	Resumen por período	Pendiente

4.5 Deudas

Método	Endpoint	Descripción	Estado
GET	/api/ledgers/{ledgerId}/deudas	Lista de deudas	Pendiente
GET	/api/ledgers/{ledgerId}/deudas/{id}	Deuda específica	Pendiente
POST	/api/ledgers/{ledgerId}/deudas	Crear deuda	Pendiente
PUT	/api/ledgers/{ledgerId}/deudas/{id}	Actualizar deuda	Pendiente
DELETE	/api/ledgers/{ledgerId}/deudas/{id}	Eliminar deuda	Pendiente
POST	/api/ledgers/{ledgerId}/deudas/{id}/pagos	Registrar pago	Pendiente
GET	/api/ledgers/{ledgerId}/deudas/pendientes	Deudas pendientes	Pendiente

4.6 Objetivos

Método	Endpoint	Descripción	Estado
GET	/api/ledgers/{ledgerId}/objetivos	Lista de objetivos	Pendiente
GET	/api/ledgers/{ledgerId}/objetivos/{id}	Objetivo específico	Pendiente
POST	/api/ledgers/{ledgerId}/objetivos	Crear objetivo	Pendiente
PUT	/api/ledgers/{ledgerId}/objetivos/{id}	Actualizar objetivo	Pendiente
DELETE	/api/ledgers/{ledgerId}/objetivos/{id}	Eliminar objetivo	Pendiente
POST	/api/ledgers/{ledgerId}/objetivos/{id}/progreso	Actualizar progreso	Pendiente

4.7 Miembros de Ledger

Método	Endpoint	Descripción	Estado
GET	/api/ledgers/{ledgerId}/miembros	Miembros del ledger	Implementado
POST	/api/ledgers/{ledgerId}/miembros/invitar	Invitar miembro	Implementado
PUT	/api/ledgers/{ledgerId}/miembros/{id}/rol	Cambiar rol	Implementado
DELETE	/api/ledgers/{ledgerId}/miembros/{id}	Expulsar miembro	Implementado
POST	/api/ledgers/{ledgerId}/miembros/{id}/aceptar	Aceptar invitación	Pendiente
POST	/api/ledgers/{ledgerId}/miembros/{id}/rechazar	Rechazar invitación	Pendiente

4.8 Reportes y Dashboard

Método	Endpoint	Descripción	Estado
GET	/api/dashboard/resumen	Resumen general	Pendiente
GET	/api/dashboard/ledgers	Estado de ledgers	Pendiente
GET	/api/dashboard/alertas	Alertas importantes	Pendiente
GET	/api/ledgers/{ledgerId}/reportes/balance	Balance general	Pendiente
GET	/api/ledgers/{ledgerId}/reportes/transacciones	Reporte transacciones	Pendiente
GET	/api/ledgers/{ledgerId}/reportes/deudas	Reporte deudas	Pendiente
GET	/api/ledgers/{ledgerId}/reportes/objetivos	Progreso objetivos	Pendiente
GET	/api/ledgers/{ledgerId}/reportes/mensual	Reporte mensual	Pendiente

Ejemplo POST `/api/ledgers/{ledgerId}/miembros/invitar`**:**

```
{
  "clienteId": "01HXXXXXXXXXXXXXXXXXXXXX"
}
```

Requiere autenticación JWT

5 Manejo de Errores

Respuestas de Error Estandarizadas:

```
{
  "timestamp": "2025-10-15T19:30:09.2826473",
```

```
  "message": "Datos invalidos",
  "type": "VALIDATION_ERROR"
}
```

Códigos HTTP Comunes:

- `400 Bad Request`: Datos inválidos o faltantes
- `403 Forbidden`: Sin permisos para la operación
- `404 Not Found`: Recurso no encontrado
- `409 Conflict`: Conflicto (ej: email duplicado, ya es miembro)

6 Arquitectura e Implementación

Clean Architecture:

- **Presentación**: Controllers REST, DTOs, mapeo de requests/responses
- **Aplicación**: Use Cases, Facades, coordinación de operaciones
- **Dominio**: Entities, Services, reglas de negocio, excepciones
- **Infraestructura**: Repositories JPA, configuración Spring

Características Técnicas:

- UUID v7 para IDs únicos y ordenados temporalmente
- Validaciones de negocio en dominio (independientes del framework)
- Manejo centralizado de excepciones con GlobalExceptionHandler
- Paginación automática con metadatos
- Documentación OpenAPI automática
- Headers de autenticación preparados (no implementados en MVP)

7 Testing y Despliegue

Ejecutar la Aplicación:

```
mvn spring-boot:run
# Acceder en: http://localhost:8090
```

Docker:

```
docker-compose up -d
```

Testing Básico:

```
# 1. Registrar usuario
curl -X POST http://localhost:8090/api/auth/signup \
  -H "Content-Type: application/json" \
  -d '{"nombreUsuario":"testuser","contrasena":"password123","
     confirmarContrasena":"password123","email":"test@example.com"}'

# 2. Login (obtener token JWT)
curl -X POST http://localhost:8090/api/auth/signin \
  -H "Content-Type: application/json" \
  -d '{"nombreUsuario":"testuser","contrasena":"password123"}'

# 3. Crear cliente (con token JWT)
curl -X POST http://localhost:8090/api/clientes \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -d '{"nombre":"Test","apellido":"User","email":"test@example.com","tipo
     ":"PERSONA","telefono":"123456789"}'

# 4. Listar clientes
curl -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  http://localhost:8090/api/clientes
```

8 Uso de la API

Documentación Interactiva: http://localhost:8090/swagger-ui.html

Uso de Swagger

Figura 1: `/api/auth/signup`

Figura 2: `/api/auth/signin`

9 Conclusión

Resumen: API RESTful completa con Clean Architecture, CRUD de clientes, gestión de membresías en ledgers colaborativos, validaciones de negocio y documentación automática.

Declaración de Uso de Inteligencia Artificial

A continuación se presentan los prompts utilizados para la generación y maquetación de este documento.

Prompt 1: Actualización de Contenido del Dossier

Los endpoints de autenticación pueden quedarse con verbos porque es convención (como signup y signin), pero todos los demás quiero que usen sustantivos. Por ejemplo, en lugar de /clientes/crear"sería /clientes", en lugar de /transacciones/agregar"sería /transacciones", etc.

También hay algo importante que corregir: en el documento dice que se pueden crear clientes manualmente con POST /clientes, pero eso está mal. Los clientes NO se crean manualmente, se crean automáticamente cuando alguien se registra como usuario. Así que quita ese endpoint de crear cliente y explica en el documento que los clientes se crean automáticamente durante el registro.

Y otra cosa: en lugar de usar símbolos pon palabras normales como Ïmplementadoz "Pendiente"para que se vea más profesional. Solo actualiza el documento técnico, nada más. *Oye, necesito que me ayudes a actualizar este documento técnico que tengo, el Mini-Dossier-Tecnico-API.md. Quiero que cambies todos los endpoints que tienen verbos por sustantivos, pero déjame explicarte bien:*

Los endpoints de autenticación pueden quedarse con verbos porque es convención (como signup y signin), pero todos los demás quiero que usen sustantivos. Por ejemplo, en lugar de /clientes/crear"sería /clientes", en lugar de /transacciones/agregar"sería /transacciones", etc.

También hay algo importante que corregir: en el documento dice que se pueden crear clientes manualmente con POST /clientes, pero eso está mal. Los clientes NO se crean manualmente, se

crean automáticamente cuando alguien se registra como usuario. Así que quita ese endpoint de crear cliente y explica en el documento que los clientes se crean automáticamente durante el registro.

Y otra cosa: en lugar de usar símbolos pon palabras normales como Ïmplementadoz "Pendiente"para que se vea más profesional. Solo actualiza el documento técnico, nada más.

Prompt 2: Conversión de Markdown a LaTeX

Toma el contenido de mi archivo 'Mini-Dossier-Tecnico-API.md' y conviértelo a un documento LaTeX completo y profesional. Utiliza la clase 'article', configura los márgenes y el idioma español. Asegúrate de que los bloques de código JSON y Bash se muestren correctamente usando el paquete 'listings', que las tablas se conviertan de forma adecuada y que la estructura de encabezados y listas se mantenga. El código debe ser limpio y estar listo para compilar. º