Manual Técnico – Airtickets (Sistema de Gestión de Boletos Aéreos)

Integrantes:

Javier Stanley Valladares Valladares VV230315 Líder de Proyecto – Desarrollador Backend – Documentador

Marvin Rene Martinez Gomez MG231425 Encargado de Base de Datos – Documentador

Jonathan Josue Cardoza Perez CP230528 Desarrollador Frontend – Documentador

Proyecto: Airtickets – Sistema de Gestión de Boletos Aéreos

Docente: Mario Alvarado

Asignatura: Desarrollo de Aplicaciones con Web Frameworks

Backend: Java 17 + Spring Boot · Seguridad: JWT · Persistencia: Spring Data JPA · BD: MySQL

Link de Github y Notion

https://github.com/marvin0martinez/Proyecto_catedra_airtickets-.git

https://www.notion.so/25e278d9a20f800dbf57ce5417a876bf?v=25e278d9a20f8015b933000cfa9aee90&source=copy_link

Índice

1. Introducción

Este manual técnico documenta la arquitectura, la configuración y el funcionamiento interno de Airtickets, una API REST para administración de usuarios, aerolíneas, vuelos, reservaciones, pagos y reclamos. Sirve como guía de instalación, despliegue, mantenimiento y defensa del proyecto final (DWF404).

2. Objetivos

• Entregar una API REST funcional al 100% con CRUD por módulos principales y documentación OpenAPI/Swagger.

• Implementar autenticación y autorización con JWT y control de acceso por roles (p. ej., ADMIN, USER).

• Garantizar persistencia relacional en MySQL con un modelo normalizado y eficiente.

• Establecer una estructura de paquetes clara para mantenimiento futuro y buenas prácticas.

3. Arquitectura y Estructura del Proyecto

Patrón de arquitectura REST con capas: Controller (exposición HTTP), Service (lógica de negocio), Repository (acceso a datos JPA), Entity (modelo de dominio), DTO y Security (filtro y configuración JWT).

Estructura detectada en src/main/java:

• sv/edu

4. Dependencias y Versiones (pom.xml)

Artifact: sv.edu.udb.airtickets-single-app:1.0.0

Dependencias:

- org.springframework.boot : spring-boot-starter-web

- org.springframework.boot : spring-boot-starter-validation

- org.springframework.boot : spring-boot-starter-security

- org.springframework.boot : spring-boot-starter-data-jpa

- com.mysql : mysql-connector-j (scope=runtime)

- org.springdoc : springdoc-openapi-starter-webmvc-ui : ${springdoc.version}

- io.jsonwebtoken : jjwt-api : ${jjwt.version}

- io.jsonwebtoken : jjwt-impl : ${jjwt.version} (scope=runtime)

- io.jsonwebtoken : jjwt-jackson : ${jjwt.version} (scope=runtime)

- org.projectlombok : lombok : ${lombok.version} (scope=provided)

- org.springframework.boot : spring-boot-starter-test (scope=test)

5. Seguridad (Spring Security + JWT)

Archivos con referencias a JWT/Bearer:

• proyecto_final/src/main/java/sv/edu/udb/airtickets/api/AuthController.java

• proyecto_final/src/main/java/sv/edu/udb/airtickets/config/SecurityConfig.java

• proyecto_final/src/main/java/sv/edu/udb/airtickets/security/JwtAuthFilter.java

• proyecto_final/src/main/java/sv/edu/udb/airtickets/security/JwtService.java

El flujo típico: login → emisión de token JWT → envío de Authorization: Bearer <token> → filtros de autorización validan y permiten acceso a endpoints protegidos según rol.

6. Endpoints Principales (descubiertos en controladores)

• AdminAirlineController

GET /api/admin/airlines/{id}

GET /api/admin/airlines/{id}

GET /api/admin/airlines/{id}

• AdminAirportController

GET /api/admin/airports/{id}

GET /api/admin/airports/{id}

GET /api/admin/airports/{id}

• AdminFlightController

GET /api/admin/flights/{id}

GET /api/admin/flights/{id}

GET /api/admin/flights/{id}

• AdminPassengerController

GET /api/admin/passengers/{id}

GET /api/admin/passengers/{id}

GET /api/admin/passengers/{id}

• AdminReservationCrudController

GET /api/admin/reservations/{id}

GET /api/admin/reservations/{id}

- AdminStatsController

  GET /api/admin/stats/summary

- AuthController

  GET /api/auth/login

- ClaimController

  GET /api/claims/me

  GET /api/claims/{id}/status

- FlightController

  GET /api/flights/search

- HealthController

  GET /api/health

- ReservationController

  GET /api/passengers

  GET /api/payments

  GET /api/reservations

  GET /api/reservations/locator/{code}

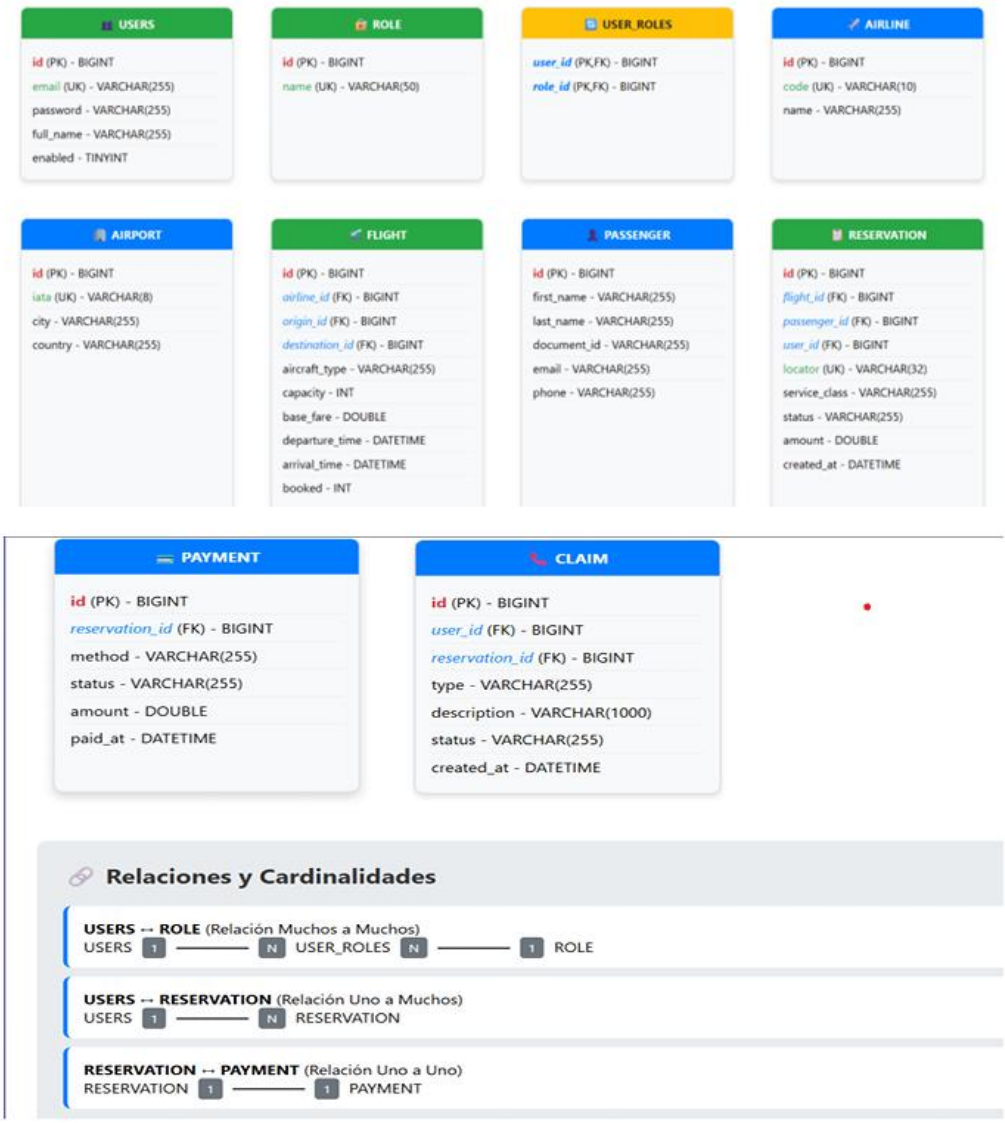
  GET /api/reservations/{id}

  GET /api/reservations/{id}/cancel

7. Base de Datos (Diagrama E-R y Tablas Clave)

Modelo relacional con entidades para países/ciudades/aeropuertos, aerolíneas, vuelos y rutas; proceso de compra con reservaciones y pagos; reclamos; clases de servicio y tripulación.

Diagrama E-R (extraído del documento técnico):



El diagrama presentado representa la estructura relacional de la base de datos diseñada en MySQL Server para gestionar los procesos de vuelos, reservaciones y servicios asociados a la aerolínea.

Tabla: Reclamos

Campo	Tipo de Dato	Tamaño	Descripción	Restricciones
ReclamoID	INT	-	Identificador único del reclamo	PK, Not Null
NumeroReclamo	NVARCHAR	15	Número de reclamo (auto generado)	Único
PasajeroID	INT	-	Pasajero que realiza el reclamo	FK → Pasajeros
ReservacionID	INT	-	Reservación asociada (opcional)	FK → Reservaciones
TipoReclamoID	INT	-	Tipo de reclamo	FK → TiposReclamo
Descripcion	NVARCHAR	1000	Detalle del reclamo	Not Null
EstadoReclamo	NVARCHAR	20	Abierto, En Proceso, Resuelto, etc.	Default='Abierto'

d) Características adicionales

Triggers:

- Actualizan los asientos vendidos de cada tarifa cuando se inserta, actualiza o elimina una reservación.
- Generan automáticamente el código de reservación.
- Generan números de reclamos consecutivos.

Índices:

Se crearon en campos clave como fechas de vuelos, estado de reservaciones, estado de pagos y documentos de pasajeros, optimizando así las consultas más frecuentes.

Con esta estructura, la base de datos garantiza un manejo eficiente de vuelos, pasajeros y transacciones, manteniendo la consistencia y confiabilidad de la información.

Diagrama Entidad Relación

8. Configuración del Entorno

Requisitos: Java 17, Maven 3.8+, MySQL 8, IntelliJ IDEA/STS, Postman o Swagger UI.

Pasos:

1) Crear BD: CREATE DATABASE airtickets CHARACTER SET utf8mb4;

2) Configurar credenciales en src/main/resources/application.properties (URL, usuario, contraseña).

3) Compilar/ejecutar: mvn clean package && mvn spring-boot:run

4) Acceder a Swagger: http://localhost:8080/swagger-ui/index.html

5) OpenAPI JSON: http://localhost:8080/v3/api-docs

9. Pruebas, Usuarios y Datos de Ejemplo

Credenciales de prueba:

• admin@air.tix / 123456 (ADMIN)

• user@air.tix / 123456 (USER)

Endpoints típicos de prueba: autenticación, CRUD de aerolíneas, vuelos, reservaciones, pagos y reclamos.

Para endpoints protegidos, enviar encabezado Authorization: Bearer <token>.

10. Despliegue

Despliegue local con Maven/Spring Boot. Para producción: empaquetar JAR y ejecutar en servidor Linux con Java 17; configurar variables de entorno para credenciales y URL de BD; preparar backups y logs rotativos.

11. Repositorio y Referencias

Repositorio GitHub: https://github.com/Valladaresst/DWF_Proyecto_de_catedra.git

Swagger UI: http://localhost:8080/swagger-ui/index.html

OpenAPI JSON: http://localhost:8080/v3/api-docs

12. Formato del Documento (conforme a rúbrica)

• Papel carta, márgenes 2 cm, interlineado 1.15, fuente Calibri 10, texto justificado.

• Secciones mínimas incluídas: introducción, objetivos, procesos/módulos, herramientas, diagrama relacional, estructura de paquetes, dependencias y versiones, URL del repositorio, usuarios predefinidos.