Proyecto Final: Sistema de Gestión de Vuelos (SGV)

Asignatura: Programación 3

1. Descripción General

El objetivo es desarrollar un sistema integral para la gestión de reservas de vuelos multi-aerolínea. La aplicación debe demostrar un dominio avanzado de la programación orientada a objetos, asegurando la separación de responsabilidades mediante una arquitectura de capas, persistencia eficiente en archivos y un esquema de validación mediante pruebas automatizadas.

2. Requisitos Funcionales (Entidades y CRUD)

El sistema deberá gestionar, de manera obligatoria, las siguientes entidades mediante operaciones de **Lectura, Creación, Actualización y Eliminación (CRUD)**:

- **Clientes:** Registro y gestión de pasajeros.
- **Vuelos:** Programación de rutas y horarios.
- **Aviones:** Inventario de flota y capacidades.
- **Aerolíneas:** Administración de las empresas prestadoras del servicio.
- **Reservas:** Módulo central que vincula las entidades anteriores para la venta de boletos.

---

3. Arquitectura y Estándares de Programación

Se evaluará rigurosamente el uso de buenas prácticas y el aprovechamiento de las características avanzadas del lenguaje:

Estructura de Capas (N-Tiers)

1. **Capa de Presentación (UI):**
    - Desarrollada preferiblemente en **Java SWING**.
    - Debe implementar **Controladores** para mediar la comunicación con la capa de servicio.
    - **Interfaz:** Una ventana principal centrada en la "Gestión de Reservas" y un sistema de menús jerárquicos para acceder a los CRUDs de las demás entidades.
2. **Capa de Servicio (Lógica de Negocio):**
    - Orquestar los casos de uso del sistema.
    - Validar las reglas de negocio antes de interactuar con la persistencia.
3. **Capa de Persistencia (Datos):**
    - Encargada exclusivamente de la lectura y escritura de información.

Calidad de Código y Uso del Lenguaje

- **Tipos Genéricos (Generics):** Uso de clases e interfaces genéricas, especialmente en la capa de persistencia (ej. un DAO genérico) para evitar la duplicidad de código y garantizar la seguridad de tipos.
- **Gestión de Excepciones:** Implementación de un sistema de excepciones personalizadas para capturar y gestionar errores específicos de negocio y de acceso a archivos.
- **Colecciones:** Uso óptimo del API de *Collections* para la manipulación de datos en memoria antes de su persistencia.

---

4. Persistencia y Almacenamiento Técnico

El manejo de archivos debe seguir una estrategia de alto rendimiento:

- **Acceso Aleatorio:** Uso de RandomAccessFile con registros de **longitud fija**.
- **Índices Densos:** Implementación de un mapa de índices (ID vs. Posición) para permitir búsquedas directas.
- **Serialización:** El almacenamiento y recuperación de los índices puede realizarse mediante objetos serializables.
- **Optimización:** Queda prohibido el uso de acceso secuencial para operaciones de búsqueda o actualización.

---

5. Plan de Pruebas (Calidad de Software)

Se debe entregar un plan de pruebas unitarias que valide la estabilidad del sistema:

A. Pruebas Unitarias y de Integración

- **JUnit 5:** Automatización de casos de prueba.
- **Mockito:** Uso de *mocks* para aislar la capa de servicio de la capa de persistencia durante las pruebas.

B. Pruebas de Estrés y Rendimiento (Performance Testing)

Para validar la eficiencia de la arquitectura de archivos e índices, el estudiante debe:

- **Script de Poblamiento Masivo:** Desarrollar un módulo o script que genere e inserte automáticamente **miles de registros aleatorios** en las entidades.
- **Análisis de Tiempos:** Realizar mediciones de tiempo de acceso (búsqueda, lectura y escritura) con grandes volúmenes de datos para demostrar la eficiencia del acceso aleatorio frente al secuencial.

---

6. Criterios de Evaluación

Criterio	Descripción
Arquitectura	Separación estricta de capas y flujo de datos coherente.
Uso del Lenguaje	Aplicación correcta de Genéricos, Excepciones y Colecciones.
Persistencia	Implementación de archivos de acceso aleatorio con índices densos.
Testing	Cobertura de pruebas unitarias y uso correcto de Mocks.
Interfaz	Usabilidad de la GUI en Swing y manejo de eventos.