Informe Detallado del Sistema de Gestión de Contenidos en Kotlin:

Descripción General:

Este proyecto desarrolla un **sistema de gestión de contenidos** en **Kotlin** que permite:

- **Cargar** películas y series desde un archivo **Excel** (.xlsx).
- **Consultar** el catálogo de contenidos.
- **Realizar un análisis predictivo** básico sobre los datos.
- **Sugerir** recomendaciones para nuevos contenidos basados en análisis.

Estructura del Proyecto:

El sistema está compuesto principalmente por:

- **Clase Contenido** → Representa una película o serie.
- **Clase Catalogo** → Maneja la colección de contenidos y operaciones.
- **Función leerOpcion()** → Valida entradas del usuario.
- **Función main()** → Administra el flujo principal del programa.

También se utilizan librerías externas para manejar archivos Excel: **Apache POI** (org.apache.poi).

Desglose de Código:

Clase Contenido:

```
class Contenido(
    private val id: String,
```

```
    private var titulo: String,
    private var tipo: String,
    private var rating: Double,
    private var duracion: Int,
    private var genero: String,
    private var año: Int
)
```

Descripción:

Define un **modelo de datos** que encapsula toda la información relevante de un contenido.

Atributos:

Atributo	Tipo	Significado
id	String	Identificador único.
titulo	String	Nombre del contenido.
tipo	String	"Serie" o "Película".
rating	Double	Valoración promedio.
duracion	Int	Duración en minutos.
genero	String	Género principal.
año	Int	Año de lanzamiento.

Métodos:

```
fun getId(): String
fun getTitulo(): String
```

```
fun getTipo(): String
fun getRating(): Double
fun getDuracion(): Int
fun getGenero(): String
fun getAño(): Int
```

- **Getters personalizados** para cada atributo (encapsulamiento correcto).

```
fun mostrarInfo()
```

- **Imprime** los datos del contenido de forma alineada y legible.
- Utiliza padEnd y format para alinear los valores.

Clase Catalogo:

```
class Catalogo {
    private val contenidos = mutableListOf<Contenido>()
}
```

Descripción:

Gestiona la **colección de contenidos** y proporciona operaciones de **agregado**, **consulta**, **carga** y **análisis**.

Métodos:

agregarContenido(contenido: Contenido)

```
fun agregarContenido(contenido: Contenido)
```

- **Agrega** un contenido nuevo **solo si no existe** otro con el mismo ID.
- **Prevención de duplicados** usando .any {}.

consultarContenidos()

fun consultarContenidos()

- **Lista** todos los contenidos almacenados.
- Muestra un mensaje especial si la lista está vacía.

cargarDesdeExcel(filePath: String)

fun cargarDesdeExcel(filePath: String)

- **Carga masiva** de contenidos desde un archivo .xlsx.
- **Pasos principales:**
    - Abre el archivo con FileInputStream.
    - Usa XSSFWorkbook para leer el Excel.
    - Itera sobre las filas (saltando la fila de encabezados).
    - Crea objetos Contenido desde las celdas de la fila.
    - Llama a agregarContenido() para añadirlos.
- **Manejo de Errores:**
    - Captura excepciones al abrir el archivo o leer celdas inválidas.
    - Mensajes de error por fila específica si algo falla.

📈 analizarContenidos()

fun analizarContenidos()

- **Analiza** los contenidos existentes para generar **estadísticas** y **predicciones**.
- **Qué calcula:**
    1. Número total de contenidos, series y películas.
    2. Promedio de **ratings** de series y películas.
    3. **Top 3 géneros** con mejor promedio de rating.
    4. **Recomendación predictiva**:
        - Tipo ideal (Serie/Película).
        - Género con mejor aceptación.
        - Duración promedio sugerida.
        - Rating promedio esperado.

Función leerOpcion()

```
fun leerOpcion(): Int
```

- Lee de forma segura un número entero ingresado por el usuario.
- Si el input no es numérico, **vuelve a pedir** hasta que sea válido.
- Previene fallos por input inválido (texto, vacío, etc.).

Función Principal main()

```
fun main()
```

- **Menú interactivo** que permite al usuario:

    - Consultar contenidos.
    - Cargar nuevos datos desde un archivo.
    - Ejecutar el análisis predictivo.
    - Salir del programa.
- Usa un do-while para mantener el programa ejecutándose hasta que el usuario elija salir (opción 0).

Librerías Utilizadas:

Librería	Propósito
org.apache.poi.ss.usermodel.*	Manejar elementos generales de Excel.
org.apache.poi.xssf.usermodel.XSSFWorkbook	Trabajar específicamente con archivos .xlsx.

Manejo de Errores:

Situación	Manejo
Archivo no encontrado	Try-Catch al abrir FileInputStream.
Error en fila de Excel	Captura e imprime error indicando número de fila.
ID duplicado	Se evita con comprobación antes de agregar.
Input inválido del usuario	Validación en leerOpcion().

Mejoras Futuras:

- Persistencia en base de datos (SQLite, PostgreSQL).
- Interfaz gráfica (GUI) para gestionar mejor los contenidos.
- Exportación del catálogo actualizado a Excel o CSV.
- Implementar Machine Learning para análisis predictivo real.

Conclusión:

Este proyecto muestra cómo diseñar un sistema de gestión robusto usando **Kotlin**, aplicando buenas prácticas como:

- **Encapsulamiento**.
- **Validación de datos**.
- **Manejo de errores**.
- **Separación de responsabilidades**.

También introduce el manejo de **archivos Excel** en Kotlin, útil en escenarios empresariales donde se trabaja con datos externos.