**Documento Diseño Fase 2  
Grupo 2**

**Contexto y alcance:**

Se esta buscando diseñar un sistema integral que sea capaz de gestionar interacciones y ventas de boletería de un parque de diversiones con el uso de un catálogo de atracciones y espectáculos.

Los 3 principales objetivos que va a tener nuestro sistema son:

* Tendrá un catalogo de atracciones y espectáculos que se pueda presentar a los usuarios (Clientes, empleados y administrador).
* Gestión de empleados con relación a sus turnos, los roles y las capacitaciones que requiere cada atracción y que tiene cada empleado.
* Debe ser capaz de vender tiquetes de manera de forma virtual y en la taquilla (Debe haber una variedad de modalidades y opciones de tiquetes).

Los principales Stakeholders del contexto son:

* Administradores (gestionan el parque y la plataforma).
* Empleados (Consultan los turnos, venden tiquetes a los clientes y operan las atracciones teniendo en cuenta los turnos consultados).
* Clientes (Compran tiquetes y consultan el catálogo de Atracciones/Espectáculos)

Requerimientos del código:

* Lenguaje Java.
* Toda la información debe ser persistente.
* La información debe almacenarse en archivos (Planos/Binarios).
* Solo la aplicación va a leer los archivos de la carpeta donde se guarde la información.
* La carpeta no puede ser la misma carpeta donde se encuentre el código fuente de la aplicación.
* La persistencia no necesariamente debe hacerse en un solo archivo: diseñe con cuidado cuántos archivos habrá y cómo van a estar estructurados.
* Todos los usuarios del sistema deben tener un login y una contraseña para poder acceder al sistema.
* Se pueden añadir funcionalidades extra que faciliten el trabajo (UML e implementación).

**Alcance**

**Incluye:**

Atracciones y espectáculos:

* Registrar y editar atracciones o espectáculos con sus respectivas características y restricciones de uso.
* Definir para cada atracción que restricciones tiene y que modalidad de tiquete permite su uso.
* Disponibilidad según época del año, clima y mantenimiento.

Empleados:

* Asignación de roles (Cocina, atracciones, limpieza, etc).
* Asignación de turnos (Mañana/tarde).
* Gestión de capacitaciones para cada atracción según el riesgo.
* Validación de requisitos para que se asigne un trabajador a cada atracción o lugar de servicio.

Tiquetes:

* Venta en línea y en la taquilla (con descuento para empleados).
* Tipo de tiquete (Generales: Básico, familiar, oro, diamante/Temporada: semanal, mensual, anual/Individual atracción/FastPass).
* Generación de un código QR para hacer uso de las atracciones.

Persistencia:

* Archivos estructurados para almacenar la información (Json, Dat, etc)
* Carpeta con información fuera del código fuente.

**No incluye:**

* Interfaz gráfica.

**Objetivos:**

La aplicación debe cumplir con ciertos requisitos que ayuden a cumplir con lo requerido por el parque de diversiones. Para ello se debe poder Clasificar las atracciones por tipo y por exclusividad y ser presentadas en el catálogo, se debe poder validar restricciones de acceso para cada atracción (no todas deben tener una restricción necesariamente), la aplicación debe poder asignar empleados a atracciones según disponibilidad de turno, capacitaciones y riesgo, Debe poder calcular los precios de cada tiquete para clientes normales y precios con descuentos para empleados, debe poder generar códigos QR que permitan el acceso a los servicios del parque y marcar los códigos como usados una vez estos ya hayan sido leídos. Adicionalmente, el código debe tener persistencia en sus archivos, se le debe asignar un login y una contraseña a cada usuario (debe cumplir con credenciales seguras) y el sistema debe ser capaz de manejar una cantidad x de usuarios e información sin problema.

**No Objetivos:**

En esta fase no se quiere realizar una interfaz gráfica, por el momento solo se quiere validar el funcionamiento del código fuente y del proceso de diseño por medio de diagramas UML. La creación de códigos QR también es hipotética por el momento, pues sin interfaz no se puede crear un modelo real de código QR que sea funcional. Los procesos de pago no son reales, son simulaciones.

**Diagramas modelo:**

* Diagrama de secuencia.
* Diagrama UML de clases.
* Diagrama de casos de uso.
* Diagrama de estados.

**APIs**

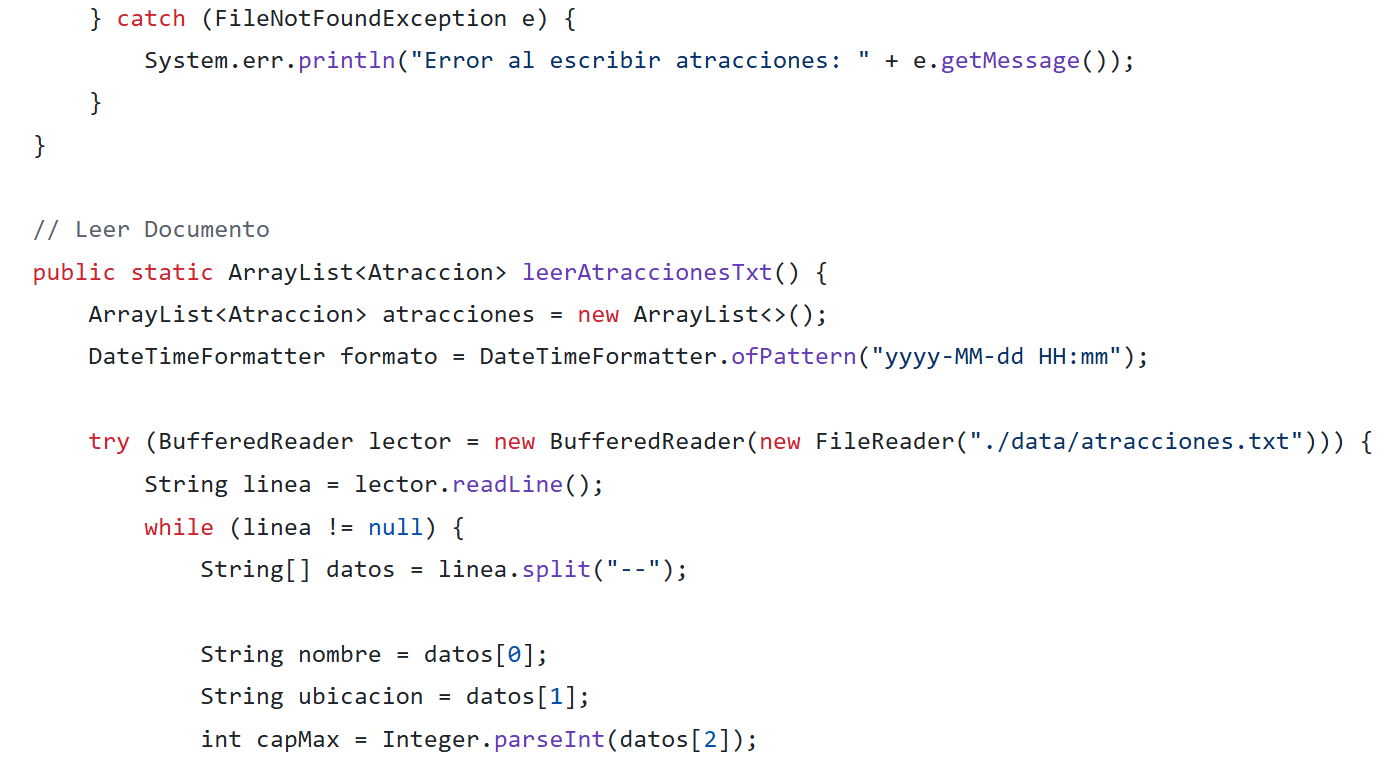
Para realizar la persistencia en el proyecto empleamos la técnica vista en clase de Lector/Escritor de archivos, la cual se trata de una implementación personalizada sencilla considerada como API interna. No se utilizaron APIs externas ya que, al ser un proyecto pequeño, las necesidades de este no las requerían, por lo que optamos por las APIs básicas de Java.io.

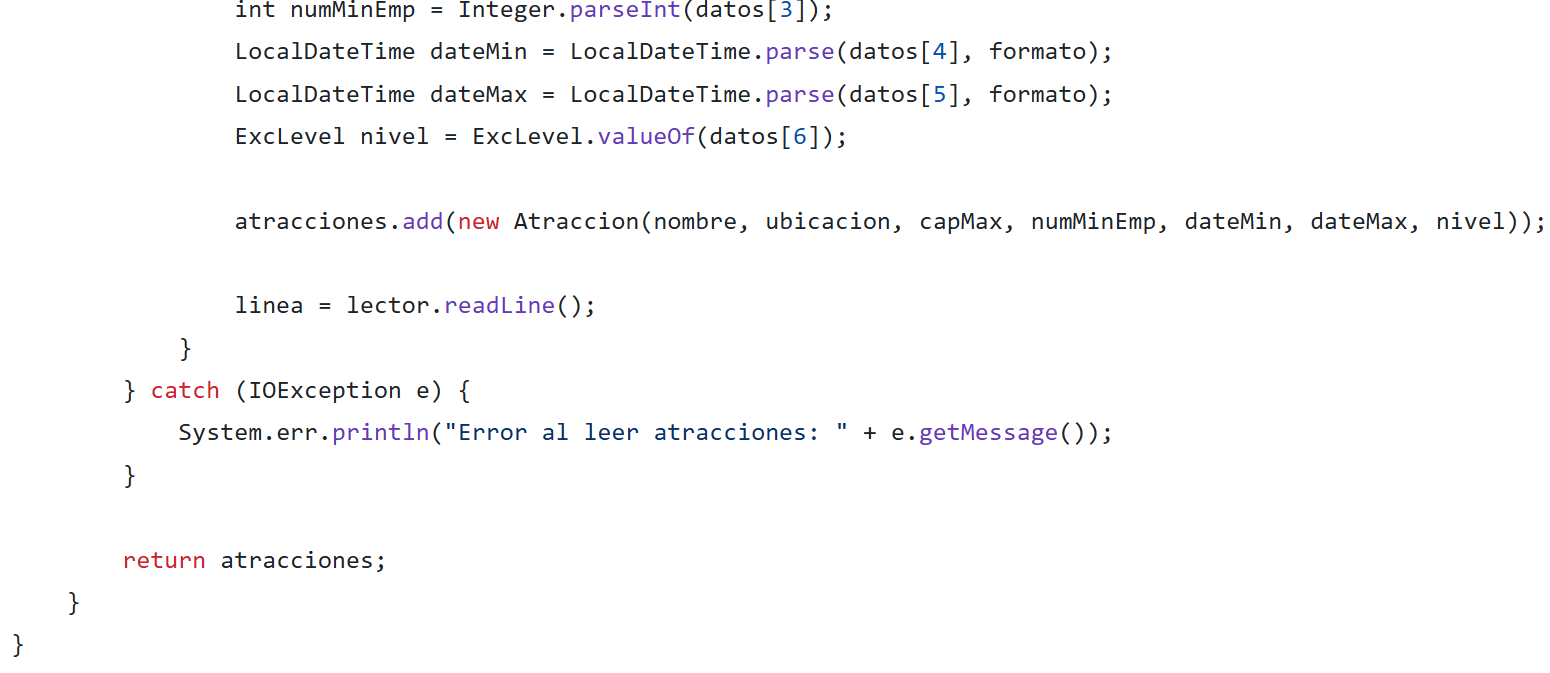
La capa de persistencia que se utilizó contiene:

* BufferedReader.
* FileReader.
* PrintWriter.
* Serialización tipo CSV.
* Conceptos previamente vistos en clase.

**Código algoritmos críticos**

****

****

****

Esta parte del código muestra la estructura que se utilizo para implementar la persistencia por medio del Lector y Escritor de archivos.

**Alternativas consideradas durante el proceso del trabajo**

1. Para realizar la persistencia, consideramos usar la librería externa json.jar ya que nos facilitaba el desarrollo del trabajo al ofrecernos herramientas ya implementadas para almacenar archivos, sin embargo, tuvimos problemas al momento de instalarla en eclipse.
2. Durante el proceso de diseño, tuvimos problemas decidiendo a que clases les aplicaríamos la persistencia.