Universidad de los Andes

FACULTAD DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PROYECTO 1

Diseño y Programación Orientada a Objetos

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN PARA UN PARQUE DE ATRACCIONES

Jerónimo López

Daniel Felipe Diab G

Juan Esteban Piñeros

12 de abril de 2025

Diseño e Implementación de un Sistema de Gestión para un Parque de Atracciones

***Resumen*—Este documento presenta el diseño e implementación de una aplicación orientada a objetos que permite la gestión integral de un parque de atracciones. El sistema contempla la administración de atracciones mecánicas y culturales, la gestión de espectáculos temporales, el manejo de empleados en distintos roles y turnos, así como la venta y validación de tiquetes con distintos niveles de acceso. El desarrollo fue realizado en Java y se enfatiza la separación de responsabilidades entre clases, la persistencia de datos mediante archivos externos y el cumplimiento de reglas específicas del negocio. Además, se diseñaron consolas de prueba que permiten demostrar la correcta implementación de las funcionalidades clave. Este documento contiene los diagramas de clases del sistema, descripciones detalladas de las decisiones de diseño y una visión estructurada de la arquitectura general, sentando las bases para futuras extensiones del proyecto.**

I. Introducción

Este documento presenta el diseño detallado e implementación del sistema de gestión para un parque de atracciones, desarrollado como parte del Proyecto 1 del curso de Diseño y Programación Orientada a Objetos (DPOO). El objetivo central del sistema es permitir la administración eficiente de los recursos y operaciones del parque, brindando soporte para la gestión de atracciones, espectáculos, empleados y la venta de tiquetes, de manera coherente con las reglas de negocio definidas.

En esta segunda entrega del proyecto, se trasciende la etapa de análisis inicial para abordar aspectos concretos de diseño y codificación. Se adopta un enfoque basado en programación orientada a objetos, donde las entidades del dominio han sido representadas mediante clases que encapsulan tanto datos como comportamientos, siguiendo principios de modularidad y responsabilidad única.

El sistema fue implementado en Java, asegurando la persistencia de la información mediante el uso de archivos externos. Asimismo, se construyeron diferentes consolas de prueba que permiten demostrar el funcionamiento de funcionalidades específicas, como:

- La creación y validación de atracciones, considerando restricciones como altura, peso, edad o condiciones de salud.
- La asignación de turnos y tareas a empleados, de acuerdo con su rol y nivel de capacitación.
- La compra y uso de tiquetes, incluyendo tiquetes básicos, de temporada, entradas individuales y servicios adicionales como el FastPass.

Este documento incluye los siguientes elementos clave:

- Diagramas de clases detallados y de alto nivel que describen la arquitectura del sistema.
- Justificación de las decisiones de diseño adoptadas durante el desarrollo.
- Descripción de los mecanismos de persistencia utilizados y estructura de los archivos.
- Explicación de los programas de prueba implementados para validar la lógica del sistema.

A través de esta documentación, se proporciona una visión clara y estructurada del sistema desarrollado, sentando una base sólida para futuras etapas del proyecto, como la integración de interfaces de usuario e interacción completa con los distintos tipos de usuario.

II. Funcionalidades

El sistema desarrollado para la gestión del parque de atracciones implementa un conjunto amplio de funcionalidades orientadas a modelar de forma realista y operativa los procesos que ocurren dentro del parque. Estas funcionalidades se organizaron considerando los diferentes tipos de usuarios, recursos físicos y roles del sistema. A continuación, se describen las principales capacidades implementadas:

1. Gestión de Atracciones y Espectáculos

- Creación de atracciones mecánicas y culturales con restricciones específicas (edad, altura, peso, salud).
- Evaluación de disponibilidad de una atracción en función de la temporada y el clima.
- Definición de espectáculos con horarios y fechas determinadas, incluyendo los de temporada.
- Asociación de cada atracción o espectáculo a una ubicación dentro del parque.

2. Administración de Empleados y Turnos

- Creación de distintos tipos de empleados: cajeros, cocineros, operadores mecánicos y personal de servicio general.
- Asignación de turnos a empleados, con información de fecha, tipo de jornada (mañana o tarde) y lugar de trabajo.
- Validación de funciones por tipo de empleado (por ejemplo, solo los cocineros pueden preparar alimentos).
- Capacitación de operadores mecánicos para determinadas atracciones y validación de su autorización.

3. Manejo de Usuarios

- Registro de usuarios con credenciales de acceso diferenciadas por rol: administrador, empleado o cliente.
- Acceso restringido a funcionalidades según el tipo de usuario.

4. Venta y Validación de Tiquetes

- Creación y compra de diferentes tipos de tiquetes:
  - Tiquete Básico
  - Tiquete Familiar
  - Tiquete Oro
  - Tiquete Diamante
  - Tiquete de Temporada
  - Entrada Individual
- Validación del acceso a atracciones según la categoría del tiquete.
- Simulación del uso de FastPass para evitar filas en atracciones seleccionadas.
- Registro del uso de tiquetes y verificación de intentos de reutilización.

5. Gestión de Ventas

- Registro de ventas presenciales en taquillas operadas por cajeros.
- Simulación de ventas online a través de la clase `VentaOnline`, incluyendo método de pago, fecha y total.

6. Consolas de Prueba

- Menú interactivo que permite probar funcionalidades como compra, uso de tiquetes, validaciones de acceso, asignación de turnos, etc.
- Visualización por consola del estado del sistema y resultados de cada operación.

III. Diseño del Sistema

El diseño del sistema se fundamenta en los principios de programación orientada a objetos, dividiendo las responsabilidades en distintos paquetes según el dominio funcional: usuarios y empleados, atracciones y espectáculos, y tiquetes. A continuación, se presentan los diagramas de clases, descripciones de las entidades principales y las decisiones clave de diseño adoptadas durante el desarrollo.

III-A. Diagrama de Clases de Diseño

El sistema está compuesto por múltiples clases distribuidas en tres paquetes principales:

- **Paquete Persona:** Contiene todas las clases relacionadas con los usuarios del sistema, como clientes, empleados y el administrador.
- **Paquete Atracciones:** Agrupa las clases que modelan atracciones mecánicas, culturales, espectáculos, ubicación y temporadas.
- **Paquete Tiquetes:** Incluye las clases relacionadas con la compra, gestión y validación de los diferentes tipos de tiquetes.

El diagrama de clases completo incluye relaciones de herencia, composición y asociación entre las clases. A continuación, se presenta una descripción de las clases más importantes.

Clases principales del paquete `persona`:

- **Usuario (abstracta):** Clase base que define las credenciales comunes para todos los tipos de usuarios.
- **Administrador:** Hereda de `Usuario`. Permite asignar turnos, modificar empleados y gestionar la operación general del parque.
- **Empleado (abstracta):** Define las propiedades y métodos comunes a todos los empleados del parque.
- **Turno:** Representa un turno de trabajo, incluyendo la fecha, tipo de turno (mañana/tarde), y lugar de trabajo.
- **Subclases de Empleado:** `Cajero`, `Cocinero`, `ServicioGeneral` y `OperadorMecanico`, cada una con capacidades específicas según su rol.

Clases principales del paquete `atracciones`:

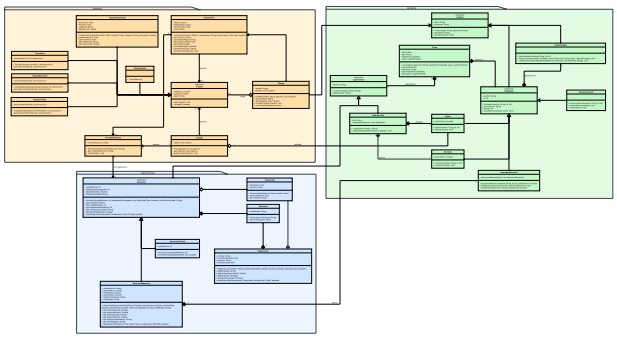
- **Atraccion (abstracta):** Clase base para modelar las atracciones del parque, con atributos como cupo, empleados requeridos, y nivel de exclusividad.
- **AtraccionCultural:** Subclase que agrega restricción por edad mínima.
- **AtraccionMecanica:** Subclase que incorpora restricciones por salud, altura, peso y riesgo.
- **Espectaculo:** Clase que representa eventos del parque, definidos por nombre, fechas y horarios.
- **Ubicacion:** Indica en qué zona del parque se encuentra una atracción o lugar.
- **Temporada:** Define un rango de fechas para disponibilidad especial de atracciones o espectáculos.

Clases principales del paquete `tiquetes`:

- **Cliente:** Usuario que adquiere y utiliza tiquetes para acceder a las atracciones.
- **Tiquete (abstracta):** Clase base con atributos como uso, fastpass y tipo de acceso.
- **Subtipos de Tiquete:** `TiqueteBasico`, `Familiar`, `Oro`, `Diamante`, `Temporada` y `EntradaIndividual`.
- **VentaOnline:** Gestiona la compra en línea de tiquetes, procesamiento de pagos y generación de facturas.
- **Taquilla:** Punto físico de venta, asociado a cajeros.

III-B. Diagrama de Clases de Alto Nivel

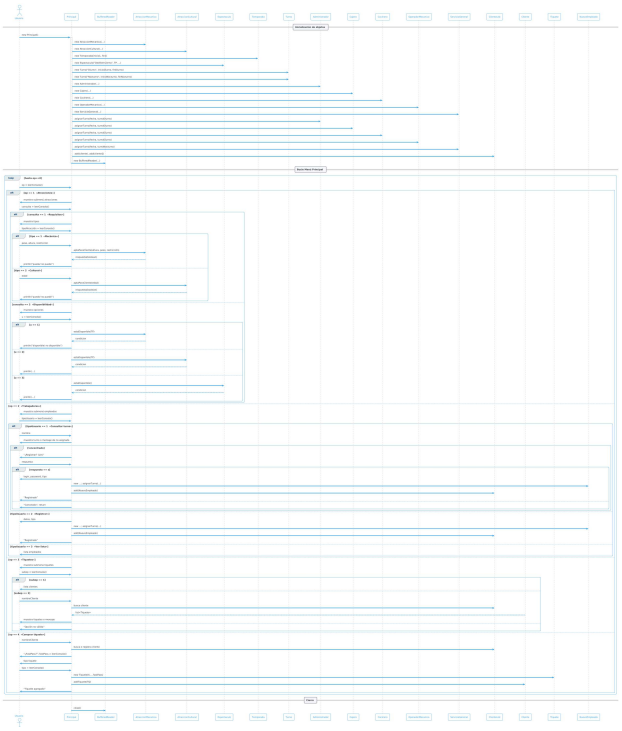
El siguiente diagrama resume las principales relaciones entre los paquetes y las entidades clave del sistema, omitiendo detalles internos (atributos y métodos) para facilitar la comprensión estructural general.



El diagrama completo con todos los métodos y atributos se encuentra en el anexo de este documento.

III-C. Diagrama de secuencia

El diagrama de secuencia representa la interacción entre el objeto cliente y el sistema de compra de tiquetes. El sistema pregunta al cliente que tipo de tiquete desea y recrea la simulacion de esta compra actualizando el estado del cliente y asignandole el tiquete que adquirio. La secuencia ilustra claramente el flujo ordenado de mensajes entre las entidades involucradas en el proceso.



El diagrama completo con todos los detalles se encuentra en el anexo de este documento.

III-D. Justificación del Diseño

La organización del sistema en paquetes separados facilita la modularidad, el mantenimiento y la extensibilidad. Se utilizaron clases abstractas para generalizar comportamientos comunes (como `Usuario`, `Empleado` y `Tiquete`) y se aplicaron principios SOLID para la separación de responsabilidades y la reutilización de código. Además:

- Las reglas de negocio específicas (como validaciones de acceso, restricciones de atracciones o asignación de turnos) se implementan en las clases correspondientes, evitando lógica duplicada.
- La herencia permite manejar tipos de atracciones y tiquetes de manera polimórfica.
- La persistencia de datos se maneja desde clases centrales que escriben y leen archivos de texto de manera estructurada.

III-E. Paquete `atracciones`

El paquete `atracciones` contiene las clases responsables de modelar las atracciones y espectáculos disponibles en el parque. Estas clases encapsulan las propiedades físicas y lógicas de las atracciones, así como sus restricciones de acceso, disponibilidad y ubicación.

Clase `Atraccion` (abstracta): Esta clase es la base para todos los tipos de atracciones en el parque. Define atributos comunes como:

- `cupoMaximo`: número máximo de visitantes permitidos.
- `empleadosRequeridos`: cantidad mínima de empleados necesarios para operar.
- `exclusividad`: nivel de acceso requerido (Familiar, Oro o Diamante).
- `climaRestringido`: indica si la atracción puede cerrarse por condiciones climáticas.
- `temporada`: objeto de tipo `Temporada` que define su periodo de disponibilidad.

Sus métodos permiten:

- Verificar disponibilidad por clima o temporada.
- Obtener el nivel de exclusividad.
- Consultar si requiere empleados asignados.

Clase `AtraccionCultural`: Subclase de `Atraccion` que representa atracciones tipo museo, teatro o casas del terror. Agrega:

- `edadMinima`: edad mínima para ingresar.

Incluye un método que evalúa si un cliente cumple la restricción de edad.

Clase `AtraccionMecanica`: Subclase de `Atraccion` que modela atracciones físicas como montañas rusas o carruseles. Introduce:

- `alturaMinima`, `alturaMaxima`: rango de altura permitida.
- `pesoMinimo`, `pesoMaximo`: rango de peso permitido.
- `restriccionesSalud`: lista de condiciones médicas no aptas (ej. vértigo, cardiopatías).
- `nivelRiesgo`: indica si el riesgo es medio o alto.

Incluye métodos para verificar si un cliente puede acceder a la atracción en función de sus condiciones físicas y de salud.

Clase `Espectaculo`: Clase independiente de `Atraccion`. Modela eventos especiales que pueden ocurrir en cualquier parte del parque. Contiene:

- `nombre`: identificador del espectáculo.
- `fechasDisponibles`, `horarios`: programación del evento.
- `esDeTemporada`: indica si el evento solo ocurre durante un periodo específico.

Métodos clave permiten validar disponibilidad para un día específico.

Clase `Ubicacion`: Clase auxiliar que representa una zona física dentro del parque. Define:

- `zona`: nombre o código de la zona.

Permite asociar atracciones con su localización dentro del parque.

Clase `Temporada`: Clase que modela un rango de tiempo en el que una atracción o espectáculo está disponible. Atributos:

- `fechaInicio`, `fechaFin`: delimitan la temporada.

Sus métodos permiten consultar si una fecha dada cae dentro del rango válido.

Resumen: El paquete `atracciones` centraliza la lógica de validación de acceso y disponibilidad de las atracciones del parque, estableciendo una clara diferenciación entre atracciones culturales, mecánicas y espectáculos, permitiendo su extensión y validación mediante principios de herencia y polimorfismo.

III-F. Paquete `tiquetes`

El paquete `tiquetes` contiene las clases que modelan el proceso de compra, gestión y validación de los diferentes tipos de tiquetes disponibles en el parque. Estas clases permiten controlar el acceso a las atracciones, validar condiciones especiales de uso, y manejar las transacciones tanto físicas como virtuales.

Clase `Tiquete` (abstracta): Clase base para todos los tipos de tiquetes del parque. Contiene atributos generales como:

- `utilizado`: indica si el tiquete ya fue usado.
- `tieneFastPass`: determina si el tiquete incluye acceso preferencial.
- `tipo`: categoría del tiquete (básico, oro, etc.).

Incluye métodos para:

- Consultar y modificar el estado del tiquete.
- Validar si permite el acceso a determinada atracción.

Clases derivadas de `Tiquete`:

- **`TiqueteBasico`**: permite el ingreso al parque, pero no a ninguna atracción.
- **`TiqueteFamiliar`**: permite acceso a atracciones de nivel `Familiar`.
- **`TiqueteOro`**: permite acceso a atracciones `Familiar` y `Oro`.
- **`TiqueteDiamante`**: acceso total a todas las atracciones del parque.
- **`TiqueteTemporada`**: acceso ilimitado por un rango de fechas definido (`fechaInicio` y `fechaFin`), con una categoría de acceso incluida y descuento aplicado.
- **`EntradaIndividual`**: da acceso una sola vez a una atracción específica, sin importar la categoría general del cliente.

Clase `Cliente`: Usuario del parque que puede adquirir uno o varios tiquetes. Contiene:

- `nombre`: identificador del cliente.
- `tiquetes`: lista de tiquetes adquiridos.

Métodos principales:

- Comprar un tiquete (según tipo).
- Consultar los tiquetes disponibles.
- Usar un tiquete y validar su acceso.

Clase `VentaOnline`: Modela una transacción de compra de tiquetes a través de internet. Atributos clave:

- `cliente`: cliente que realiza la compra.
- `metodoPago`: forma de pago utilizada.
- `fechaCompra`: momento de la compra.
- `total`: valor de la transacción.

Incluye métodos para procesar el pago, generar facturas y enviar confirmaciones.

Clase `Taquilla`: Representa una taquilla física en el parque. Contiene:

- `cajeros`: lista de empleados asignados a la taquilla.

Sus métodos permiten registrar ventas presenciales de tiquetes y asignar cajeros para la atención al público.

Resumen: El paquete `tiquetes` encapsula todas las operaciones relacionadas con la adquisición y validación de tiquetes. Gracias al uso de herencia y especialización, se manejan fácilmente múltiples tipos de entrada, cada una con reglas específicas de acceso, vigencia y beneficios. También se contempla tanto la venta física como digital, lo cual permite al sistema adaptarse a diferentes canales de atención al cliente.

III-G. Paquete `persona`

El paquete `persona` contiene las clases relacionadas con los distintos tipos de usuarios del sistema: clientes, empleados y administradores. Este paquete también incluye las clases necesarias para modelar los turnos y lugares de trabajo dentro del parque. Se hace uso de herencia y composición para garantizar la reutilización de atributos y comportamientos comunes.

Clase `Usuario` (abstracta): Clase base para todos los tipos de usuario del sistema. Define atributos comunes como:

- `login`: nombre de usuario.
- `password`: contraseña de acceso.

Provee métodos para acceder y modificar las credenciales, y sirve como superclase para `Cliente`, `Empleado` y `Administrador`.

Clase `Administrador`: Usuario con permisos especiales para gestionar todo el sistema. Hereda de `Usuario`. Sus principales responsabilidades incluyen:

- Asignar turnos a empleados.
- Modificar la información de atracciones, empleados y espectáculos.
- Consultar información sensible del sistema.

Clase `Empleado` (abstracta): Clase base para los diferentes tipos de trabajadores del parque. Define atributos como:

- `nombre`
- `idEmpleado`
- `turnos`: lista de turnos asignados.

Sus métodos permiten consultar turnos por fecha y verificar disponibilidad. Las subclases de `Empleado` representan roles específicos con restricciones únicas.

Subclases de `Empleado`:

- **`Cajero`**: Encargado de las ventas en taquillas. Tiene métodos para registrar ventas y validar transacciones.
- **`Cocinero`**: Encargado de la preparación de alimentos. Posee un atributo para verificar si está capacitado en cocina.
- **`ServicioGeneral`**: Se encarga de labores de limpieza y mantenimiento en el parque.
- **`OperadorMecanico`**: Operador de atracciones mecánicas. Tiene una lista de atracciones para las que está certificado, y métodos para verificar su autorización en una atracción específica.

Clase `Turno`: Representa un turno laboral. Contiene:

- `fechaTurno`
- `tipoTurno`: apertura o cierre.
- `lugarTrabajo`: instancia de un lugar donde se asigna el empleado.

Permite consultar la información asociada al turno y verificar conflictos de horario.

Clase `LugarTrabajo` (abstracta): Clase abstracta que representa un lugar físico dentro del parque donde puede asignarse personal. Define:

- `nombreLugar`

Clase `LugarServicio`: Subclase de `LugarTrabajo` que representa un lugar de atención al público, como una taquilla, tienda o cafetería. Permite gestionar la asignación de empleados por turno.

Resumen: El paquete `persona` encapsula todos los elementos relacionados con los usuarios del sistema y el manejo del recurso humano. Su diseño facilita la gestión de múltiples tipos de empleados con roles claramente definidos, respetando las restricciones operativas del parque mediante validaciones específicas por tipo de cargo.

IV. Persistencia de Datos

El sistema cuenta con un paquete de persistencia de datos plenamente funcional que utiliza archivos planos en formato CSV. Esta implementación asegura que información clave del sistema como los empleados, turnos, clientes y tiquetes se conserve entre sesiones al almacenar su estado en archivos externos y restaurarlo posteriormente, asegurando la continuidad operativa mejorando el funcionamiento y aumentando notablemente la experiencia del usuario.

Implementación de la persistencia

Para gestionar esta funcionalidad, se desarrolló la clase ArchivoPlano ubicada en el paquete persistencia, esta está encargada de agrupar los métodos necesarios para leer y escribir archivos de texto. Esta clase ofrece dos funciones principales:

- leer(String nombreArchivo): lee línea por línea el contenido de un archivo CSV y lo devuelve como una lista de cadenas.
- escribir(String nombreArchivo, ArrayList<String> lineasTexto): permite guardar los cambios realizados durante la ejecución, sobrescribiendo el archivo con la información más actual.

Código en Java usado para leer datos:

```java
package persistencia;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class ArchivoPlano {
    public ArrayList<String> leer (String nombreArchivo){
        ArrayList<String> lineasTexto = new ArrayList<String>();
        try {
            BufferedReader br = new BufferedReader(new FileReader(nombreArchivo));
            String linea;
            while((linea = br.readLine()) != null) {
                lineasTexto.add(linea);
            }
            br.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return lineasTexto;
    }
}
```

Carga automática de datos al iniciar el sistema

Cada vez que el sistema arranca, carga automáticamente los archivos ubicados en la carpeta datos/. Estos archivos están organizados por tipo de entidad. Por ejemplo:

- empleados.csv: contiene datos como el tipo de empleado, credenciales de acceso, nombre, ID, departamento y el turno asignado.
- clientes.csv: almacena las credenciales de los clientes y los tiquetes que poseen, separados por punto y coma.

El sistema transforma esta información en objetos específicos, como instancias de Administrador, Cajero, Cliente, TiqueteOro, entre otros. Para lograr esto, se emplean estructuras de control como switch, que permiten identificar el tipo de entidad a crear, junto con métodos de apoyo que asignan turnos, atracciones o listas de tiquetes según corresponda.

Los archivos estarán organizados de forma modular:

- `clientes.csv`: con los datos de cada cliente.
- `empleados.csv`: lista de empleados y sus turnos.

Justificación del enfoque

Se eligió trabajar con archivos planos por su simplicidad, portabilidad y bajo costo de implementación, cualidades que se ajustan bien a la etapa actual del proyecto. Esta estrategia

permite mantener un diseño modular, ya que cada tipo de entidad cuenta con su propio archivo independiente, lo cual facilita tanto la edición como el mantenimiento de los datos.

El formato CSV resulta especialmente práctico: es claro, fácil de leer, se puede modificar con cualquier editor de texto y, al mismo tiempo, mantiene una estructura ordenada y predecible.

Además, este enfoque no limita el crecimiento del sistema. En el futuro, si el volumen de datos o la cantidad de usuarios crece, será posible migrar a una base de datos más robusta sin necesidad de rehacer todo el sistema debido al diseño modular que separa los datos de la lógica del programa.

Resumen: Con esta implementación, el sistema ahora puede guardar y cargar datos de forma estructurada mediante archivos CSV. Esto asegura que los cambios hechos durante el uso se mantengan en ejecuciones futuras, lo que mejora la confiabilidad y facilidad de uso del programa. Además, esta base sólida deja abierta la puerta para escalar hacia soluciones más avanzadas sin perder la claridad y sencillez necesarias en las primeras fases del desarrollo.

V. Programas de Prueba

Para validar el correcto funcionamiento del sistema, se desarrollaron múltiples programas de prueba ejecutables por consola. Estas consolas permiten demostrar que las funcionalidades clave del parque de atracciones han sido implementadas de forma correcta, sin necesidad de una interfaz gráfica.

Cada consola se enfoca en una funcionalidad específica del sistema y está diseñada para mostrar de forma clara el estado interno de los objetos, las validaciones realizadas y los resultados de las acciones ejecutadas.

Consola Principal

La clase `Principal` contiene un menú principal que permite navegar entre las distintas funcionalidades del sistema, incluyendo:

- Cargar la información desde archivos.
- Consultar atracciones disponibles.
- Consultar empleados y sus turnos asignados.
- Validar si un cliente puede acceder a una atracción específica.
- Simular la compra y uso de tiquetes por parte de un cliente.

Pruebas de Atracciones

Las pruebas asociadas al manejo de atracciones permiten:

- Verificar la creación correcta de atracciones culturales y mecánicas.
- Consultar la disponibilidad de una atracción según la temporada y el clima.
- Comprobar si un cliente cumple las condiciones físicas y de salud para acceder a una atracción mecánica.
- Validar restricciones por edad mínima en atracciones culturales.

Pruebas de Empleados y Turnos

El sistema permite ejecutar pruebas relacionadas con la gestión de empleados:

- Crear distintos tipos de empleados (cajero, cocinero, operador, mecánico, etc.).
- Asignar turnos a empleados, indicando el lugar y el tipo de turno.
- Consultar los turnos asignados a un empleado en una fecha dada.
- Verificar que un empleado solo pueda asumir roles para los que esté capacitado (por ejemplo, un cajero no puede cocinar).
- Validar que un operador mecánico esté autorizado para una atracción de riesgo alto específica.

Pruebas de Tiquetes

Las pruebas relacionadas con los tiquetes incluyen:

- Compra de distintos tipos de tiquetes (básico, familiar, oro, diamante, temporada e individuales).
- Verificación del uso correcto de un tiquete (marcado como utilizado).
- Validación del acceso a atracciones según el tipo de tiquete y su nivel de exclusividad.
- Simulación del uso de FastPass para evitar filas en atracciones seleccionadas.
- Detección de intentos de reutilización de tiquetes ya usados.

Pruebas de Persistencia

El sistema carga y guarda la información mediante archivos de texto, por lo cual se implementaron pruebas que permiten:

- Leer los datos iniciales desde los archivos de persistencia.
- Verificar que los cambios realizados en la ejecución (nuevos turnos, tiquetes comprados, etc.) sean almacenados correctamente.
- Volver a cargar los datos en una nueva ejecución para comprobar la persistencia.

Resumen: Los programas de prueba desarrollados permiten validar de forma precisa todas las funcionalidades del sistema, asegurando que las reglas de negocio se cumplan correctamente. Estas consolas sientan la base para futuras pruebas automatizadas o interfaces de usuario en próximas entregas del proyecto.

VI. Conclusiones y Trabajo Futuro

El desarrollo de este proyecto ha permitido aplicar de manera práctica los conceptos fundamentales de la programación orientada a objetos, incluyendo la abstracción, encapsulamiento, herencia y polimorfismo. A través del diseño e implementación de un sistema completo para la gestión de un parque de atracciones, se logró construir una arquitectura modular, extensible y coherente con los requerimientos planteados.

Logros alcanzados

- Se diseñó e implementó una estructura de clases robusta, dividida en paquetes lógicos según las responsabilidades del sistema.
- Se garantizó la persistencia de datos mediante archivos externos, con formatos estructurados y mecanismos de lectura y escritura confiables.
- Se desarrollaron consolas de prueba interactivas que permiten validar todas las funcionalidades críticas del sistema, incluyendo la gestión de atracciones, empleados y tiquetes.
- Se aplicaron principios de buenas prácticas de diseño, como la separación de responsabilidades, el uso de clases abstractas y el modelado de relaciones adecuadas entre entidades.

Trabajo futuro

Para las siguientes fases del proyecto, se plantean las siguientes líneas de trabajo:

- Implementación de una interfaz de usuario por consola, que permita a cada tipo de usuario (cliente, empleado, administrador) interactuar directamente con el sistema.
- Validación de entradas en tiempo de ejecución para asegurar que los datos ingresados por los usuarios cumplan con los formatos y restricciones requeridas.
- Inclusión de pruebas automatizadas (por ejemplo, con JUnit) para asegurar la integridad del sistema ante cambios futuros.
- Posible migración del sistema de archivos planos a una base de datos relacional o no relacional para mejorar la escalabilidad y seguridad del sistema.
- Implementación de un sistema de autenticación más robusto y control de acceso granular por rol de usuario.

Cierre: Este proyecto representa una base sólida sobre la cual se puede continuar construyendo un sistema completo de administración de parques de atracciones. El enfoque orientado a objetos y la claridad en el diseño facilitarán su evolución en las futuras entregas del curso.