Nombres:

- Juan Andres Pinto Pedraza - 202412084
- David Esteban Lavarde Osorio - 202225363

PROYECTO #1: PARQUE DE DIVERSIONES

Este documento presenta el diseño y desarrollo de una aplicación orientada a objetos en Java para la gestión de un parque de diversiones. La aplicación permite modelar atracciones, manejar empleados y visitantes, y gestionar la venta de boletos, incluyendo el registro y persistencia de datos.

A través de diagramas UML, se detallan las estructuras de clases, las relaciones entre entidades y el comportamiento del sistema, proporcionando una visión clara de la arquitectura y el flujo de trabajo del software.

INTRODUCCIÓN

El proyecto consiste en una aplicación que simula el funcionamiento interno de un parque de diversiones. Mediante una estructura de clases bien definida, el sistema permite representar los distintos tipos de atracciones, manejar los roles de los empleados, permitir el acceso de visitantes, y ejecutar acciones clave como la compra de boletas o el registro de nuevos empleados. Se ha hecho énfasis en una arquitectura modular, permitiendo que nuevas funcionalidades puedan añadirse con facilidad.

ARQUITECTURA DEL SISTEMA

El sistema se basa en una jerarquía centrada en la clase Atraccion, de la cual derivan otras más especializadas: Mecanica, Cultural, Evento y LugarDeServicio. Esta herencia permite extender funcionalidades particulares para cada tipo de atracción.

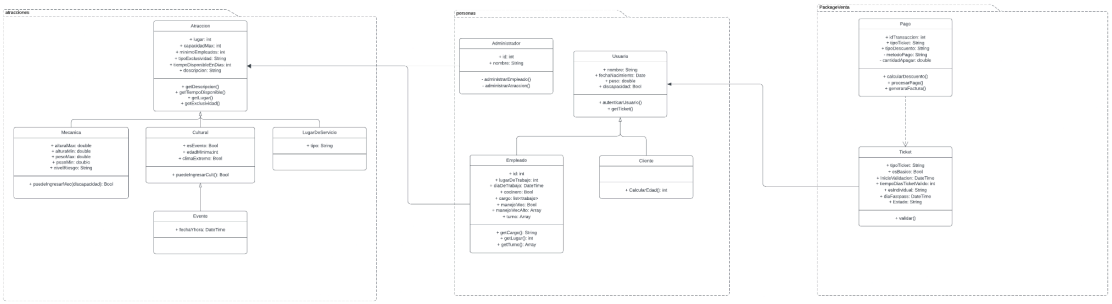

DIAGRAMA DE CLASES COMPLETO

El diagrama anterior representa todas las clases principales del sistema, incluyendo sus atributos, métodos y relaciones. Se destacan también clases como Empleado, Visitante, y Boleta, cada una con una responsabilidad clara en el modelo.

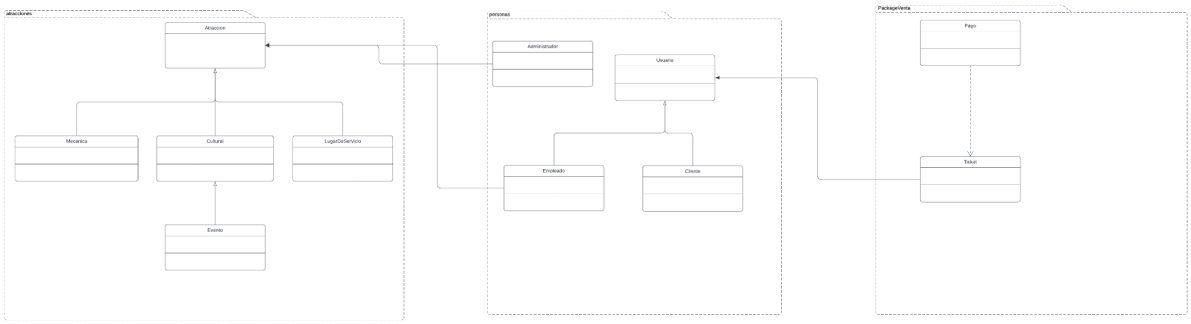

DIAGRAMA DE CLASES DE ALTO NIVEL

Este segundo diagrama ofrece una visión más general del sistema, permitiendo observar cómo se agrupan y relacionan las clases sin entrar en detalle de implementación. Es útil como referencia visual rápida de la arquitectura global.

FUNCIONALIDADES CLAVE

Gestión de Empleados

La clase Empleado permite representar a los trabajadores del parque. Se incluyen mecanismos para registrar, modificar y eliminar empleados. Un administrador es responsable de estas acciones, lo cual se refleja en los diagramas de secuencia.

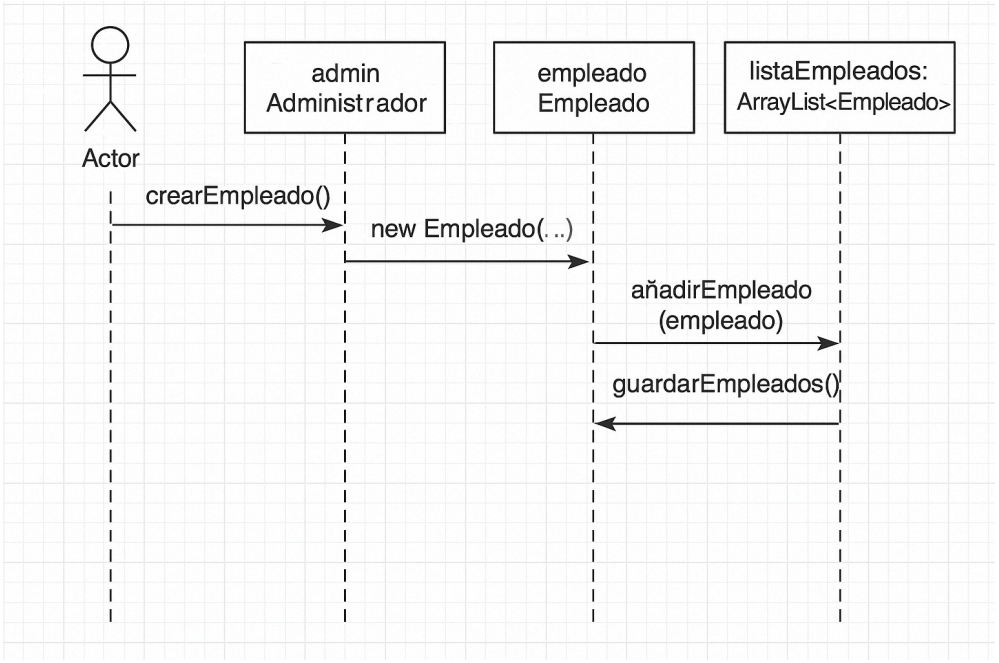

DIAGRAMA DE SECUENCIA. (REGISTRO DE EMPLEADO)

El diagrama anterior ilustra cómo se realiza el registro de un empleado desde la interfaz de administración: creación del objeto, adición a la lista de empleados, y almacenamiento usando persistencia de archivos.

Venta de Boletas

El visitante puede comprar entradas para una atracción específica, siempre que esta se encuentre disponible. El sistema calcula el precio, registra la venta y guarda los datos para su posterior recuperación.

Además, se incluye control de excepciones para manejar errores como la selección de una atracción inexistente, o datos inválidos durante el ingreso.

Módulo de Autenticación

Una parte central del proyecto es el sistema de autenticación, el cual permite controlar el acceso mediante usuarios y contraseñas. Cada tipo de usuario (administrador, visitante, empleado) tiene permisos específicos dentro del sistema.

La autenticación se realiza comparando credenciales contra datos almacenados en archivos planos. Este enfoque, aunque simple, permite mantener la lógica del sistema encapsulada y segura dentro de los parámetros del proyecto.

Persistencia de Datos

El sistema guarda y recupera la información utilizando archivos .txt, almacenados fuera del código fuente en una carpeta externa. Estos archivos contienen información sobre empleados, atracciones, usuarios y boletas vendidas.

empleadosP.txt – contiene la lista de empleados registrados.

atraccionesP.txt – almacena las atracciones del parque.

La clase encargada de manejar la persistencia incluye métodos de escritura y lectura mediante BufferedReader y BufferedWriter, y maneja excepciones de tipo IOException para asegurar la integridad de los datos.

Modularidad y Extensibilidad

Una de las fortalezas del sistema es su diseño modular. La separación de lógica en paquetes y clases específicas permite mantener el código organizado y fácilmente modificable. Por ejemplo, añadir un nuevo tipo de atracción o funcionalidad de filtrado solo requiere extender las clases existentes o añadir nuevas implementaciones sin alterar el núcleo del sistema.

Uso de Polimorfismo

El sistema hace uso de polimorfismo al tratar las atracciones desde la superclase Atraccion, permitiendo almacenar instancias de Mecanica, Cultural, Evento, y LugarDeServicio en una misma lista, y ejecutar métodos comunes de forma dinámica, sin importar su tipo específico.

Separación de Capas

Aunque no está dividido en capas formales como MVC (Modelo-Vista-Controlador), el proyecto hace una separación implícita:

Modelo: clases como Atraccion, Empleado, Boleta.

Controlador / Lógica de negocio: manejo de datos, control de flujo, validaciones.

Vista: interfaz basada en consola, que interactúa directamente con el usuario.

Esto favorece la mantenibilidad del sistema y permite escalar con relativa facilidad hacia una interfaz gráfica o una versión web en el futuro.