Documento de diseño

Este documento detalla el diseño del sistema de gestión para un parque de diversiones, con un enfoque en la justificación de decisiones clave relacionadas con la estructura y funcionalidad del sistema. El objetivo principal es proporcionar claridad sobre la organización del código, la lógica de interacción entre componentes, y cómo estas decisiones contribuyen a la mantenibilidad y escalabilidad del sistema a largo plazo.

El sistema ha sido organizado en seis paquetes principales, cada uno con una responsabilidad claramente definida. Esta separación favorece la modularidad, la independencia funcional entre componentes y facilita futuras extensiones del sistema sin comprometer la estabilidad del código existente.

El diagrama de clases de diseño incluye todas las clases del sistema, con sus atributos, métodos y relaciones, ofreciendo una visión detallada y estructurada de la arquitectura. Además, se presenta un diagrama de clases de alto nivel que muestra únicamente las relaciones entre clases, sin entrar en los detalles internos, lo cual permite una comprensión rápida y general del sistema.

Asimismo, se han desarrollado diagramas de secuencia para funcionalidades críticas, como el proceso de compra de tiquetes, la validación de acceso a atracciones y la asignación de roles a los empleados. Estos diagramas ilustran cómo interactúan los diferentes componentes en escenarios clave del sistema, facilitando su análisis, validación y futura implementación.

Paquete administrador

Controla el sistema central de operación del parque.

- Clase ParqueDeDiversiones: Es la clase principal, que centraliza operaciones: agregar atracciones, empleados, clientes, ventas, etc.

Paquete usuarios

Gestiona la información y comportamiento de los usuarios del parque.

- Clase Usuario (abstracta): Contiene atributos comunes como nombre, cédula, y métodos de autenticación.
- Clase Cliente: Hereda de Usuario. Puede comprar tiquetes, recibir descuentos y acceder a servicios del parque.
- Clase Empleado: También hereda de Usuario. Puede tener uno o varios roles, que determinan sus funciones en el parque.

Paquete roles

Define los distintos roles que puede desempeñar un empleado.

- Clase Rol: Puede representar roles como Cajero, Cocinero, Servicio General u Operador de Atracción.
- Esta estructura permite asignación múltiple de roles a un mismo empleado.

Paquete atracciones

Modela las atracciones del parque.

- Clase Atraccion (abstracta): Tiene atributos comunes como nombre, capacidad y nivel de riesgo.
- Subclases:
    - Mecanica: incluye atracciones que requieren operador y mantenimiento técnico.
    - Cultural: incluye museos, teatros, zonas de lectura.

Paquete restricciones

Modela las restricciones de uso de atracciones.

- Interfaz Restriccion: Define el método cumple(Usuario usuario).
- Clases concretas:
    - RestriccionEdad
    - RestriccionAltura

Paquete tiquetes

Gestiona la venta y el control de acceso al parque.

- Clase Tiquete (abstracta): Tiene tipo (básico, familiar, oro, diamante), fecha de validez, y permite control de acceso.
- Subtipos de Tiquetes:
    - Temporada
    - FastPass
    - EntradaIndividual

Paquete LugarDeServicio

Este paquete agrupa los distintos lugares físicos dentro del parque que no son atracciones, pero que prestan servicios fundamentales para el funcionamiento y experiencia de los visitantes, como cafeterías, taquillas y tiendas.

- Clase Lugar (abstracta): Representa un espacio de servicio dentro del parque. Define atributos como nombre, tipo, capacidad máxima, estado (abierto/cerrado), y una lista de atracciones asociadas. Declara el método abstracto realizarActividad(), que define la operación principal de cada lugar.
- Subclases concretas de Lugar:
    - Cafeteria: Realiza la actividad de servir comida y bebidas.
    - Taquilla: Se encarga de la venta de tiquetes.
    - Tienda: Gestiona la venta de productos y souvenirs dentro del parque.

Justificación de diseño

- La separación en paquetes sigue el principio de responsabilidad única, facilitando la comprensión, pruebas e implementación incremental.
- Se usan clases abstractas e interfaces para permitir la extensión futura (por ejemplo, nuevos tipos de atracción o de restricciones).
- Se permite la composición flexible: por ejemplo, una atracción puede tener varias restricciones.
- El diseño es extensible y abierto a nuevas funcionalidades, como control climático, reservas anticipadas, integración con aplicaciones móviles, etc.

Persistencia de Datos

El sistema implementa la persistencia de datos mediante archivos de texto (.txt), lo cual se justifica por los siguientes motivos:

1. Claridad y simplicidad: El uso de archivos .txt permite almacenar la información de forma estructurada y fácilmente legible, lo que facilita la verificación manual durante el desarrollo y pruebas del sistema. Esto es especialmente útil en proyectos educativos o en fases tempranas donde se requiere comprender el flujo de datos sin la complejidad de una base de datos.
2. Coherencia con el enunciado del problema: Dado que el sistema de parque de diversiones es un sistema independiente, no distribuido, y sin requerimientos de concurrencia o alta escalabilidad, el uso de archivos planos resulta suficiente para cumplir con los requisitos

planteados. El enfoque está alineado con los objetivos del proyecto: modelar un sistema funcional y completo desde el punto de vista lógico.

3. Modularidad y reutilización: La lógica de persistencia fue encapsulada en clases específicas responsables de leer y escribir los datos. Estas clases forman parte de una capa de persistencia modular:
    - ManejadorPersistenciaClientes: maneja clientes.txt
    - ManejadorPersistenciaEmpleados: maneja empleados.txt
    - ManejadorPersistenciaTiquetes: maneja tiquetes.txt
    - ManejadorPersistenciaAtracciones: maneja atracciones.txt
    - ManejadorPersistenciaLugares: maneja lugares.txt (si aplica)

   Esta separación permite futuras migraciones a tecnologías como JSON, XML o bases de datos relacionales sin afectar el resto del sistema.

4. Facilidad de pruebas y portabilidad: Al no depender de un sistema de gestión de base de datos externo, el sistema puede ejecutarse en cualquier entorno sin necesidad de instalación adicional. Esto lo hace portable y más accesible para pruebas rápidas o demostraciones.

Este diseño presenta una visión integral del sistema de Parque de Diversiones, justificando cada decisión basada en principios de diseño de software como la modularidad, separación de responsabilidades y mantenibilidad. La implementación del sistema se alineará con este diseño, asegurando que la documentación sea consistente con el código final.

Diagrama Final

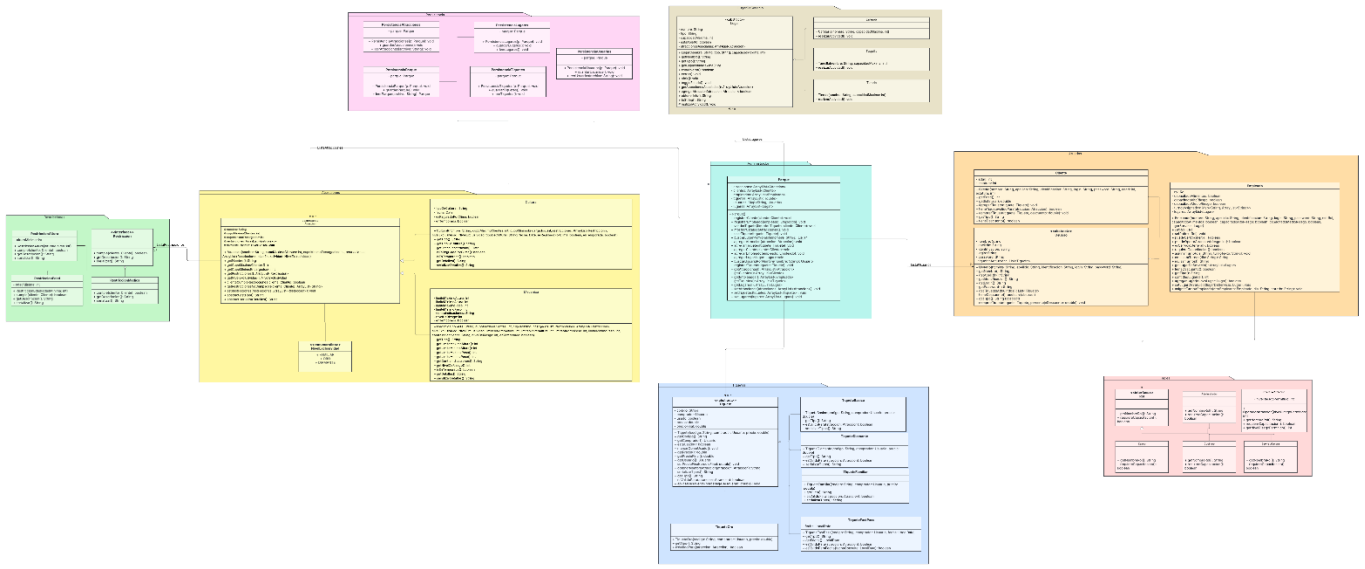

Diagrama de alto nivel

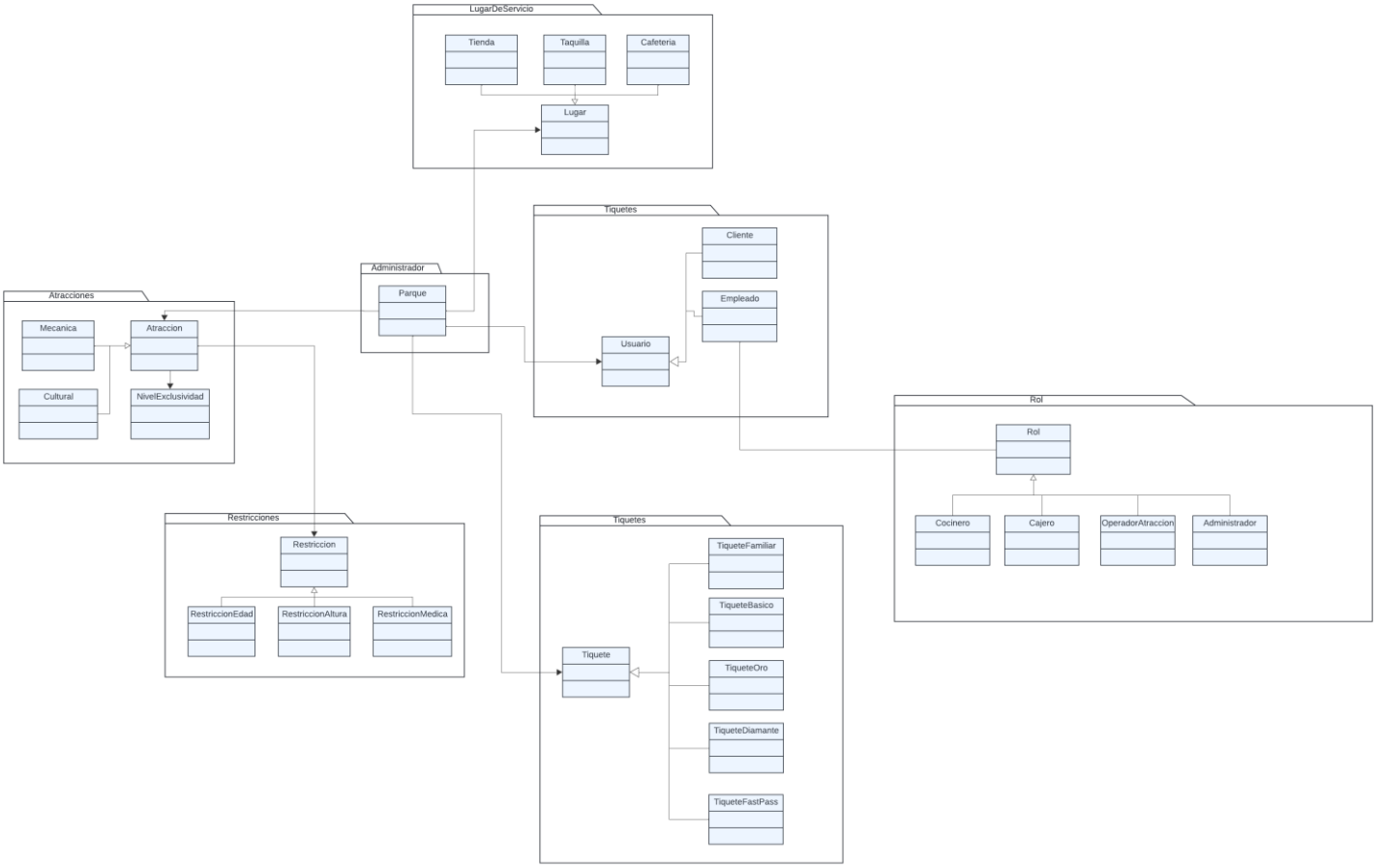

Diagrama de actividad para Cliente

Diagrama de actividad para administrador

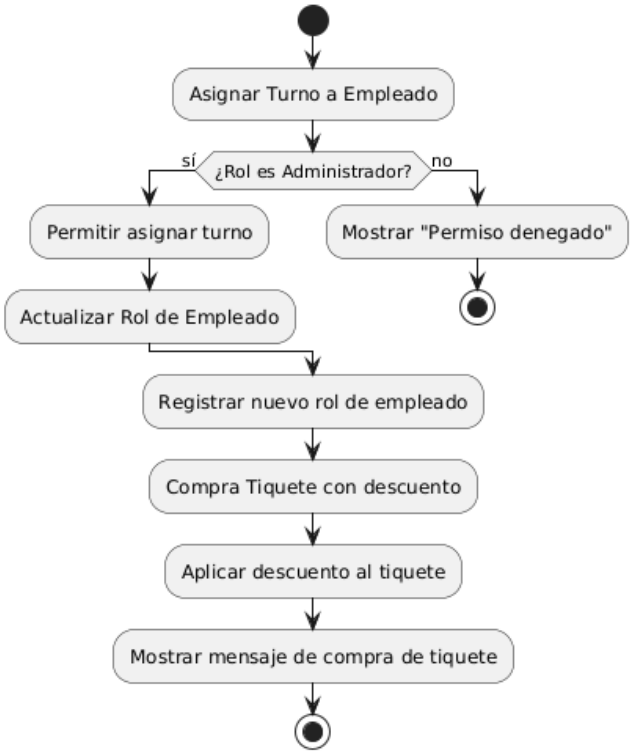

Diagrama de actividad Empleado