David Cantor – 202411394

Juliana Vera – 202411275

**Documento Diseño – Entrega 3**

**-** Diagrama de clases incluyendo interfaz

- Diagrama de clases de alto nivel

- Diagrama de clases de alto nivel de la interfaz

- Diagramas de secuencia

**Contexto general del diseño**

Nuestro modelo busca tener un sistema de alta cohesión y bajo acoplamiento, lo que garantiza una mayor extensibilidad y facilidad de mantenimiento. Se ha apostado por la centralización de la lógica del negocio a través de gestores especializados, como GestorPersonas, GestorTiquetes y GestorAtracciones, que permiten encapsular la funcionalidad crítica y favorecer la reutilización de código en distintas áreas del programa. Esta aproximación modular no solo organiza el sistema de forma clara, sino que también facilita la integración de nuevas funcionalidades sin alterar la estructura existente.

**Módulo de Personas**

En la sección de personas se aplicó un proceso de abstracción y herencia para modelar de manera eficaz los distintos tipos de usuarios del sistema, ya que, aunque existen diferencias entre ellos, comparten atributos fundamentales como el login, la contraseña y otros comportamientos comunes. La clase base Persona define estos elementos, mientras que clases derivadas como Empleado, Cliente y Administrador amplían o especializan dichos comportamientos. Los administradores se encargan de orquestar la operación del parque, interactuando con todos los gestores para crear y gestionar empleados, asignar roles, puestos de trabajo y turnos, y verificar la integridad de la información. Los empleados, por su parte, tienen responsabilidades en la validación y venta de tiquetes, tarea que se ejecuta en función de su rol y de los permisos asignados, lo que les permite operar determinadas atracciones. Asimismo, se les provee de mecanismos para visualizar su puesto de trabajo, rol actual y capacitaciones, elementos esenciales para el correcto desempeño de sus funciones.

**Módulo de Tiquetes**

El paquete de tiquetes centraliza todo lo relacionado con la comercialización y gestión de productos, abarcando tiquetes, fastPass, facturas y la administración de estos elementos. Se creó una clase abstracta, ItemVenta, que establece un contrato común para los objetos que se venden, permitiendo que tanto Tiquete como FastPass hereden sus propiedades y métodos. Esta decisión estratégica prevé la incorporación futura de nuevos productos, ya que cualquier objeto adicional de venta podrá implementarse extendiendo directamente de ItemVenta. La gestión de estos objetos se consolida en el GestorTiquetes, que se encarga de su almacenamiento, actualización y acceso, centralizando las interacciones que, de otra forma, dispersarían la lógica del negocio y complicarían el mantenimiento del código.

**Módulo de Atracciones**

La sección de atracciones es responsable de administrar el acceso, la operación y la disponibilidad de cada evento en el parque, ya sean atracciones mecánicas, culturales o espectáculos. El diseño contempla una división clara de métodos para cada tipo de evento, asegurando la correcta integración con otras partes del sistema. Por ejemplo, las atracciones mecánicas requieren la presencia de operadores capacitados, y el sistema valida tanto la capacidad operativa de los empleados (a través de sus capacitaciones y roles) como las restricciones de seguridad que deben cumplir los clientes (como condiciones de salud, altura y peso, además de restricciones ambientales como clima, fecha y hora). Para facilitar la reutilización y garantizar la coherencia en el comportamiento, se implementó una interfaz que permite que las distintas clases de atracción compartan métodos comunes, simplificando la extensión y el mantenimiento del código.

**Persistencia de Datos**

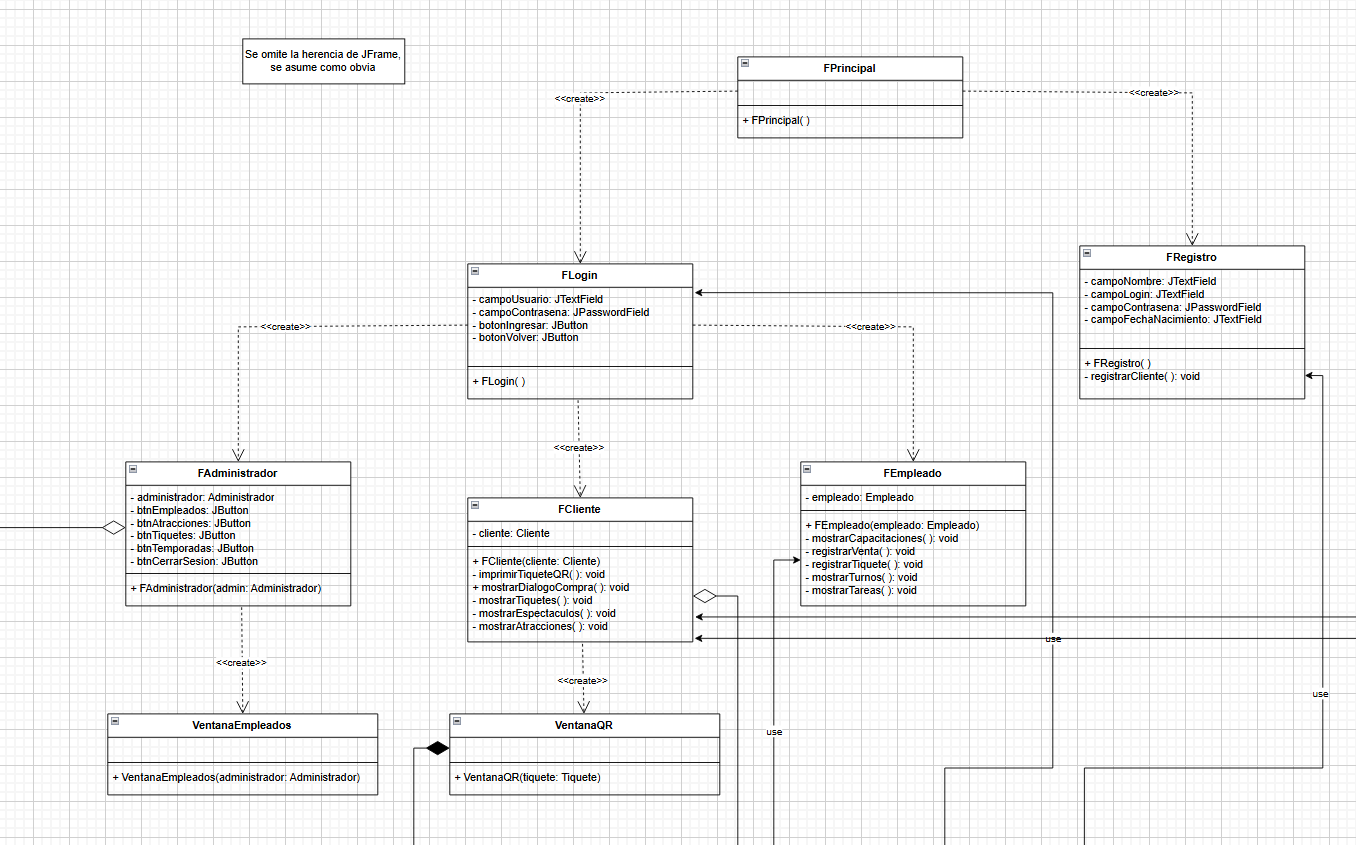
En cuanto a la persistencia, se optó por guardar cada objeto en archivos separados para mantener una organización clara y modular de la información. Este enfoque evita la concentración excesiva de datos en un único archivo, lo que podría dificultar el acceso y la manipulación de la información. Por ejemplo, dado que la clase Cliente contiene listas de facturas, tiquetes y fastPass, se decidió almacenar cada uno de estos elementos de forma individual, manteniendo una referencia en el objeto que lo contiene. Esto no solo mejora la escalabilidad del sistema, permitiendo actualizaciones y manipulaciones específicas sin afectar a otros módulos, sino que también previene la duplicación de información y simplifica la recuperación de datos en futuras iteraciones del proyecto.

**Otras Decisiones de Diseño**

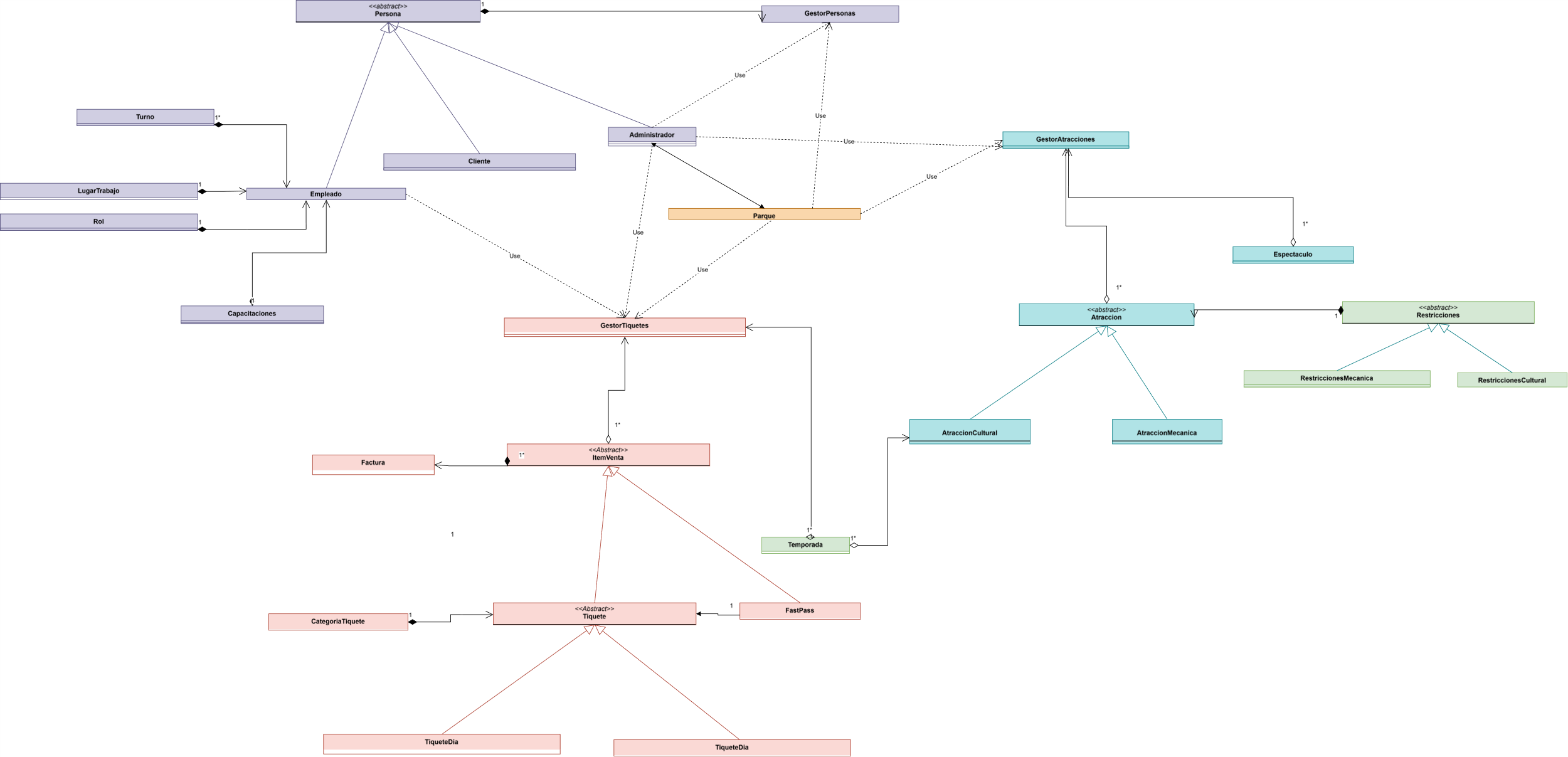
Entre las decisiones adicionales, se destaca el uso de HashMap en los gestores para facilitar búsquedas rápidas y eficientes basadas en claves únicas, lo que resulta fundamental para la gestión de empleados y clientes. La separación de LugarTrabajo y Rol se implementó para permitir una mayor flexibilidad en la reubicación de empleados, ya que distintos puestos de trabajo pueden requerir permisos y responsabilidades diferentes. Asimismo, el uso de ArrayList para el manejo de colecciones se eligió por su simplicidad y eficiencia en operaciones secuenciales, cuando no se requiere un acceso directo indexado de alta complejidad. En conjunto, estas decisiones no sol optimizan el rendimiento del sistema, sino que también refuerzan la capacidad del parque para evolucionar y adaptarse a nuevos requerimientos a lo largo del tiempo.

**Diagrama de clases (incluyendo interfaz)**

Abrir link para una versión más legible donde se ven las conexiones:  
<https://drive.google.com/file/d/13N8ifeXSXqYyvYALeQaC99rYqcEQlK_Y/view?usp=sharing>

**A screenshot of a computer screen

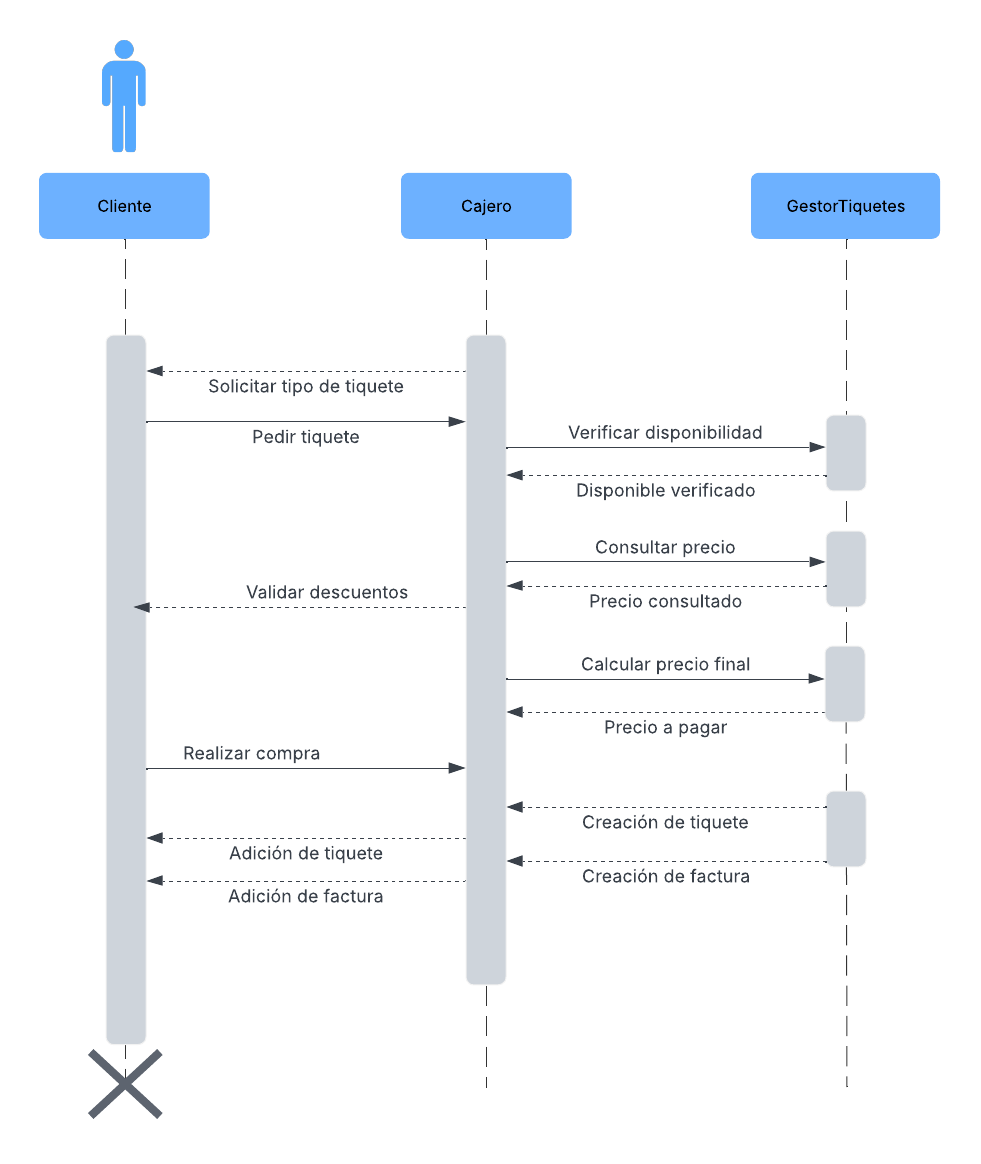
AI-generated content may be incorrect.**

**Diagrama de clases de alto nivel**

**Diagrama de clases de alto nivel de la interfazA diagram of a diagram

AI-generated content may be incorrect.**

**Diagrama de secuencia para venta de tiquetes**

****