Proyecto 1 Diseño y Programación a Objetos

Integrantes: Pablo Alejandro Pérez – 202413227

Lucas Ruiz – 202321331

Juan Manuel Hernández –202412958

Grupo 7

**1.Contexto:**

En este proyecto se construirá un sistema que apoye las operaciones administrativas de un parque de diversiones. Este sistema tendrá 3 funcionalidades principales.

1. Catálogo de las diferentes atracciones que tiene el parque.
2. Sistema de gestión de empleados y labores en las atracciones.
3. Permitirá la venta de tiquetes a los visitantes del parque

**2. Justificación:**

Para la realización de este proyecto se tuvieron muy en cuenta las principales funciones del programa que se desglosaron en cuatro gestores que vendrían siendo gestorPersonas, gestorTiquetes, gestorTurnos y gestorAtracciones, por medio de la consola podremos acceder a las principales funciones de estos gestores que vamos a explicar a continuación:

**2.1 GestorPersonas:**

Este gestor tiene como principal función distinguir entre los roles de los usuarios que existen en el parque: diferencia los roles del empleado (que puede tener una especialidad si está encargado de una atracción en específico) o puede contar con otro tipo de especialidad que no involucre las atracciones mecánicas, siendo cocinero o desempeñando otro rol dentro del parque. Además, contamos con el administrador, que posee la capacidad de acceso y gestión completa del sistema. Adicionalmente, contamos con el cliente, cuyo principal objetivo es la compra de tiquetes en el parque.

Gestor Personas también utiliza una persistencia de usuarios que permite guardar, dentro de un archivo, la lista de usuarios registrados; recordemos que tanto los empleados, administrador y clientes tienen un login y pasword, los cuales guardaremos en este archivo. Por ende, este archivo, además de asegurar estos datos, también funcionará como herramienta para gestionar los turnos dentro del parque. Esta gestión se realizará por medio de la creación de un turno para un empleado en determinado día de la semana; esto deriva en el trabajo que se tiene que realizar dentro de una atracción en un tiempo establecido. Claramente, gracias al GestorPersonas, podremos verificar que el empleado asignado a esta tarea esté libre para poder realizarla.

**2.2 GestorTurnos:**

A pesar de que no se relaciona directamente con el parque es una pieza fundamental de la implementación porque se utiliza en gestión atracciones y en gestión personas, este tiene una relación más estrecha con gestorPersonas debido a lo explicado anteriormente, pero a su vez también tiene una gran importancia con GestionAtracciones, importancia la cual abordaremos más adelante en el documento.

**2.2 GestorTiquetes:**

El GestorTiquetes tiene como principal funcionalidad administrar la compra y el precio de los tiquetes. A partir de esta clase, se pueden realizar no solo las compras, sino también la creación de identificadores y el cambio de precios. Esta clase colabora con la clase abstracta “Tiquete” para confirmar que se cumplan las condiciones de compra: que no esté usado; que, si es TiqueteDeTemporada, tenga fechas establecidas; o que el tiquete tenga una categoría asociada (Diamante, Oro, Familiar).

Adicionalmente, realiza una confirmación de la compra del tiquete, validando que la operación haya sido satisfactoria. Junto a esto, GestorTiquetes trabaja con PersistenciaTiquetes, la cual le permite guardar los códigos de los tiquetes y registrar el tipo de tiquete a partir de un identificador y un contador (es decir: si el tiquete es general, se guardará con el prefijo “GENE”; si es individual, con “INDUE”; y el contador distingue cuántos tiquetes de cada tipo existen). Al mismo tiempo, PersistenciaTiquetes está relacionada con PersistenciaFastPass —encargada de guardar los FastPass adquiridos por los clientes— y con CatalogoPrecios, que contiene el archivo con los precios guardados y asignados.

**2.4 GestorAtracciones:**

El GestorAtracciones tiene la responsabilidad de organizar las atracciones del parque y trabaja con la clase abstracta “Atraccion” para cumplir esta tarea. Como sabemos, existen distintos tipos de atracciones, como espectáculos, atracciones culturales y atracciones mecánicas. Estas, a su vez, cuentan con restricciones que se deben cumplir de manera obligatoria para su operación.

Algunas de estas restricciones son el clima, que exista un mínimo de trabajadores, y también, si se llega a dar el caso, que un empleado especializado maneje una atracción que tenga un alto riesgo. Por lo tanto, tanto Atraccion como GestorAtracciones se encargan de validar estos requisitos.

GestorAtracciones y Atraccion cuentan con dos herramientas importantes:

1. La relación de GestorAtracciones con la clase abstracta LugarTrabajo, que a su vez se relaciona con la clase Turnos, la cual tiene como objetivo asignar a los empleados a la atracción que sea necesaria. Esto, claramente, realizando las verificaciones de gestionTurnos explicadas anteriormente.

1. La relación de la clase abstracta Atraccion con PersistenciaAtracciones, para poder guardar las atracciones y además tener acceso a estas.

**3. Decisiones Clave al momento de la implementación:**

Aumentar el número de persistencias: Esto se debe a que consideramos necesario aumentar el número de textos para guardar más información que puede llegar a ser útil para el programa, como seria el contador o los fastpass que adquieren los clientes

Uso de restricciones clima: Se hace una clase para hacer referencia a una restricción de clima la cual tiene el nombre y una lista de los espectáculos y atracciones que pueden ser afectados por la restricción. Esta decisión es crucial al no ser un String atributo de estos ya que facilita accesibilidad de la restricción y a quienes afecta. De esta manera se acerca el problema al administrador y no solo es un problema de la atraccion. Facilita las acciones de parar el funcionamiento de un grupo de atracciones, o de autorizar la operación de estas.

Uso del atributo funcionando: A pesar de ser algo obvio, el atributo funcionando es implícito en la guia, que facilita operaciones en las cuales se tenga que cancelar la operación de una atracción o espectáculo sin sacarla del programa. Esta decisión se puede ver en la restricción clima.

Uso de la clase especialidad: La clase Especialidad tiene un nombre y una lista de atracciones y de lugares de trabajo. Esta clase es importante para poder ver el empleado donde puede trabajar. Este requerimiento se puede cumplir con un boolean de accesibilidad a los trabajos, pero esta manera tiene más alcance en los empleados y en las atracciones. También está más preparada para posibles cambios en el futuro del código.

**4. Implementaciones incorporadas a lo largo del proyecto:**

A partir de lo requerido en este proyecto 2, se realizaron distintas modificaciones a lo planteado anteriormente durante el proyecto 1, empezando por un análisis a la forma de manejar los datos del programa, de manera que Especialidades, Clima y RestriccionSalud, en vez de tener una lista de atracciones, se optó porque fuera un string. Adicionalmente, en la persistencia, para el correcto manejo de esta, todo list se convirtió en ArrayList y también se le añadió el atributo de compra a Cliente.

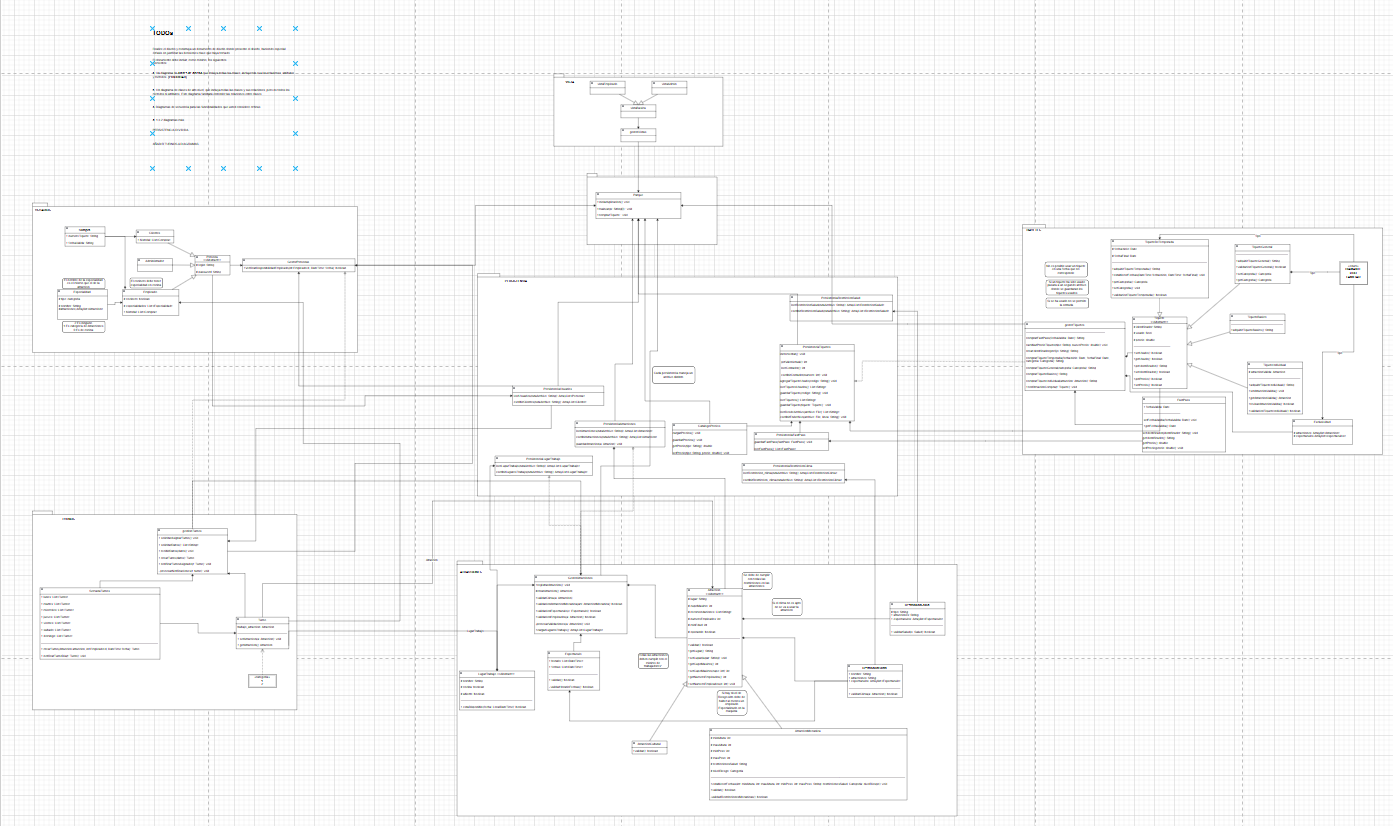
Un aspecto nuevo que se mencionó anteriormente es la incorporación de RestriccionSalud, una restricción que se añade a la parte de Atracciones, la cual se basa en una lectura de las posibles afectaciones de salud que puedan tener los clientes, que puedan limitarlos a montarse a determinadas atracciones. Por medio de PersistenciaRestriccionSalud se puede leer y escribir este tipo de restricciones.

De manera similar, se consideró que es fundamental leer los lugares de trabajo y escribir nuevos, por lo que se implementó una nueva persistencia llamada PersistenciaLugarTrabajo. Esta conecta con LugarTrabajo, claramente, pero a su vez hace que el GestorAtracciones implemente métodos de esta persistencia, generando respectivamente una relación de dependencia.

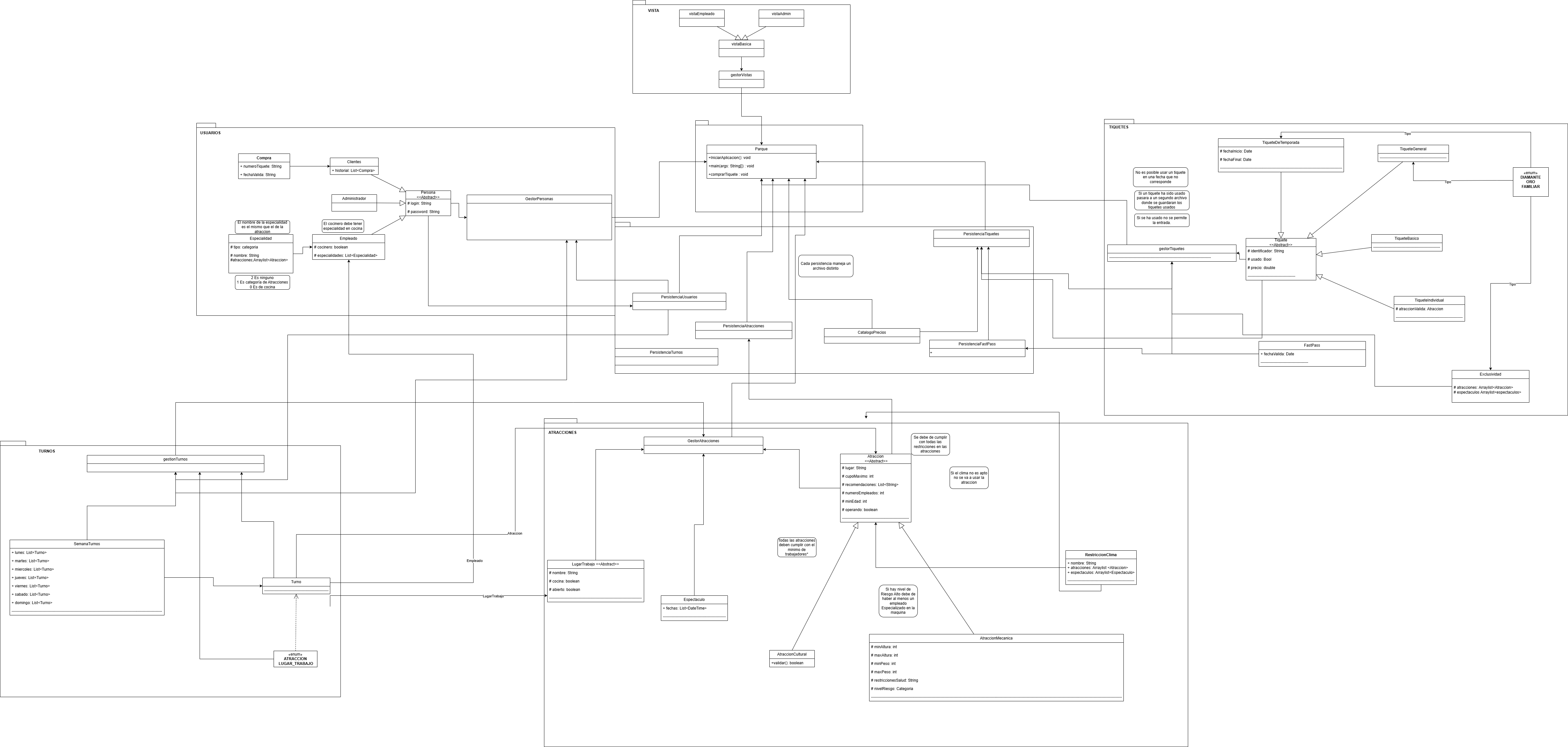
En conclusión, los cambios durante el proyecto 2 fueron, en mayor parte, dirigidos a la persistencia, que deriva de los requerimientos que este mismo solicitaba. Los diagramas presentados a continuación, de igual forma, también fueron modificados.

**4. Diagramas Elementales:**

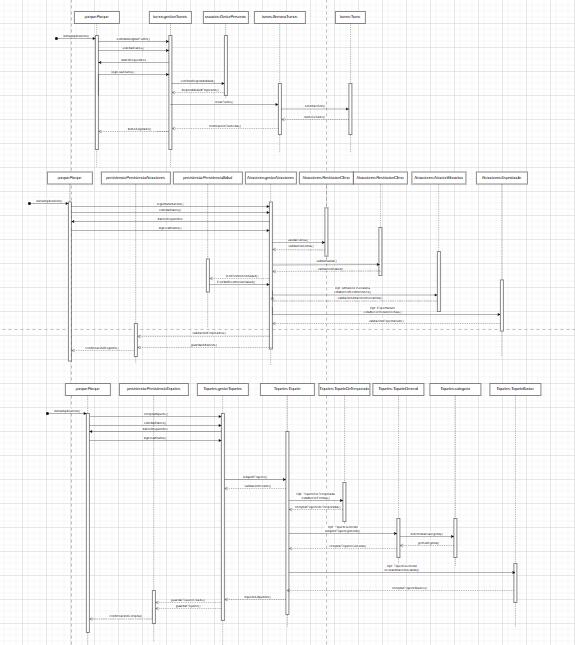
**4.1 Diagrama de clases:**



**4.2 Diagrama de clases de alto nivel:**



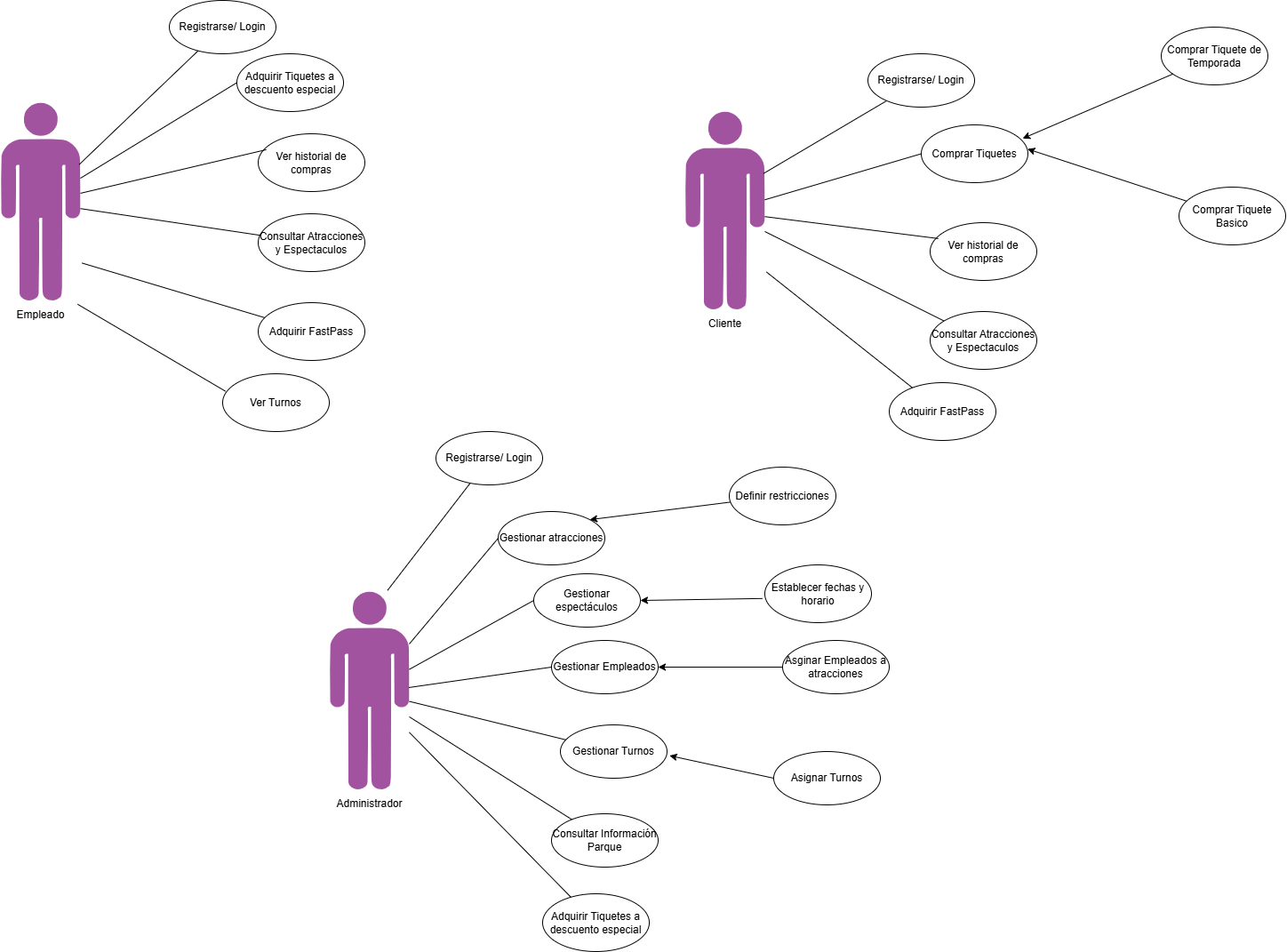
**4.3 Diagrama de secuencia:**



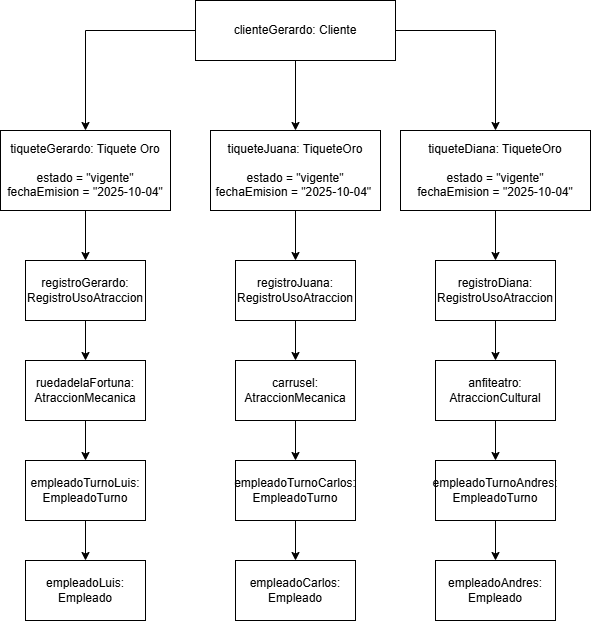
**Nota: si desea ver los diagramas expandidos, siga este enlace:** [Diagramas Segunda entrega](https://uniandes-my.sharepoint.com/:f:/g/personal/p_perezm_uniandes_edu_co/EjFxCJkL4CxMnProjCvZaoEBtMOpvLWv0L2AQ0fgK3GmnQ?e=aEWemb) En caso de no servirle, contactar a [p.perezm@uniandes.edu.co](mailto:p.perezm@uniandes.edu.co)

**5. Diagramas Adicionales:**

**5.1 Diagrama de uso:**



**5.2 Diagrama de objetos:**



**6. Pruebas:**

PruebasGestorAtracciones:

Realiza testActivarRestriccionClima el cual llama al gestor para deshabilitar o marcar como no disponible atracciones o espectáculos que puedan estar afectados por la lluvia, principalmente tiene como objetivo comprobar que al activar una restricción de clima se filtren correctamente las atracciones.

PruebaPersistenciaAtracciones:

Utilizar testEscritorAtraccion y testLectorAtraccion para crear dos objetos de atraccion y también verificar que la lectura de la lista desde el archivo sea correcta, de manera similar ocurre con los otros lectores como lo son testLectorRestriccionClima, testLectorRestriccionSalud, testLectorLugarTrabajo, testLectorEspectaculo; Su principal objetivo es asegurar de que las clases de persistencia puedan leer y escribir de manera óptima las atracciones y espectáculos, junto con sus restricciones.

PruebaPersistenciaUsuario:

Trabaja de una manera muy similar a la PruebaPersistenciaAtracciones solo que ahora trabaja con la persistencia de usuarios como los empleados y sus especialidades o los mismos clientes, verifica por medio de testLectorEmpleado, testLectorCliente, testLectorEspecialidad y testEscritorCliente que todo funcione sin errores, retornando lecturas de los datos esperados para empleados y clientes.

Adicionalmente para las pruebas unitarias se aprovechó para hacer las pruebas de la clase del GestorTiquetes que aprovechan de una vez la persistencia, y prueban que no solo funcione la persistencia si no que los métodos de comprar tiquetes de cada tipo sirvan adecuadamente, también verifica que se cree bien el contador.

La prueba de turnos hace algo similar, verifica que las pruebas estén guardando los turnos y que los puedan leer.

**7. Historias de Usuario:**

**Historia de Usuario-Cliente:**

Registro dentro del sistema del parque para la compra del tiquete

Entradas:

-Nombre de usuario (String)

-Contraseña (String)

Flujo del programa:

1. El cliente inicio por medio de la consola el “Registro de Cliente”
2. Ingresa su nombre y contraseña
3. Se realiza una validación de que el nombre de usuario no exista.
4. El sistema guarda el nuevo cliente en el archivo clientes.txt

Salida:

-Archivo clientes.txt se actualiza con el nuevo cliente

-” Registro exitoso”

**Historia de Usuario- Administrador:**

El administrador quiere registrar una nueva atracción dentro del parque

Entradas:

-Nombre de la atracción (String)

-Tipo (Mecanica/Cultural/Espectaculo)

-Lugar de trabajo asociado (String)

-Número mínimo de empleados (int)

-Restricciones de clima (Lluvia u otro)

-Restricciones de salud (Problemas cardiacos u otro)

-Parámetros específicos (Edad mínima o Altura mínima)

Flujo del programa:

1. El administrador al ya estar registrado inicia sesión en la consola y se dirige al Gestor de Atracciones
2. Selecciona “Registrar Atraccion” e ingresa los datos solicitados.
3. Se valida que se cumplan los requisitos
4. Se llama a PersistenciaAtracciones para guardar la nueva atracción
5. En caso de ser un nuevo lugar de trabajo se llama a PersitenciaLugarTrabajo.

Salida:

-Se actualiza atracciones.txt y si aplica también se actualiza Lugartrabajo.txt.

-” La atracción fue registrada”

**Historia de Usuario- Empleado:**

El empleado quiere consultar y acatar su turno de trabajo para saber dónde y cuándo se debe presentar.

Entradas

-Usuario (String)

-Contraseña (String)

-Fecha o rango de fecha a consultar (DateTime)

-ID del turno (int)

Flujo del programa:

1. Inicia sesión en la consola y se dirige al Gestor de Turnos
2. Consulta los turnos y el sistema lee semana\_turnos.txt
3. Se muestra la lista de turnos disponibles para el empleado
4. El empleado lee la información del turno y acepta uno
5. En PersistenciaTurnos se guarda el turno aceptado por empleado

Salida:

-Archivo semana\_turnos.txt actualizado

-” Turno acatado”