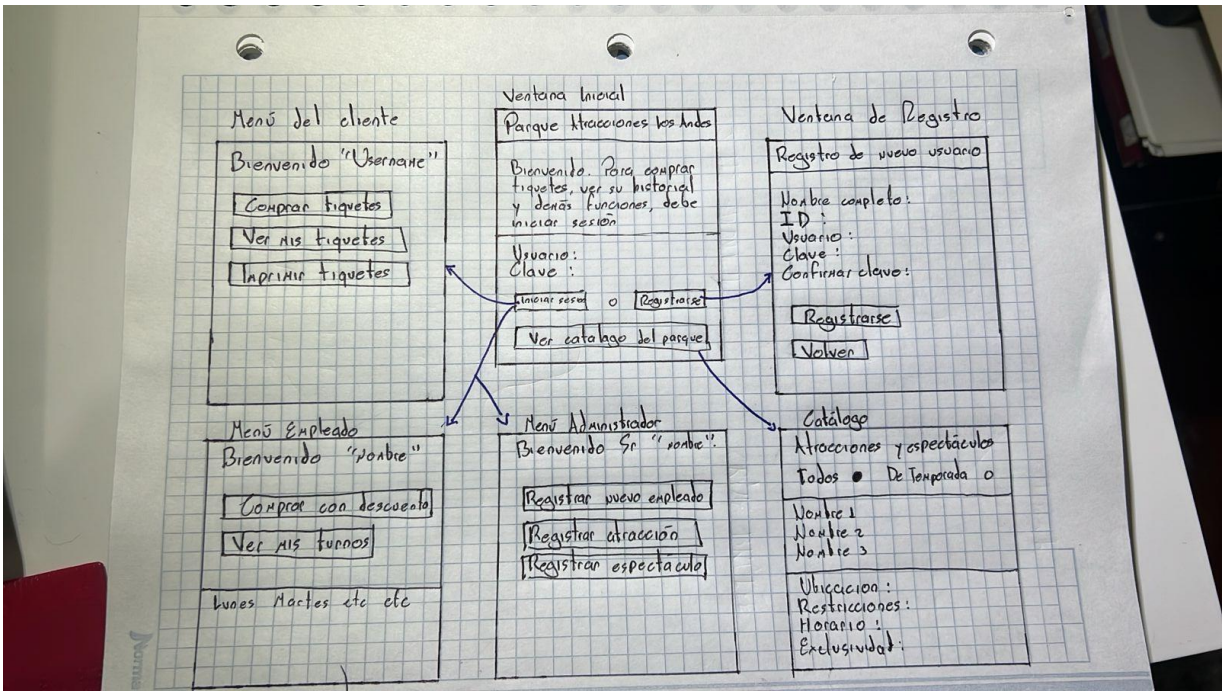
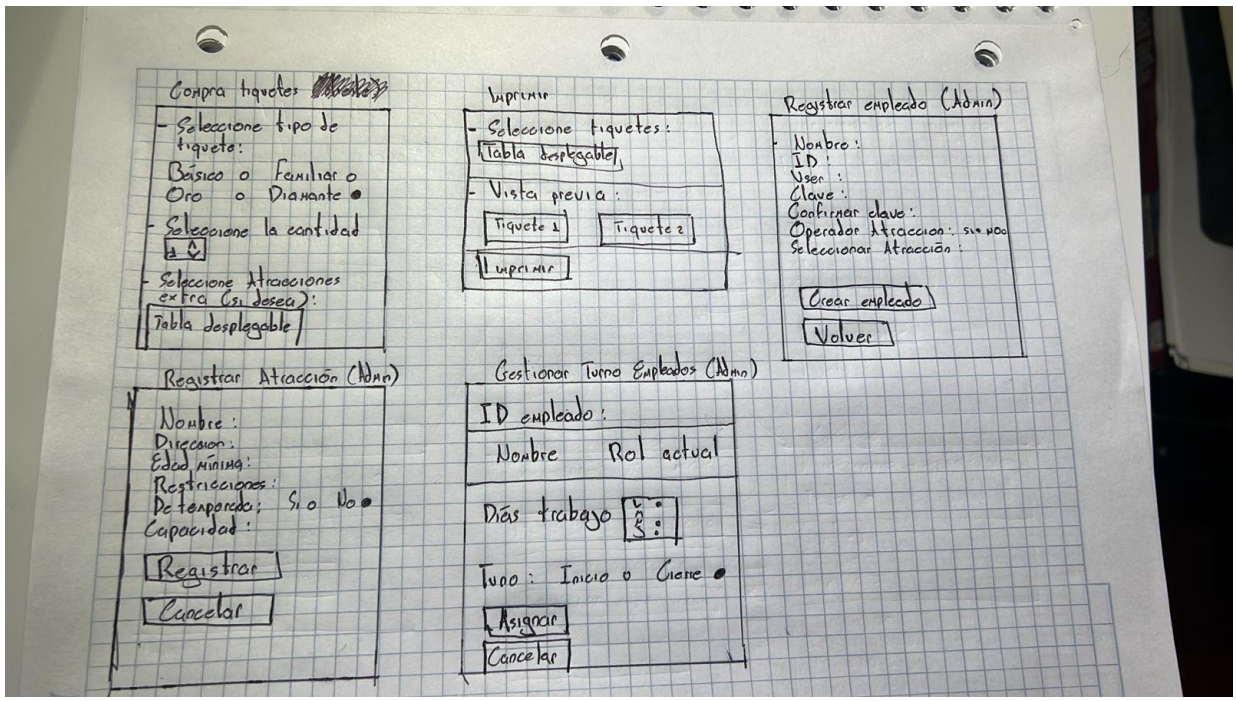
**Documento de Diseño para las Interfaces.**

**Sistema de Ventanas:** Nuestro diseño se basó en un sistema de ventanas, orientados a desacoplar el servicio de compra de tiquetes, propio de todos los tipos de usuarios, con las interfaces específicas de los trabajadores del parque, pues todos, sin importar su estatus, deben poder acceder a un portal de compras. A continuación se muestra un diagrama del sistema:



**Ventana Inicial**

Esta es la ventana que aparecerá a penas se inicialice el programa. Va a contar con la posibilidad de escribir sus credenciales en dos boxes separados, para que el programa pueda identificar si son clientes naturales, empleados o Administradores. Además, orientado a aquellos que estén interesados en ver solamente, sin necesidad de crear una cuenta, se agregó un botón de mostrar el catálogo del parque**. Por último, se agregó un botón para registrar nuevos usuarios***.

***Ventana Menú del Cliente (Naturales)**

En esta ventana aparecerán solo 3 botones: Comprar Tiquetes, Ver mis tiquetes e Imprimir Tiquetes

**Ventana Comprar Tiquetes**

Esta ventana recopilará la información necesaria para que la lógica pueda generar el tiquete asociado a la compra en la clase Gestor Ventas. En este orden de ideas, el diseño es tipo formulario. Se podrá seleccionar:

- Tipo de Tiquete (Básico, Familiar, Oro o Diamente)
- Cantidad de Tiquetes (Permitirá Typear el número en un box)
- Atracciones Extras

**Ventana Ver mis Tiquetes**

Desplegará la persistencia asociada a las lista de tiquetes registrados con ese usuario.

**Ventana Imprimir Tiquetes**

Permitirá seleccionar de la lista de tiquetes, vía tabla despegable, un tiquete para imprimir. Al darle al botón de imprimir, se desplegará una imagen como la siguiente, en la que la información del tiquete se mostrará en un código QR

****Ventana Catálogo Parque**

Mostrará la persistencia asociada al Catálogo del Parque que se hace automáticamente cuando se actualiza la clase ParqueAtracciones en el modelo.

***** Ventana Registrar Usuario**

Permitirá recolectar la información para poder llamar al constructor del Usuario. Tendrá las siguientes opciones para llenar en boxes:

- Nombre Completo
- ID
- Usuario
- Clave
- Confirmar Clave

Y un botón para confirmar el formulario

***Ventana Menú Empleado**

Contará con dos botones: Comprar con Descuento y Ver mis Turnos

**Ventana Comprar con Descuento**

Redirigirá al menú de cliente, pero con la salvedad de los descuentos asociados a la compra del staff.

**Ventana Ver mis Turnos**

Se desplegará la persistencia asociada a los empleados, que muestran las actividades a realizar por día.

***Ventana Menú Administrador**

Tendrá 3 botones asociados a las funcionalidades principales: Registrar nuevo empleado, Registrar atracción, Registrar Espectáculo y Gestionar Turno Empleados:

**Ventana Registrar nuevo Empleado**

Permitirá recolectar la información para poder llamar al constructor del Usuario. Tendrá las siguientes opciones para llenar en boxes:

- Nombre Completo
- ID
- Usuario

- Clave
- Confirmar Clave
- Seleccionar Tipo de Operador (Alto, Medio Riesgo)
- Seleccionar Atraccion(es).

Y un botón para confirmar el formulario

**Ventana Registrar Atracción**

Otra ventana tipo formulario, contará con la información necesaria para poder llamar correctamente al constructor de las atracciones. En este sentido tendrá los siguientes boxes para rellenar:

- Nombre
- Direccion
- Edad Mínima
- Restricciones
- Capacidad

Y un botón para seleccionar si es de Temporada o FastPass

**Ventana Registrar Espectáculo**

Otra ventana tipo formulario, contará con la información necesaria para poder llamar correctamente al constructor de los espectáculos. En este sentido tendrá los siguientes boxes para rellenar:

- Nombre
- Direccion
- Edad Mínima
- Restricciones
- Capacidad

**Ventana Gestionar Turno Empleados**

Se mostrará la persistencia asociada a los empleados en una tabla despegable, de tal manera que el administrador pueda ver toda la información de los empleados registrada hasta el momento. Además, tendrá la opción de poner el nombre del empleado, seleccionar los días de trabajo y la actividad que se quiere agregarle al empleado (Todo esto via tablas desplegables, que sea que el administrador solo deba oprimir un botón). Al final, estará un botón “Asignar”: Si el nombre está correcto, se registrará correctamente la información; sino, se pedirá que se llenen de nuevo los datos.

**SISTEMA MVC:**

Para este programa, se utilizó el patrón de diseño Modelo-Visor-Controlador. Hasta ahora habíamos realizado correctamente, y pensando en esta etapa, la lógica del programa, que será nuestro modelo. El visor serán las diferentes clases asociadas a cada una de las ventanas mencionadas, y los controladores serán los que le dirán: qué ventana mostrar, y le mandará la información recolectada por cada ventana a la lógica, para luego ponerla en un formato que el visor pueda reconocer. En este orden de ideas, se muestran las clases asociadas a los controladores. Se optó por un controlador por usuario, pues creemos que las funcionalidades se empaquetan correctamente según la persona que los vaya a utilizar: esta división sigue el principio de responsabilidad única, lo cual mejora la mantenibilidad, escalabilidad y claridad del código.

**ControladorCliente**

Este controlador se encarga de todas las operaciones que puede realizar un usuario estándar (visitante del parque). Incluye métodos como:

- comprarTiqueteIndividual, comprarTiqueteRegular, comprarTiqueteTemporada y comprarFastPass: Estos métodos gestionan la adquisición de diferentes tipos de tiquetes por parte del cliente.
- verTiquetes: Permite al usuario consultar sus compras anteriores.
- consultarCatalogo: Proporciona acceso al catálogo de atracciones y espectáculos.

El ControladorCliente interactúa con la clase Usuario, el ParqueAtracciones y, de manera importante, con el GestorVentas, a quien delega la lógica específica de la venta de tiquetes. También utiliza QRCodeGenerator para generar los códigos QR de los tiquetes comprados.

**ControladorEmpleado**

Este controlador maneja las acciones específicas de los empleados, quienes son también usuarios pero con permisos adicionales. Las funciones incluyen:

- verTurnos: Permite al empleado consultar sus asignaciones de trabajo (Labor).
- comprarTiquete: Los empleados también pueden comprar tiquetes, pero este método es más general y parametrizado.

El ControladorEmpleado interactúa con la clase Empleado (subtipo de Usuario), el ParqueAtracciones y el GestorVentas, además de usar QRCodeGenerator. Aunque su funcionalidad de compra de tiquetes es más limitada, su diseño modular permite reutilizar lógica común con el controlador de cliente.

**ControladorAdministrador**

Este controlador encapsula todas las operaciones administrativas del sistema. Tiene las siguientes responsabilidades:

- **Gestión de personal**: registrarEmpleado y asignarTurno.
- **Gestión de atracciones y espectáculos**: Métodos como registrarAtraccion, eliminarAtraccion, modificarAtraccion, registrarEspectaculo, etc.
- **Compra de tiquetes**: Puede comprar tiquetes como cliente o empleado, usando comprarTiqueteComoAdmin.
- **Reportes**: Genera informes y catálogos mediante generarInforme y generarCatalogo.

El ControladorAdministrador interactúa con Administrador, ParqueAtracciones, GestorVentas y QRCodeGenerator.

**Interacción entre Clases Controladoras:**

La interacción entre clases está diseñada para mantener una **estructura jerárquica desacoplada**:

1. Los **controladores** actúan como intermediarios entre la interfaz de usuario (o capa de presentación) y las entidades del dominio (Usuario, ParqueAtracciones, etc.).
2. **GestorVentas** es un componente centralizado al que todos los controladores acuden para realizar ventas, garantizando una lógica de negocio uniforme para los tiquetes.
3. El **QRCodeGenerator** es un servicio utilitario que los controladores usan sin necesidad de acoplarlo directamente a las clases de modelo.
4. El **ParqueAtracciones** actúa como repositorio y coordinador de entidades clave como Atraccion, Espectaculo, Empleado, etc., y permite la consulta y modificación de estas.

Esta arquitectura favorece una **alta cohesión y bajo acoplamiento**, lo que facilita el mantenimiento y futuras ampliaciones del sistema.

**ControladorPrincipal**

Este controlador es el punto de entrada al sistema. Se encarga de funciones generales que no están relacionadas con un usuario ya autenticado, como:

- autenticarUsuario: Verifica credenciales y determina el tipo de usuario (Cliente, Empleado o Administrador).
- registrarUsuario: Permite registrar nuevos usuarios en el sistema.

**Razón de su existencia**: Centraliza las operaciones de acceso al sistema, evitando duplicación de lógica de autenticación. También se asegura de que el resto del sistema trabaje con objetos Usuario válidos.

**Comunicación**: Se comunica con el ParqueAtracciones para registrar usuarios y validar accesos. Una vez autenticado, delega el flujo a los controladores específicos (Cliente, Empleado o Administrador).

**UML's:** Se visualizan mejor en los PNG adjuntos: "UML controladores E interfaces" y "UML IMAGEN". Este último muestra el UML del Modelo. Respectivamente: