Nombres:

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

PROYECTO #1: PARQUE DE DIVERSIONES

Este documento presenta el diseño y desarrollo de una aplicación orientada a objetos en CV 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.

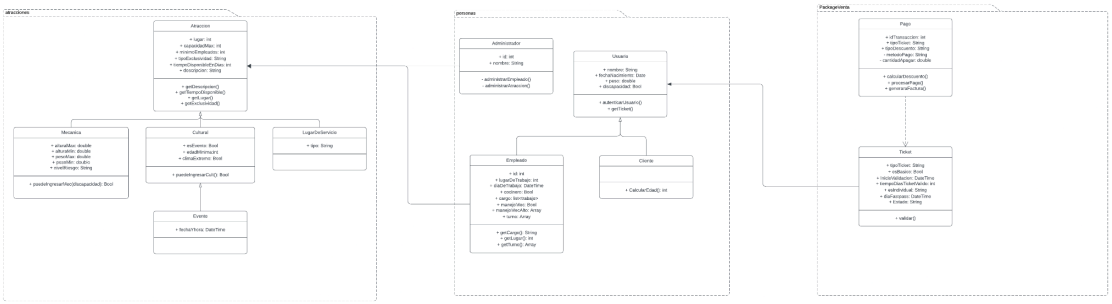


DIAGRAMA DE CLASES COMPLETO

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.

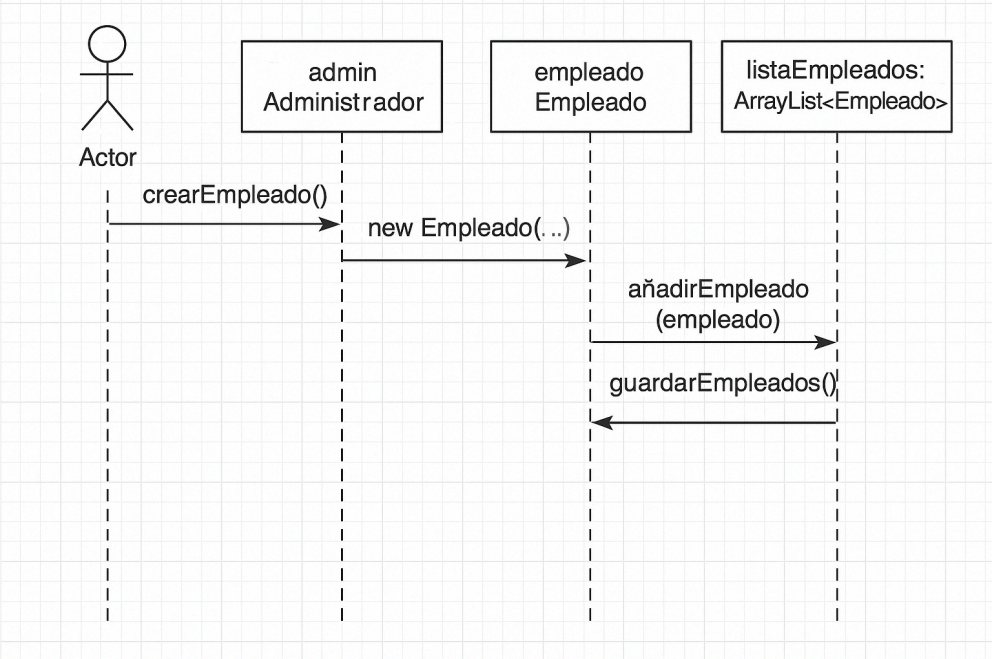


DIAGRAMA DE SECUENCIA. (REGISTRO DE EMPLEADO)

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.

adminsP.txt - contiene la lista de administradores registrados.

clientesP.txt - contiene la lista de los clientes 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.

PROYECTO #2: PARQUE DE DIVERSIONES

1. Historias de Usuario

En esta fase del proyecto se definieron y aplicaron historias de usuario detalladas para cada uno de los tres roles del sistema: Administrador, Empleado y Cliente. Estas historias orientaron tanto el desarrollo de la lógica de negocio como la estructura de las interfaces de consola.

Administrador:

- Historia A1: Añadir atracción
  Como administrador, quiero poder añadir nuevas atracciones, para que estén disponibles para los visitantes y el personal.
  Entradas: tipo de atracción, nombre, parámetros operativos (capacidad, duración, exclusividad, etc.).|
  Resultado esperado: la atracción queda registrada y accesible en el sistema.
- Historia A2: Editar atracción
  Como administrador, quiero modificar la información de una atracción existente, para actualizarla conforme a nuevas condiciones del parque.
  Entradas: selección de atracción y datos modificados.
  Resultado esperado: la información se actualiza exitosamente.
- Historia A3: Eliminar atracción
  Como administrador, quiero poder remover atracciones obsoletas o en mantenimiento, para mantener vigente la oferta del parque.
  Entradas: ID de la atracción.
  Resultado esperado: la atracción desaparece de la lista de atracciones.
- Historia A4: Gestionar empleados
  Como administrador, quiero agregar, editar o eliminar empleados del sistema, para mantener actualizada la plantilla de personal.
  Entradas: datos personales, puestos, turnos, áreas asignadas.
  Resultado esperado: cambios reflejados en la base de empleados.

Empleado:

- Historia E1: Consultar información personal
  Como empleado, quiero ver mi información laboral, para conocer mis tareas asignadas, turnos y áreas de operación.
  Entradas: inicio de sesión.
  Resultado esperado: despliegue de información personalizada.
- Historia E2: Registrar asistencia/tareas
  Como empleado, quiero registrar mi asistencia o tareas diarias, para mantener un historial de trabajo actualizado.
  Entradas: tipo de tarea, horario, comentarios.
  Resultado esperado: datos almacenados para consulta administrativa.
- Historia E3: Reportar problemas
  Como empleado, quiero reportar incidentes operativos, para que sean atendidos por el administrador.
  Entradas: descripción del incidente.
  Resultado esperado: mensaje de confirmación y registro del problema.

Cliente:

- Historia C1: Registro como cliente
  Como visitante, quiero registrarme en el sistema, para acceder a los servicios del parque.
  Entradas: nombre, correo, contraseña, edad, método de pago.
  Resultado esperado: creación exitosa del perfil.
- Historia C2: Autenticarse
  Como cliente, quiero iniciar sesión con mis credenciales, para acceder a las funcionalidades del sistema.
  Entradas: usuario y contraseña.
  Resultado esperado: autenticación correcta y acceso al sistema.
- Historia C3: Consultar atracciones
  Como cliente, quiero ver la lista de atracciones disponibles, para planificar mi visita.
  Entradas: ninguna (requiere autenticación previa).
  Resultado esperado: lista de atracciones con detalles operativos.
- Historia C4: Comprar tiquetes
  Como cliente, quiero adquirir tiquetes para las atracciones, para disfrutar del parque.
  Entradas: atracción seleccionada, método de pago.
  Resultado esperado: confirmación de la compra.

2. Pruebas Automatizadas (JUnit)

Como parte fundamental del desarrollo orientado a calidad, se implementó un conjunto robusto de pruebas automáticas utilizando JUnit 5, que validan la lógica interna del sistema y el cumplimiento de los requerimientos funcionales derivados de las historias de usuario.

Pruebas Unitarias

Las pruebas unitarias se centraron en verificar comportamientos clave:

Gestión de empleados:

- Búsqueda de empleados por ID.
- Validación de empleados inexistentes.
- Edición de datos como lugar de trabajo, puestos, turnos, etc.

Gestión de atracciones:

- Creación de atracciones de tipo mecánica y cultural.
- Edición y eliminación de atracciones.
- Búsqueda de atracciones por código de ubicación.

Estas pruebas garantizan que los métodos centrales del negocio se comporten correctamente ante entradas válidas e inválidas.

Aplicación de TDD

En algunos casos como la edición de empleados o atracciones, se aplicó el enfoque de Desarrollo Guiado por Pruebas (TDD), escribiendo primero los tests antes de codificar la lógica que los satisface. Esto permitió desarrollar funcionalidades con mayor seguridad y control de errores.

Pruebas de Integración

Aunque no se evaluaron las interfaces de consola directamente, como lo establece el enunciado, las pruebas unitarias actúan también como pruebas de integración entre componentes del modelo, como Administrador, Empleado y Atracción, probando la colaboración entre clases.

3. **Interfaces de Consola**

Se implementaron tres interfaces independientes mediante clases main, una para cada tipo de usuario. Estas están organizadas de forma modular y todas integradas en un solo proyecto Eclipse. Cumplen con requisitos clave de autenticación, validación, carga automática de datos y persistencia.

Autenticación

- Todos los usuarios deben iniciar sesión con correo y contraseña.
- Las credenciales se validan según el tipo de usuario (cliente, empleado, administrador) usando la clase Autenticador.
- El sistema insiste hasta que se ingresen credenciales válidas y luego muestra un mensaje de bienvenida personalizado.

Carga y persistencia automática

- Los datos se cargan automáticamente desde archivos preestablecidos al iniciar cada aplicación.
- No se solicita al usuario ingresar rutas ni nombres de archivo.
- Los cambios realizados se guardan de manera automática tras cada operación relevante o al finalizar la sesión.

Consola del Administrador

Opciones disponibles:

- Gestionar empleados
- Gestionar atracciones
- Ver reportes
- Salir

Esta consola permite manipular tanto la plantilla de personal como las atracciones del parque. Está diseñada para expandirse con funcionalidades de análisis y generación de reportes.

Consola del Empleado

Opciones disponibles:

- Registrar asistencia/tareas
- Consultar asignaciones
- Reportar problemas
- Salir

Brinda acceso al personal para llevar el control de su jornada y comunicar incidencias de forma sencilla.

Consola del Cliente

Opciones disponibles:

- Ver atracciones disponibles
- Comprar tiquete
- Ver historial de compras
- Salir

Proporciona una experiencia básica y funcional para los visitantes del parque, con posibilidad de expansión hacia servicios personalizados.

Validación de entradas

- Todas las interfaces validan que las opciones de menú sean numéricas.
- En caso de error, el sistema notifica al usuario y solicita una nueva entrada, evitando interrupciones por errores de formato.

PROYECTO #3: PARQUE DE DIVERSIONES

Objetivo General

El objetivo principal de esta fase fue evolucionar el sistema desde una aplicación basada en consola hacia una aplicación con interfaz gráfica de usuario (GUI) desarrollada con Java Swing, manteniendo y extendiendo la lógica de negocio construida en las fases anteriores.

Principales Mejoras Implementadas

Interfaz Gráfica (Swing)

Se migró toda la interacción con el usuario a una interfaz gráfica, manteniendo las funcionalidades previas e incorporando una experiencia visual más amigable y funcional.

- Ventanas independientes para cada rol del sistema: Administrador, Empleado y Cliente.
- Menús, botones y diálogos diseñados para facilitar la navegación y la interacción.
- Componentes Swing usados: JFrame, JPanel, JButton, JTextField, JComboBox, JList, JSpinner, entre otros.

Impresión de Tiquetes con Código QR

Se agregó una funcionalidad central: generación e impresión de tiquetes con un código QR dinámico, que incluye:

- Tipo de tiquete
- ID del tiquete
- Fecha de expedición en tiempo real
- Generación del QR usando la librería ZXing

Además:

- El sistema detecta si un tiquete ya fue impreso y solicita confirmación para reimpresiones.
- El QR es escaneable desde cualquier cámara de celular.

Registro de Clientes desde la Interfaz

Se añadió una ventana dedicada para permitir que nuevos clientes se registren desde la interfaz, sin requerir modificación manual de archivos.

- Valida que el correo no esté registrado previamente.
- Guarda los datos en el archivo clientesp.txt automáticamente.

Validación y Autenticación por Roles

La autenticación se mantuvo, ahora conectada a través de la interfaz gráfica.

- Cada usuario es redirigido a su ventana correspondiente según su rol.

- Se usan archivos planos (adminsP.txt, empleadosP.txt, clientesp.txt) como base de datos simple.

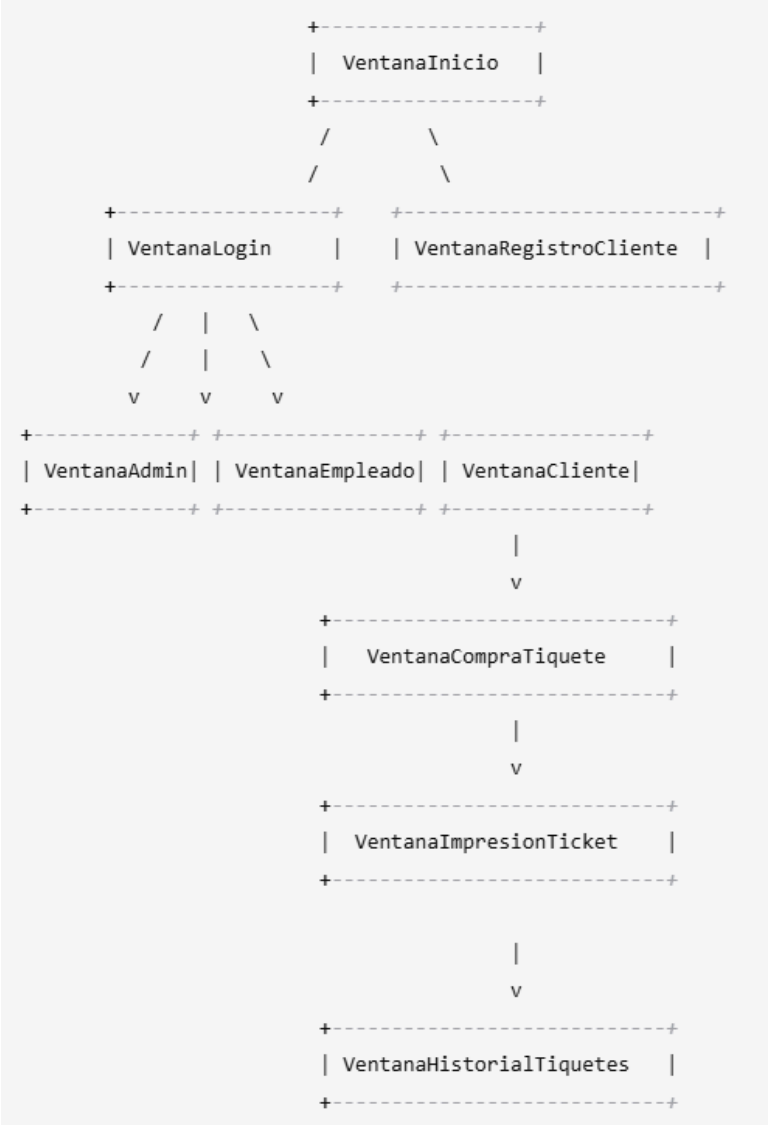
Estructura General del Sistema

La arquitectura del sistema se mantuvo modular, organizando el código en paquetes:

- scr.gui: interfaz gráfica Swing
- scr.personas: clases Usuario, Empleado, Administrador, Cliente
- scr.atracciones: jerarquía de atracciones
- scr.main.Venta: lógica de compra y manejo de tiquetes
- scr.atracciones.logica: autenticación y utilidades de negocio

Diagrama de la Interfaz (GUI)

Se diseñó un diagrama adicional mostrando las ventanas y su relación entre sí, representando los flujos de navegación para cada rol.



VentanaInicio

├──> VentanaLogin

│ ├──> VentanaAdmin (si inicia como administrador)

│ ├──> VentanaEmpleado (si inicia como empleado)

│ └──> VentanaCliente (si inicia como cliente)

│ ├──> VentanaCompraTiquete

│ ├──> VentanaImpresionTicket

│ └──> VentanaHistorialTiquetes

└──> VentanaRegistroCliente

Descripción de cada ventana gráfica (GUI)

Una breve explicación de cada clase de interfaz (.java) y su propósito.

- VentanaInicio: Pantalla de bienvenida donde el usuario elige su rol (Admin, Empleado, Cliente, Registro).
- VentanaLogin: Pantalla de autenticación para todos los roles.
- VentanaAdmin: Panel con opciones para gestionar empleados y atracciones.
- VentanaEmpleado: Vista con información personal y acciones del empleado.
- VentanaCliente: Menú principal del cliente: ver atracciones, comprar tiquete, historial.
- VentanaCompraTiquete: Formulario para seleccionar tipo de tiquete, días y generar uno.
- VentanaImpresionTicket: Vista previa del tiquete con QR y confirmación de impresión.
- VentanaHistorialTiquetes: Lista de tiquetes comprados por el cliente.
- VentanaRegistroCliente: Formulario de registro de nuevos clientes.

Decisiones de Diseño

- Se optó por Java Swing por ser parte del currículo del curso y permitir control completo sobre la GUI.
- Se reutilizó el modelo de dominio existente, lo cual permitió centrarse en construir la interfaz sin rehacer la lógica.
- La autenticación por archivos planos se mantuvo para respetar la simplicidad del sistema y evitar el uso de bases de datos externas.
- Se integró el QR como funcionalidad avanzada utilizando la librería externa ZXing.

Validaciones y Pruebas

- Se probaron todos los flujos de usuario: inicio de sesión, compra, impresión, registro, navegación y salida.
- Se verificó la lectura de QR desde dispositivos móviles.
- Se probó que los archivos .txt se actualizan correctamente con cada acción del usuario.

Conclusiones

Esta tercera fase permitió transformar un sistema funcional pero limitado a consola en una aplicación interactiva y mucho más amigable para el usuario. Se reforzaron habilidades como el diseño de interfaces, integración de librerías externas, modularización del código y uso de patrones de diseño. El sistema quedó listo para futuras extensiones como conexión a base de datos o migración a entornos móviles o web.