

**UNIVERSIDAD TECNOLÓGICA DEL PERÚ**

**ALGORITMOS Y ESTRUCTURA DE DATOS**

**Diseño y desarrollo de un Sistema de Control de Estacionamiento Inteligente para la Empresa PlazaPlus, Lima 2025**

**INTEGRANTES:**

* Garcia Serrato, Angello Fabricio
* Ortiz Angeles, Eduard Frank
* Ramirez Rivera, Adrian Jose Felix
* Sihue Saavedra, Julio Cesar
* Velarde Ochoa, Obed Victor Jesus

**DOCENTE**

CARLOS EFRAIN PALOMINO VIDAL

**SECCION**

20429

**Lima – Perú**

**2025**

Índice

[INTRODUCCIÓN 3](#_i7rxugh4gjuv)

[I.](#_agwalm4z6vn) CAPITULO 1 3

[1.1.](#_epkmzj1inbpc) Aspectos Generales 3

[1.2.](#_u7c0cxcdalww) Organigrama 3

[1.3.](#_gissra76n2bn) Misión y Visión 4

[1.4.](#_ogbmz2ilodyy) Objetivos Estratégicos 4

[1.5.](#_oipdj1o7amp0) Problemática 5

[1.6.](#_l91m356w3i6q) Alternativas de Solución 5

[1.7.](#_7tf4z1mjjzmu) Solución Elegida 6

[II.](#_7n8t2txh48fc) CAPITULO 2 7

[2.1](#_5bzhens1iemh) Estado del Arte 7

[2.2](#_tovwk9nm1ygh) Marco Teórico 9

[III.](#_io1t8fvsesi3) CAPITULO 3 14

[3.1](#_a666mgstbs) Alcance 14

[3.2](#_17l7ldewmbd4) Requerimientos Funcionales y No Funcionales 15

[3.3](#_fa2ywpaod37i) Restricciones 18

[3.4](#_l6nyxvl5nq0) Diagrama de Clases 18

[Ilustración 1 18](#_xh33sunxy6ay)

[3.4.1 Vista Por Paquetes 18](#_o4mpgjcnmso3)

[Ilustración 2 19](#_avo3h5c4aml7)

[Ilustración 3 20](#_crysq9flfi51)

[Ilustración 4 21](#_oickrcoar89p)

[Ilustración 5 22](#_vdh9ok6ju83n)

[Paquete Controlador 23](#_9tempxjtx5ax)

[Ilustración 6 23](#_925f8w9ye39a)

[Paquete Vista 24](#_x60vxfrnidxg)

[Ilustracion 7 24](#_ty5a8tpuji8)

[Ilustracion 8 24](#_ty5a8tpuji8)

[3.5](#_dq8j0l8t60py) Prototipo o Interfaces 25

[IV.](#_b6s3phek7exf) CAPITULO 4 37

[4.1](#_nfcbhn41ycaf) Diseño de la Aplicación 37

[4.2](#_kza90kqcinfe) Clases usadas en la aplicación 42

[4.3](#_5vaggdhg3tmm) Codigo Fuente 45

[4.3.1](#_h1aoxcbda1q4) Paquete Modelo 45

[V.](#_vun5jk2gg28k) Bibliografía y Anexo 54

## INTRODUCCIÓN

Como integrantes de un equipo de consultores de sistemas, hemos sido convocados por la empresa PlazaPlus para diseñar y desarrollar un sistema que automatice la gestión de sus estacionamientos. Actualmente, la asignación desordenada de plazas, los cobros manuales inconsistentes y la falta de control en los ingresos y salidas afectan tanto la eficiencia operativa como la experiencia de los clientes.

Con el fin de dar solución a estas problemáticas, se plantea la construcción de un aplicativo de escritorio en Java, que integre estructuras de datos y algoritmos para el control inteligente del estacionamiento. El sistema seguirá un enfoque iterativo e incremental, garantizando entregas continuas, escalabilidad y mejoras progresivas en beneficio de la empresa y sus usuarios.

# CAPITULO 1

## Aspectos Generales

La empresa PlazaPlus es una organización dedicada a la gestión de estacionamientos en centros comerciales de Lima. Su objetivo principal es brindar a sus clientes un servicio de estacionamiento cómodo, seguro y eficiente. Actualmente, la empresa enfrenta dificultades relacionadas con la supervisión de ingresos y salidas, lo que ocasiona demoras y falta de control en el proceso.

Para responder a estas necesidades, se plantea el desarrollo del proyecto “Diseño y desarrollo de un Sistema de Control de Estacionamiento Inteligente para la Empresa PlazaPlus, Lima 2025”, el cual busca automatizar la gestión de plazas, optimizar los tiempos de atención y garantizar transparencia en los procesos administrativos.

## Organigrama

La Figura 1 representa la estructura jerárquica y funcional de la empresa PlazaPlus, proporcionando una visión general de las áreas que intervienen en la gestión operativa y administrativa del estacionamiento. Esta representación es fundamental para comprender cómo se distribuyen las responsabilidades y flujos de información dentro de la organización, lo que permite diseñar un sistema informático alineado con su realidad empresarial.

La incorporación del organigrama permite identificar claramente los puntos de interacción entre el sistema y los distintos departamentos, asegurando que el flujo de datos sea coherente con la dinámica organizacional.

Figura 1

Organigrama funcional de la empresa PlazaPlus

Diagrama

El contenido generado por IA puede ser incorrecto.

Nota. - Elaboración propia,2025.

## Misión y Visión

Misión. - Ofrecer una experiencia de estacionamiento segura, eficiente y automatizada, garantizando comodidad y confianza a los clientes mediante el uso de tecnologías inteligentes y accesibles.

Visión. - Consolidarse hacia el año 2030 como la empresa líder en gestión inteligente de estacionamientos en Lima, destacando por su innovación, sostenibilidad y calidad en el servicio al cliente.

## Objetivos Estratégicos

* Optimizar la gestión de los procesos internos del estacionamiento mediante un sistema automatizado que integre el control de plazas, ingresos, salidas, cobros y reportes.
* Reducir los errores humanos eliminando la duplicidad y falta de control generada por los procesos manuales
* Mejorar la generación de reportes e indicadores de gestión, facilitando la auditoría, el control administrativo y la toma de decisiones estratégicas basadas en datos.
* Fortalecer la eficiencia del personal operativo y administrativo, proporcionándoles herramientas digitales que agilicen sus tareas y aumenten la satisfacción de los clientes.

## Problemática

En la actualidad, los estacionamientos de PlazaPlus enfrentan problemas relacionados con la asignación desordenada de plazas, lo que ocasiona demoras y molestias a los clientes. Además, los cobros manuales son inconsistentes, generando pérdidas económicas por errores humanos o falta de control.

La falta de visibilidad en los ingresos y salidas de vehículos dificulta la gestión administrativa y la toma de decisiones estratégicas. Estas limitaciones afectan directamente la experiencia del cliente, reducen la eficiencia operativa y disminuyen la competitividad de la empresa frente a alternativas más modernas de la competencia.

Frente a ello, PlazaPlus requiere un sistema automatizado que permita ordenar el flujo de vehículos, garantizar cobros justos y obtener información en tiempo real para la toma de decisiones.

## Alternativas de Solución

Se plantean tres alternativas tecnológicas para atender la problemática identificada en la empresa PlazaPlus. La comparación se realiza considerando criterios de costo, tiempo, confiabilidad, escalabilidad y adecuación académica

La evaluación de las opciones de desarrollo fue crucial. La comparación detallada de las alternativas se presenta en la siguiente Tabla.

## Tabla 1

Alternativas de Solución para la Problemática de PlazaPlus

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITERIO** | **ALTERNATIVA A:**  **Control Manual Digitalizado** | **ALTERNATIVA B: Sistema con Sensores** **IoT + App Móvil** | **ALTERNATIVA C: Aplicación de Escritorio con Algoritmos de Control (Propuesta)** |
| **Descripción** | Uso de planillas digitales (Excel/Access) para registrar ingresos y salidas. | Colocación de sensores físicos en cada plaza, conectados a una app móvil. | Desarrollo de un sistema de escritorio en Java que gestiona plazas, cobros y reportes mediante algoritmos. |
| **Costo de Implementación** | Bajo | Alto (hardware + infraestructura IoT) | Medio (software desarrollado en NetBeans con BD local/servidor). |
| **Tiempo de Desarrollo** | Corto (1-2 meses) | Largo (6-12 meses) | Medio (3-4 meses de desarrollo académico). |
| **Confiabilidad** | Baja, depende del operador | Alta, con detección en tiempo real | Alta, con control automatizado mediante algoritmos. |
| **Escalabilidad** | Limitada | Alta, con conexión a la nube | Media-Alta, adaptable a múltiples pisos y plazas |
| **Adecuación Académica** | Muy baja (no usa estructuras de datos avanzadas). | Baja (se centra más en hardware que en algoritmos). | Muy alta (implementa colas, pilas, árboles, búsquedas y ordenamientos). |
| **Beneficio** Principal | Digitaliza parcialmente el proceso. | Automatiza totalmente con hardware avanzado. | Automatiza el proceso de forma académica y funcional, con resultados medibles. |

Tabla 1 Elaboración propia,2025.

## Solución Elegida

Tras la evaluación comparativa, se selecciona la Alternativa C: Aplicación de Escritorio con Algoritmos de Control.

Esta solución permite automatizar los procesos clave de PlazaPlus mediante un sistema desarrollado en Java (NetBeans IDE 21), utilizando estructuras de datos y algoritmos que garantizan un desempeño confiable y escalable. Entre sus principales virtudes destacan:

* Uso de estructuras de datos avanzadas: colas para gestionar ingresos y salidas, pilas para auditoría de movimientos y árboles binarios para búsqueda eficiente de plazas.
* Generación de reportes inteligentes sobre ocupación, tiempos de permanencia y flujo de vehículos.
* Mapa visual del estacionamiento, organizado por pisos y sectores.
* Escalabilidad académica y empresarial, con posibilidad de integración futura con sistemas web o móviles.

De esta manera, PlazaPlus podrá resolver sus problemas actuales, mejorar la experiencia del cliente y contar con una solución tecnológica alineada tanto a sus necesidades empresariales como a las competencias académicas desarrolladas en el curso de Algoritmos y Estructuras de Datos.

## CAPITULO 2

## Estado del Arte

La gestión automatizada de estacionamientos ha evolucionado significativamente en la última década gracias a los avances en tecnologías de información, sensores inteligentes y algoritmos de optimización. A nivel mundial, las soluciones modernas de Smart Parking Systems buscan mejorar la eficiencia del uso de plazas, reducir el tráfico interno y ofrecer mayor comodidad a los usuarios mediante aplicaciones o sistemas automatizados (Gupta et al., 2021).

En América Latina, países como México, Chile y Colombia han implementado proyectos de estacionamientos inteligentes en centros comerciales y aeropuertos, integrando sensores, cámaras y sistemas informáticos que permiten conocer en tiempo real la disponibilidad de plazas (Cruz & Fernández, 2020). Sin embargo, estas soluciones suelen requerir una infraestructura tecnológica costosa y mantenimiento especializado, lo cual limita su adopción por parte de empresas medianas o locales.

En el contexto peruano, la mayoría de los estacionamientos comerciales continúan gestionando sus operaciones de forma manual o semi-digital, utilizando planillas de cálculo y registros físicos. Esta situación genera pérdida de información, errores en los cobros y demoras en el control de entradas y salidas (Pacheco, 2022). Algunas iniciativas privadas, como los sistemas implementados en centros comerciales de Lima Norte, han comenzado a incorporar software de registro digital, pero sin una integración total ni algoritmos de control automatizado.

Frente a este panorama, el desarrollo de una aplicación de escritorio en Java representa una alternativa eficiente, económica y adaptable a la realidad local. Esta opción permite automatizar la gestión de plazas, ingresos y salidas de vehículos mediante el uso de estructuras de datos y algoritmos de búsqueda y ordenamiento, sin depender de hardware externo o conexión en la nube.

Como menciona Sedgewick (2011), los algoritmos y estructuras de datos son la base para construir sistemas eficientes que optimicen recursos y minimicen tiempos de respuesta. Aplicar estos fundamentos a la gestión de estacionamientos permite emular comportamientos inteligentes —como la asignación automática de plazas o la priorización de turnos— de manera puramente lógica y programática.

En este sentido, el sistema PlazaPlus propone una solución académica y práctica, basada en la ingeniería de software y el análisis de algoritmos, que permite a las empresas gestionar sus operaciones con precisión y escalabilidad. Este enfoque no solo mejora el control operativo, sino que también contribuye a la modernización digital del sector de estacionamientos en Lima.

## Marco Teórico

El presente sistema se fundamenta en tres pilares técnicos: el patrón de diseño Modelo-Vista-Controlador (MVC), el uso de estructuras de datos dinámicas (listas enlazadas, colas, pilas y árboles binarios) y el empleo de algoritmos de búsqueda y ordenamiento para la eficiencia del sistema.

El desarrollo del Sistema de Control de Estacionamiento Inteligente para PlazaPlus trasciende la mera programación funcional. Constituye una demostración práctica y rigurosa de los principios cardinales de la Ingeniería de Software y la Ciencia de la Computación. Como bien afirma Joyanes (2016), “El estudio de Algoritmos y de Estructuras de Datos son disciplinas académicas que se incorporan a todos los planes de estudios universitarios” (p. 21). Este capítulo establece la fundamentación epistemológica y técnica que justifica la selección de Estructuras de Datos Abstractas (TAD) y algoritmos como pilares centrales para garantizar la eficiencia operativa, la integridad de los datos y, crucialmente, la escalabilidad asintótica del sistema.

La correcta elección e implementación de una estructura de datos es una decisión de diseño crítica que impacta directamente en el rendimiento temporal (O(n)) y espacial del software. Al modelar el flujo de un estacionamiento, transformamos la problemática empírica de la gestión de vehículos en un problema formalmente computable y optimizado mediante la aplicación de estos constructos teóricos.

**Estructuras de Datos en aplicaciones clínicas.**

**Modelando el Flujo Dinámico**

Las estructuras de datos lineales son esenciales para modelar secuencias de eventos donde existe una relación de precedencia clara entre los elementos. Su simplicidad conceptual permite simular procesos reales de manera secuencial y determinística.

**Listas Enlazadas. La Arquitectura Dinámica de la Memoria**

La Lista Enlazada se posiciona como una alternativa superior a las matrices o arrays estáticos, cuya ineficiencia en la inserción o eliminación de datos en posiciones intermedias se traduce en una complejidad de O(n) debido a la necesidad de desplazar elementos contiguos.

**Detalle de Procesos**

La arquitectura nodular de la Lista Enlazada —donde cada elemento o nodo contiene el dato y una referencia (puntero) al siguiente nodo— libera al sistema de la necesidad de contigüidad física en memoria. Al respecto, la Universidad Veracruzana (s.f.) indica que “Una lista enlazada es una colección lineal de elementos llamados nodos. El orden entre ellos se establece mediante punteros; direcciones o referencias a otros nodos”.

Las operaciones de inserción y eliminación en los extremos se ejecutan mediante la simple reorganización de enlaces en un tiempo asintótico constante (O(1)). Esta eficiencia es el resultado de que el proceso solo requiere actualizar los punteros del nodo anterior y del nuevo nodo, independientemente del tamaño total de la lista.

**Conexión con PlazaPlus**

Esta propiedad de eficiencia O(1) es crítica para la alta disponibilidad y la rapidez de respuesta en el sistema. Se utiliza específicamente en la implementación del UsuariosDAO y el control de accesos, garantizando que el registro de una nueva cuenta o la eliminación de una entrada se realice sin latencia, incluso con un crecimiento dinámico de los datos.

**Colas**

Una Cola (Queue) es una estructura de datos abstracta que sigue estrictamente el principio FIFO (First-In, First-Out). Desde una perspectiva de ingeniería, esta estructura representa el modelo ideal para la gestión de flujos de atención, donde el orden de llegada es prioritario.

Detalle de Procesos

La funcionalidad de la Cola se define mediante dos operaciones canónicas:

* Encolar: Proceso de inserción de un nuevo elemento en la cola (parte final) de la estructura.
* Desencolar: Proceso de extracción del elemento que se encuentra en la cabeza (parte frontal) de la estructura.

Cuando estas operaciones se optimizan, mantienen una eficiencia óptima de O(1).

Conexión con PlazaPlus

En el sistema, la Cola se utiliza para modelar la línea de espera en el punto de acceso (garita de ingreso) del estacionamiento. Esta implementación garantiza la equidad algorítmica en el servicio, procesando a los vehículos en el orden exacto de su llegada y asegurando un flujo de atención determinado y justo.

**Pilas**

Una Pila (Stack) es una estructura de datos abstracta que opera bajo el principio LIFO (Last-In, First-Out). Esta naturaleza es inherentemente útil para modelar secuencias recursivas y procesos de memoria de corto plazo.

Detalle de Procesos

Las operaciones fundamentales de la Pila son:

* Push (Apilar): Proceso de inserción de un nuevo elemento en la cima de la Pila.
* Pop (Desapilar): Proceso de extracción del elemento que se encuentra actualmente en la cima de la Pila.

Ambas operaciones mantienen una complejidad de O(1) en una implementación eficiente.

**Conexión con PlazaPlus**

En este sistema, la Pila se aplica estratégicamente para la auditoría y trazabilidad de transacciones críticas. Al registrar los últimos movimientos (ingresos, pagos, cambios de estado) en una Pila, se facilita la revisión inmediata y secuencialmente inversa de los eventos, cumpliendo con los requisitos de reversibilidad y trazabilidad.

**Estructuras de Datos No Lineales: Optimización de la Búsqueda**

Para manejar la complejidad y el volumen creciente de datos de las plazas de estacionamiento, la estructura debe romper con la linealidad de la memoria, adoptando una organización jerárquica para optimizar el tiempo de acceso.

**Árboles Binarios de Búsqueda (BST)**

Es una estructura de datos jerárquica diseñada para optimizar las operaciones de búsqueda en grandes volúmenes de datos.

Detalle de Procesos

Un BST se rige por la Propiedad del Árbol Binario de Búsqueda, que establece un orden intrínseco: para cualquier nodo, todos los valores en su subárbol izquierdo son menores, y todos los valores en su subárbol derecho son mayores.

Esta organización permite que las operaciones de búsqueda, inserción y eliminación se realicen con una complejidad de O(\log n) (logarítmica) en el caso promedio. Este desempeño es superior al de una lista simple. Es crucial señalar que, si el árbol no se balancea, la complejidad puede degradarse a O(n).

Conexión con PlazaPlus

El BST es vital para la Gestión de Plazas de Estacionamiento. El sistema utiliza esta estructura para representar la jerarquía de plazas por nivel y disponibilidad. El rendimiento de O(\log n) es esencial para que la consulta de la disponibilidad de una plaza específica se realice de manera instantánea, un requisito de alto rendimiento para el control de tráfico en tiempo real.

**Algoritmos de Búsqueda y Ordenamiento**

Dentro del paquete modelo. algoritmos, las clases AlgoritmosBusqueda y AlgoritmosOrdenamiento implementan procedimientos clásicos como búsqueda binaria, secuencial, ordenamiento burbuja, selección e inserción, aplicados a listas de usuarios, vehículos y tickets.

Estos algoritmos mejoran la eficiencia en consultas y reportes, garantizando que el sistema mantenga tiempos de respuesta óptimos incluso con un alto número de registros.

**Interfaz Gráfica y Experiencia de Usuario**

Las interfaces desarrolladas con Java Swing ofrecen paneles amigables y componentes reutilizables para cada módulo del sistema (login, administrador, reportes). Se incluyen elementos gráficos personalizados (Encabezado, PieDePagina, FormIngreso, FormEgreso), además de recursos visuales (PlazaPlus-logo.png) que refuerzan la identidad visual del sistema.

De acuerdo con Deitel & Deitel (2011), el uso de Swing proporciona flexibilidad en la creación de interfaces gráficas robustas, con soporte para eventos, manejo de formularios y validación de entradas.

**Serialización y Persistencia**

Aunque el sistema no utiliza un gestor de base de datos externo, maneja persistencia local mediante serialización de objetos, permitiendo guardar el estado actual de listas, colas y árboles en archivos binarios. Este enfoque combina simplicidad con confiabilidad para entornos académicos o de ejecución local.

## CAPITULO 3

## Alcance

El presente proyecto tiene como propósito diseñar e implementar un sistema de escritorio en Java que permita a la empresa PlazaPlus gestionar de forma automatizada las operaciones de su estacionamiento, abarcando desde el registro de vehículos hasta la generación de reportes administrativos.

El sistema, desarrollado bajo el patrón Modelo-Vista-Controlador (MVC), contempla el uso de estructuras de datos dinámicas y algoritmos de búsqueda y ordenamiento para garantizar un manejo eficiente de la información sin depender de bases de datos externas.

**Funciones principales del sistema.**

* Registro, control y liberación de plazas de estacionamiento organizadas por pisos.
* Ingreso y salida de vehículos, controlado por colas y listas enlazadas.
* Generación de tickets de ingreso y salida mediante objetos serializados.
* Gestión de usuarios y roles (administrador).
* Emisión de reportes de ocupación, ingresos económicos y flujo vehicular diario.
* Visualización gráfica del estado del estacionamiento en tiempo real.

**Resultados esperados**

* Reducir los errores humanos en la asignación de plazas y cobros.
* Optimizar el tiempo de atención al cliente.
* Centralizar la información de operaciones.
* Aumentar la trazabilidad y control interno.
* Sentar las bases para una futura integración con sistemas móviles o web.

## Requerimientos Funcionales y No Funcionales

Requerimientos Funcionales

**1. Gestión de plazas de estacionamiento:**

El sistema permitirá registrar, asignar, liberar y visualizar plazas por piso, empleando estructuras tipo árbol binario (ArbolBinario) para representar la jerarquía y disponibilidad.

**2. Registro y control de vehículos:**

Se gestionarán los datos de los vehículos que ingresan y salen del estacionamiento. Las operaciones de entrada y salida se manejan mediante colas (FIFO) (Cola), asegurando la atención en orden de llegada.

**3. Emisión de tickets:**

Cada ingreso genera un objeto Ticket que almacena información del vehículo, hora de ingreso, salida.

**4. Gestión de usuarios:**

Los usuarios son administrados mediante el UsuarioDAO, utilizando listas enlazadas (ListaEnlazada) para el registro dinámico de cuentas.

**5. Auditoría de movimientos:**

Las operaciones más recientes (por ejemplo, egresos anulados o movimientos revertidos) se almacenan en una pila (LIFO) (Pila), permitiendo deshacer acciones recientes y mantener trazabilidad.

**6. Generación de reportes:**

El GeneradorPDF genera el formato que se usa en ventana reportes para que este produzca estadísticas sobre el historial de ingresos.

**7. Interfaz gráfica de usuario (GUI):**

Desarrollada con Java Swing, incluye módulos como:

* LoginVista → acceso y autenticación de usuarios.
* VentanaPrincipal → panel central con acceso a todas las funciones.
* VentanaAccesoAdministrador→ vistas general admin
* VentanaReportes → visualización de estadísticas y reportes.

**Requerimientos No funcionales**

**1. Ejecución local:**

El sistema funcionará como una aplicación de escritorio ejecutable con JDK 24, sin requerir conexión a internet.

**2. Independencia de bases de datos:**

No depende de un gestor relacional. Toda la información se almacena en memoria mediante estructuras dinámicas y archivos serializados.

**3. Usabilidad:**

La interfaz debe ser intuitiva y clara, diseñada para ser utilizada por personal administrativo con conocimientos básicos de informática.

**4. Tiempo de respuesta:**

Las operaciones de búsqueda y actualización deben ejecutarse en menos de 0.3 segundos, garantizando fluidez en la interacción del usuario.

**5. Portabilidad:**

Compatible con sistemas operativos Windows, Linux y macOS, siempre que tengan instalado el entorno Java.

**6. Mantenibilidad:**

El código está organizado por paquetes y clases documentadas, permitiendo la incorporación de nuevas funciones sin afectar las existentes.

**7. Seguridad:**

El acceso está protegido mediante validación de credenciales en el módulo de inicio de sesión (LoginVista), diferenciando roles de administrador y usuario

## Restricciones

* El sistema no utiliza sensores físicos ni hardware externo, por lo que la detección de vehículos es lógica (simulada).
* No opera en entornos multiusuario ni red local, su ejecución es monousuario y local.
* La persistencia de datos se basa en serialización de objetos, limitada a volúmenes medianos de información.
* Los tipos de vehículos deben configurarse manualmente en el código o a través de formularios internos.
* El sistema no está integrado con plataformas de pago en línea ni bases de datos en la nube.

## Diagrama de Clases

**Vista General:**

La **Ilustración 1** presenta una vista integral de la arquitectura del sistema, donde se representan los tres paquetes principales: *Vista*, *Controlador* y *Modelo*. En esta figura se observa cómo las interfaces gráficas del paquete *Vista* interactúan con los controladores responsables de gestionar el flujo lógico del sistema, mientras que estos, a su vez, se comunican con las clases del *Modelo*, encargadas del manejo y persistencia de los datos. Esta representación global permite comprender el recorrido completo de la información, así como la distribución de responsabilidades entre los diferentes módulos. En conjunto, el diagrama proporciona una perspectiva clara y estructurada del funcionamiento interno del sistema, constituyendo una base sólida para su análisis, desarrollo y futura expansión.

## 

## Ilustración 1

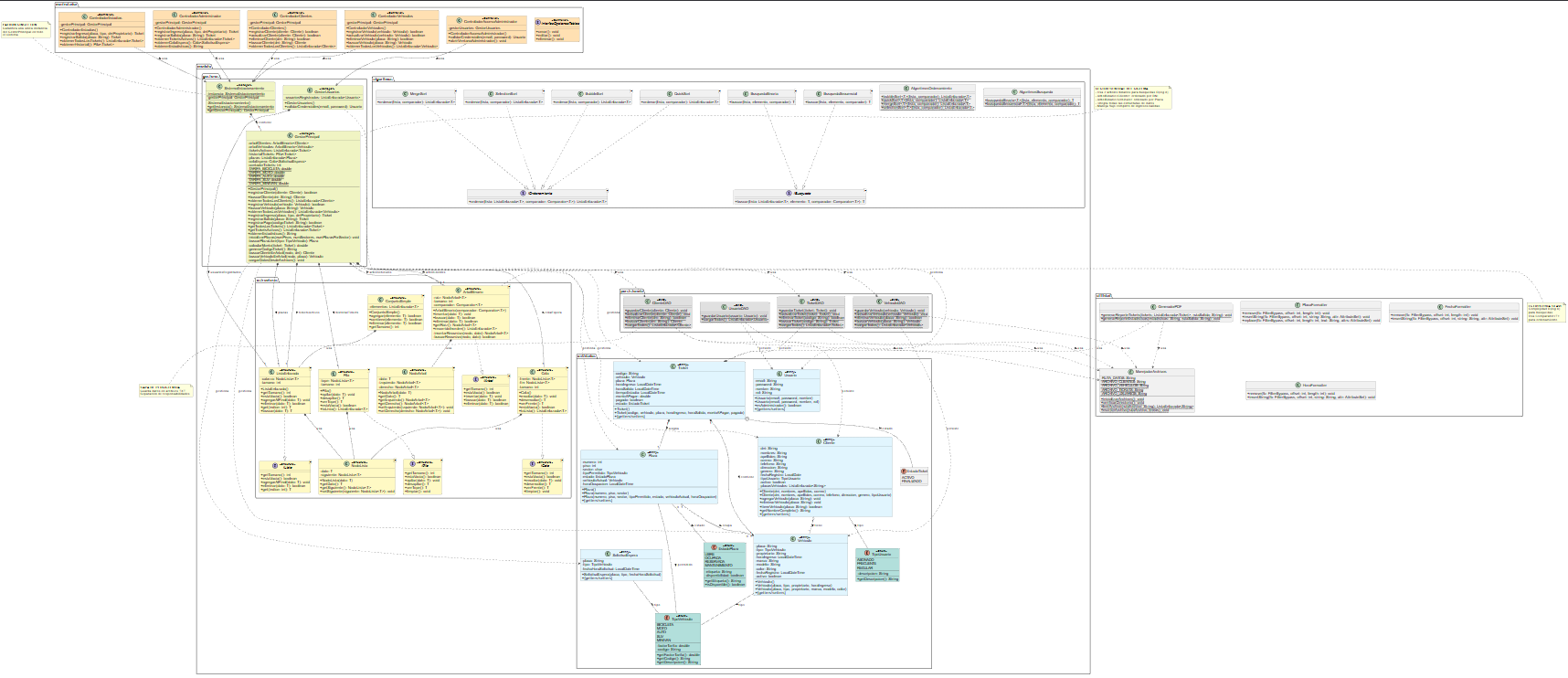


Ilustración 1 Elaboración propia,2025.

## 3.4.1 Vista Por Paquetes

**Paquete Modelo**

Este paquete se utiliza para definir la estructura del sistema, ya que contiene todas las entidades de negocio y las estructuras de datos que soportan la operación de PlazaPlus.  
Incluye clases como Usuario, Vehículo, Plaza y Ticket, además de estructuras como listas, colas, pilas y árboles binarios.  
Su función principal es representar y organizar los datos, asegurando que el sistema sea coherente, eficiente y escalable.

## Ilustración 2

La Ilustración 2 muestra el Diagrama de Entidades de Negocio que estructura el modelo central de PlazaPlus, definiendo los objetos y relaciones que permiten la gestión del estacionamiento. Se incluyen entidades como Cliente, que administra su información personal y los vehículos que posee mediante su relación con Vehículo, clasificado por TipoVehiculo. La entidad Plaza representa la infraestructura física, regulando su disponibilidad mediante EstadoPlaza y permitiendo tipos específicos de vehículos. Además, SolicitudEspera registra solicitudes asociadas a una plaza cuando no hay disponibilidad inmediata. La entidad Ticket integra la ocupación de una plaza, los tiempos de uso y su estado mediante EstadoTicket, vinculando al vehículo correspondiente. Finalmente, Usuario gestiona la autenticación y las operaciones administrativas del sistema. En conjunto, estas entidades forman la base del flujo de datos y control que soporta el funcionamiento de PlazaPlus.

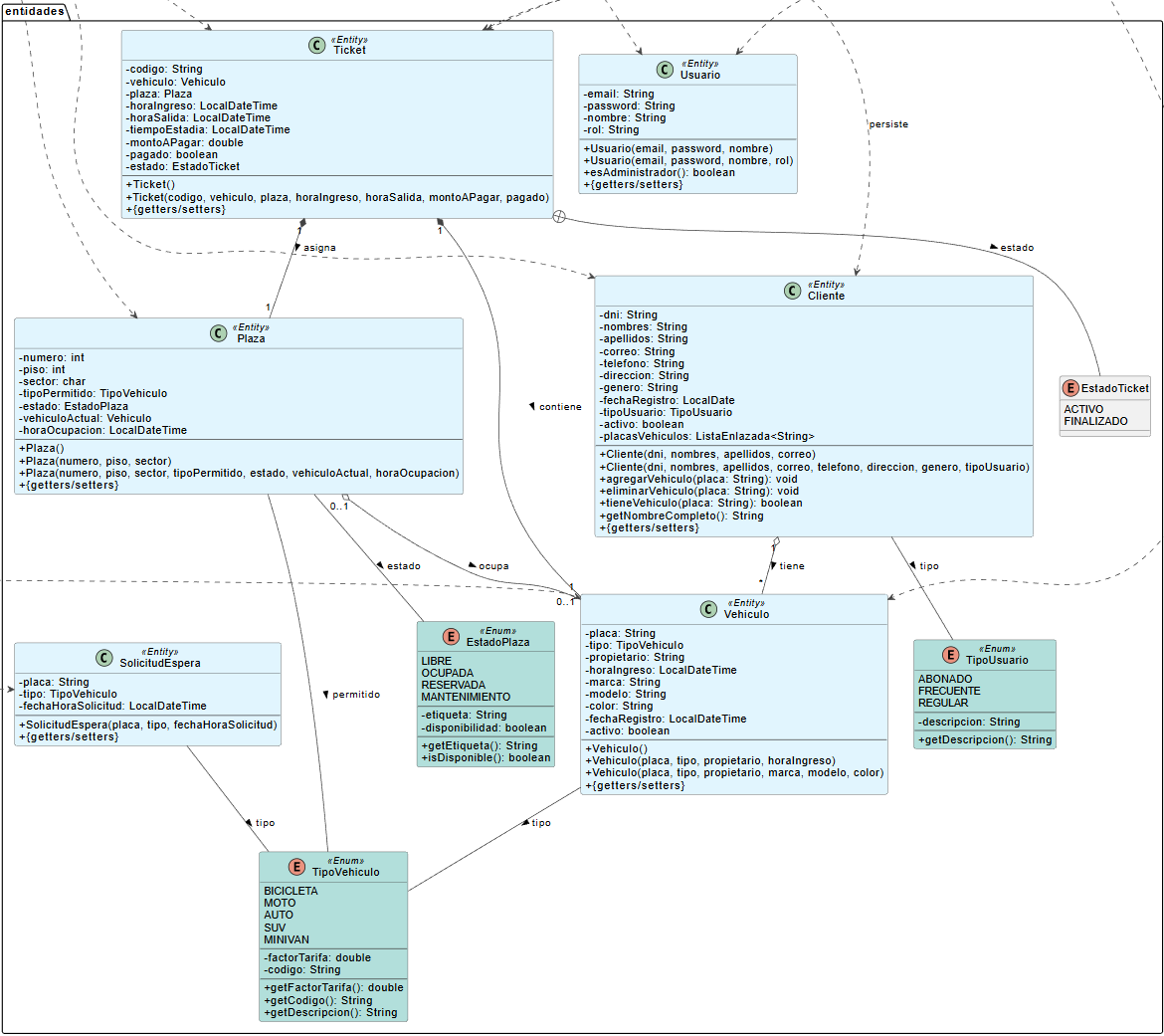


Ilustración 2 Elaboración propia,2025.

## Ilustración 3

## La Ilustración 3 muestra el paquete de Persistencia de PlazaPlus, compuesto por los objetos DAO responsables del acceso y gestión de datos de las entidades principales. Cada DAO implementa operaciones básicas como guardar, actualizar, eliminar y buscar registros, garantizando una separación clara entre la lógica de negocio y el manejo de almacenamiento. ClienteDAO, UsuarioDAO, TicketDAO y VehiculoDAO permiten mantener la integridad de la información y facilitan la comunicación entre el sistema y su capa de datos.

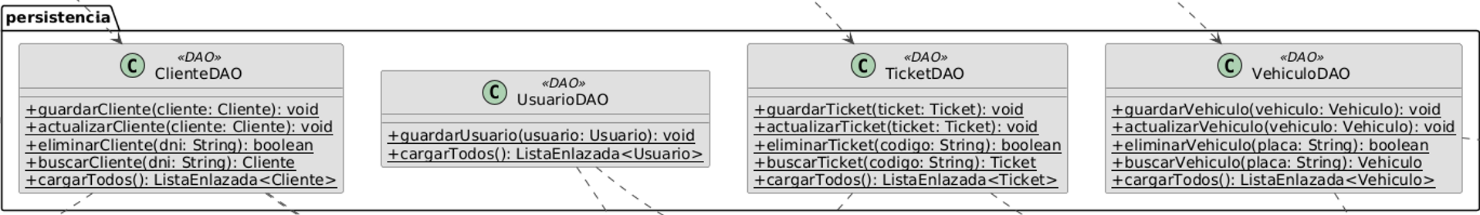


Ilustración 3 Elaboración propia,2025.

## Ilustración 4

La Ilustración 4 muestra el conjunto de estructuras de datos personalizadas empleadas en PlazaPlus, incluyendo las interfaces ILista, IPila e ICola, y sus respectivas implementaciones mediante clases como ListaEnlazada, Pila, Cola y ConjuntoSimple, todas basadas en nodos dinámicos para optimizar la gestión de elementos. Estas estructuras reemplazan colecciones genéricas por versiones adaptadas al dominio del sistema, permitiendo controlar su comportamiento y complejidad. También se incorpora un ÁrbolBinario, construido a partir de NodoArbol, el cual ofrece operaciones recursivas de inserción y búsqueda adecuadas para organizar datos jerárquicos de forma eficiente. En conjunto, estas implementaciones proporcionan un soporte estructural flexible y eficiente, garantizando que las operaciones internas del sistema mantengan un rendimiento acorde a las necesidades de PlazaPlus.

## Ilustración 4 Elaboración propia,2025.

## Ilustración 5

La Ilustración 5 presenta las clases del paquete algoritmos, encargadas de abstraer y unificar los procesos de ordenamiento y búsqueda utilizados en PlazaPlus. A través de las interfaces IOrdenamiento e IBusqueda, el sistema desacopla la lógica algorítmica y permite aplicar distintas estrategias según la necesidad, incluyendo implementaciones como MergeSort, SelectionSort, BubbleSort y QuickSort, todas orientadas a operar sobre listas enlazadas comparables. En cuanto a la recuperación de datos, las clases BusquedaSecuencial y BusquedaBinaria implementan métodos especializados para explorar colecciones lineales u ordenadas, respectivamente. Finalmente, AlgoritmosOrdenamiento y AlgoritmosBusqueda actúan como gestores que centralizan la ejecución de estas técnicas, garantizando consultas eficientes y facilitando la escalabilidad en la generación de reportes y operaciones internas del sistema.

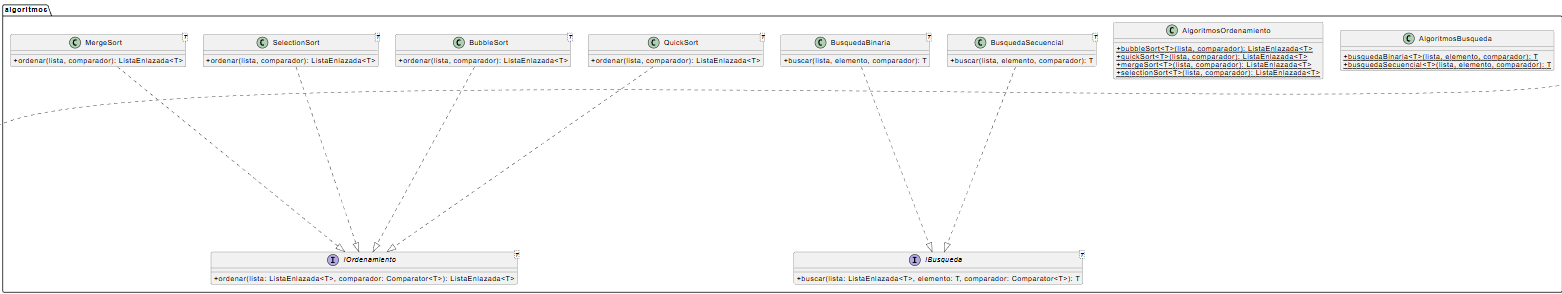


Ilustración 5 Elaboración propia,2025.

## 

## Ilustración 6

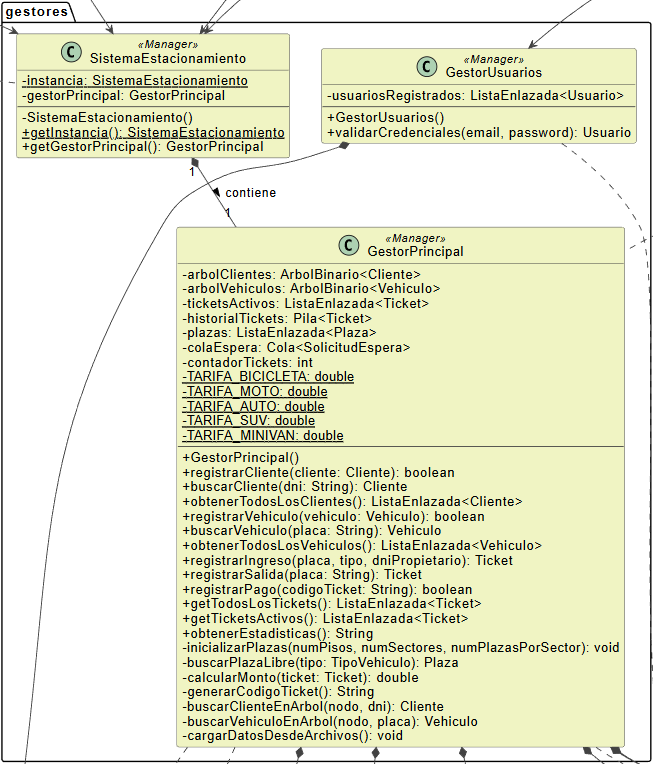
La Ilustración 6 representa la capa de gestores del sistema, donde se concentra la lógica de negocio que coordina las operaciones principales de PlazaPlus. El SistemaEstacionamiento aplica el patrón *Singleton* para garantizar una única instancia global y proveer acceso centralizado al GestorPrincipal, quien administra estructuras como los árboles de clientes y vehículos, la lista de plazas disponibles, los tickets activos y el historial asociado. Este gestor ejecuta funciones clave como el registro de clientes y vehículos, la asignación de espacios mediante *buscarPlazaLibre*, el control de ingresos y salidas, el cálculo de montos y la generación de estadísticas. Paralelamente, el GestorUsuarios gestiona los usuarios del sistema y valida sus credenciales para garantizar un acceso seguro. En conjunto, estos componentes aseguran una organización modular, cohesión funcional y un control eficiente de las operaciones internas del estacionamiento.

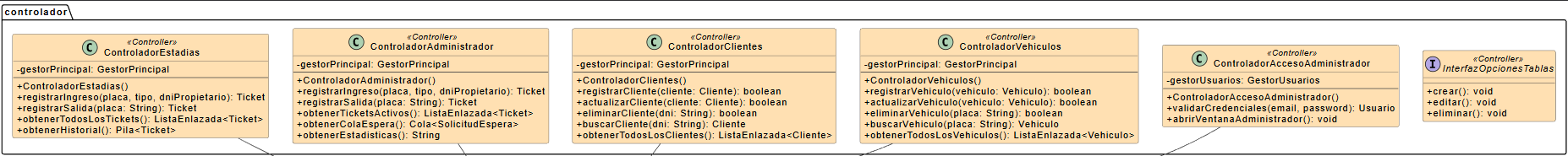
Ilustración 6 Elaboración propia,2025.

## Paquete Controlador

Este paquete se utiliza para coordinar el flujo entre la interfaz (Vista) y la lógica del sistema (Modelo–Negocio).  
Implementa controladores como ControladorLogin y ControladorPrincipal, encargados de recibir acciones del usuario, validarlas y delegarlas a los gestores correspondientes.  
Su propósito es mantener la separación MVC, asegurando orden, modularidad y fácil mantenimiento.

## Ilustración 7

La Ilustración 7 muestra las clases del paquete *Controlador*, las cuales coordinan el flujo del sistema siguiendo el patrón MVC. Cada controlador cumple una función específica: *ControladorAccesoAdministrador* maneja la autenticación mediante el *GestorUsuarios*, mientras que *ControladorEstadias*, *ControladorClientes*, *ControladorVehiculos* y *ControladorAdministrador* gestionan las operaciones del sistema según su ámbito. Todas estas clases delegan la lógica de negocio al *GestorPrincipal*, garantizando una separación clara de funciones y una estructura modular.

Ilustración 7 Elaboración propia, 2025.

## 

## 

## Paquete Vista

El paquete Vista agrupa todas las interfaces graficas responsables de mostrar la información al usuario y capturar sus interacciones.

## Ilustración 8

La Ilustración 8 muestra las clases del paquete *Vista*, encargadas de gestionar la interacción visual con el usuario mediante ventanas, paneles y componentes gráficos. Este paquete incluye interfaces iniciales como *VentanaInicio* y *VentanaAccesoAdministrador*, que dan paso a *VentanaAdministrador*, donde se organizan módulos especializados como *PanelVehiculos*, *PanelUsuarios*, *PanelIngresos* y otros paneles operativos. Además, incorpora elementos transversales definidos en *EstiloComponentes*, que unifican la apariencia y comportamiento de los componentes. En conjunto, este paquete asegura una navegación intuitiva y una presentación coherente de la información procesada por el sistema.

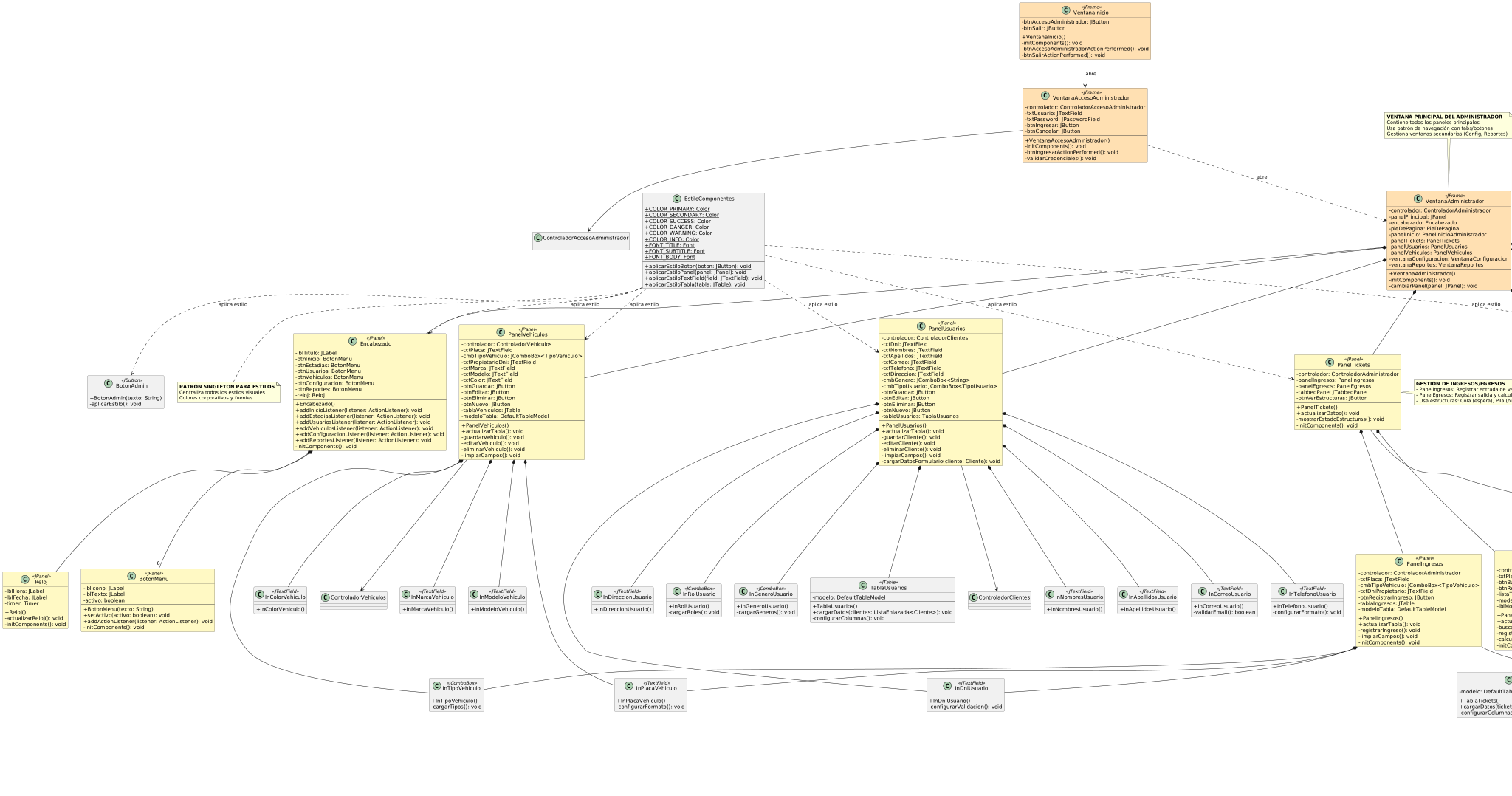
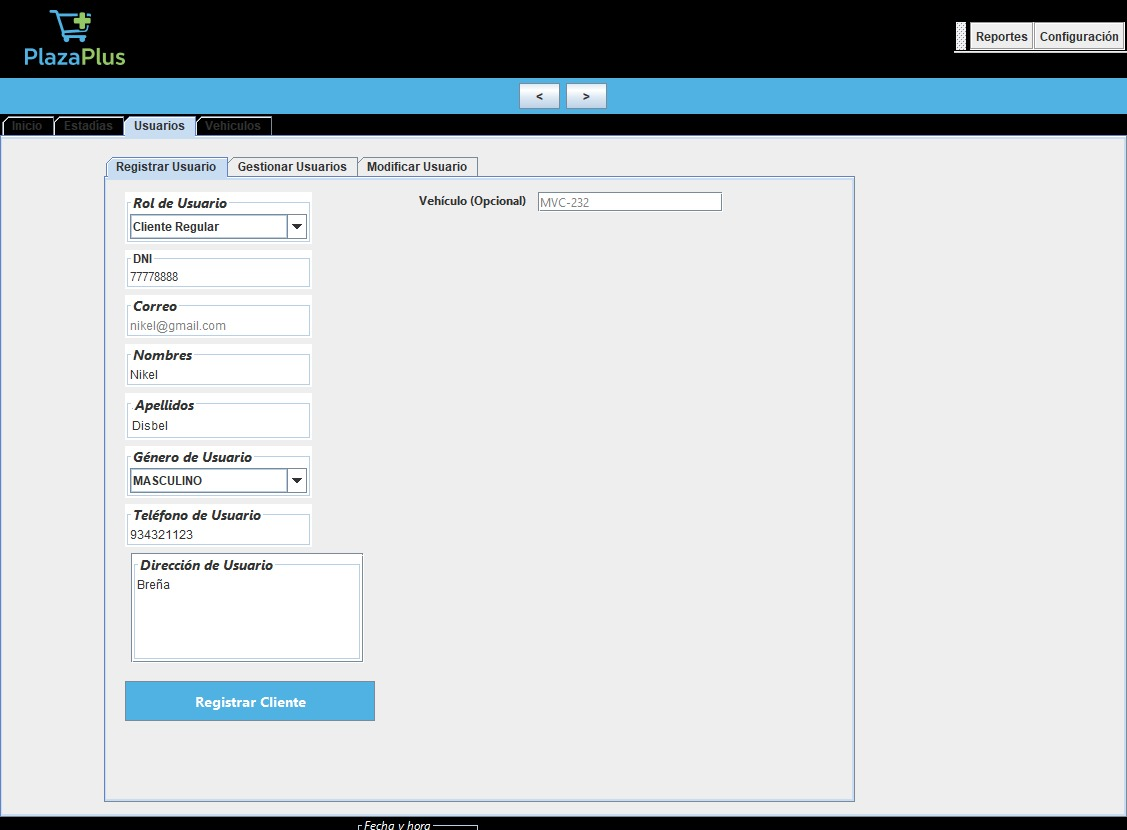


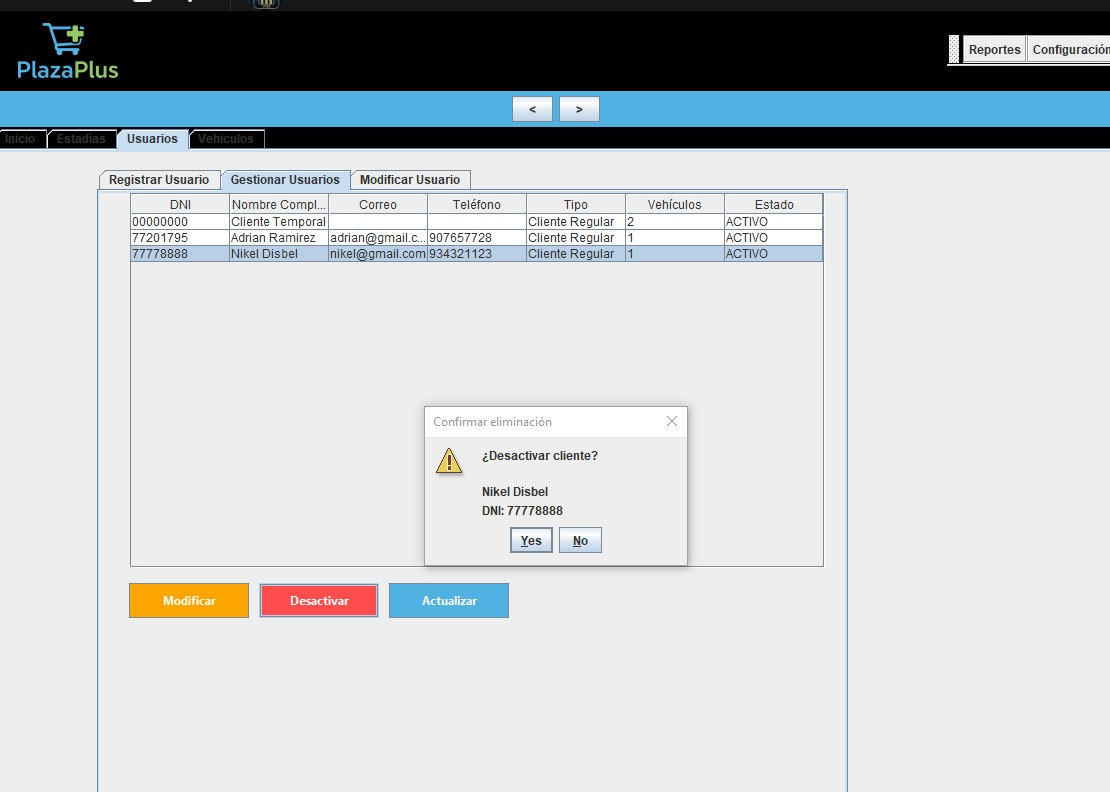
Ilustración 8 Elaboración propia, 2025.

## Prototipo o Interfaces

**Registro Usuario**

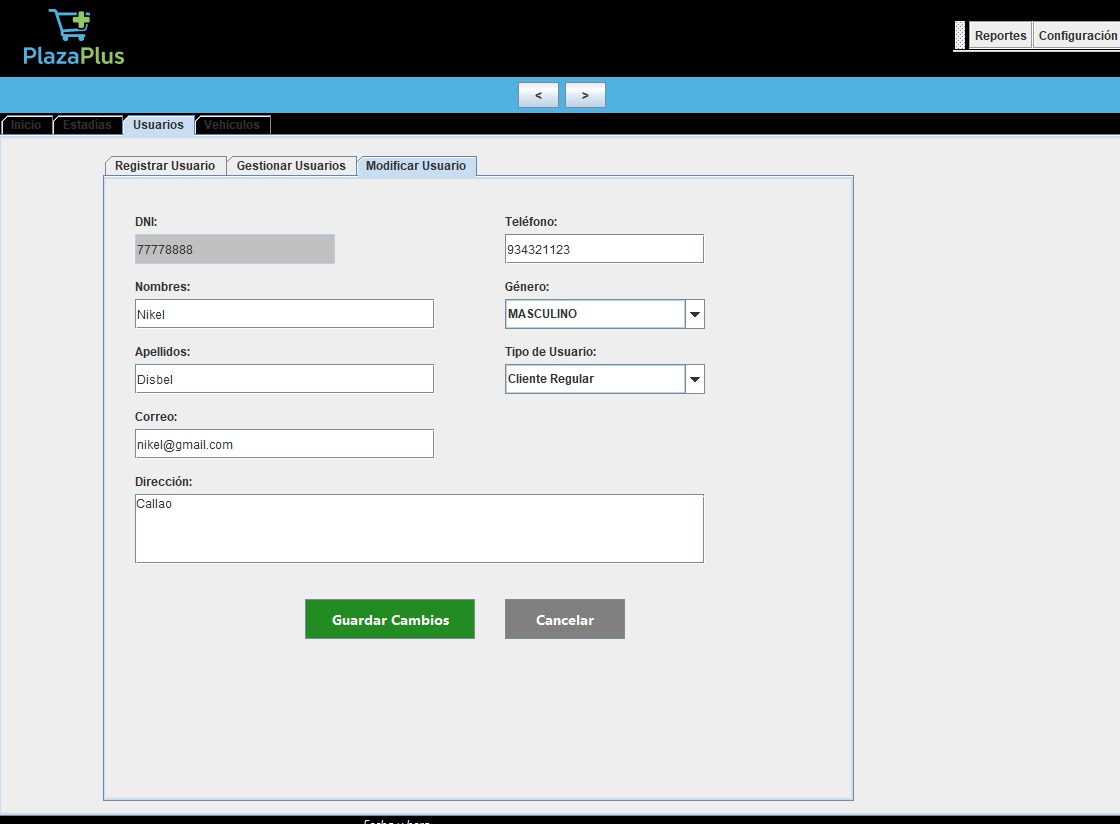
La sección "Registrar Usuario" en el software PlazaPlus es un formulario completo diseñado para ingresar nuevos perfiles al sistema, donde la información se clasifica comenzando por el Rol de Usuario (en este caso, Cliente Regular), seguido de los datos de identificación personal obligatorios (DNI, nombres, apellidos, género) y la información de contacto (correo, teléfono y dirección); además, se incluye un campo Vehículo opcional para vincular un activo al cliente, y todo el proceso de recopilación de datos finaliza con el botón "Registrar Cliente" en la parte inferior.

**Gestionar Usuario**

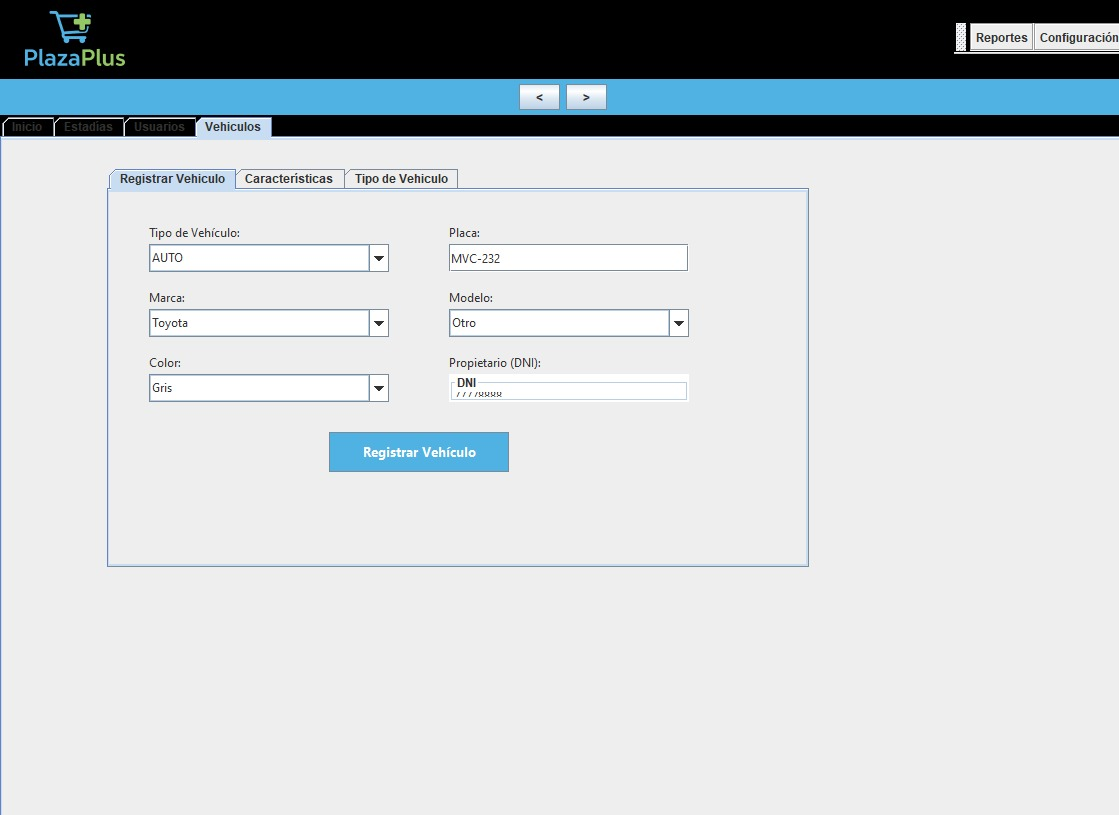
En este apartado del sistema PlazaPlus se permite gestionar a los usuarios registrados, mostrando sus datos básicos como DNI, nombre, correo, teléfono, tipo de cliente y estado. Desde aquí el administrador puede seleccionar un usuario para modificar su información, desactivarlo —previa confirmación mediante una ventana emergente— o actualizar la lista para visualizar los cambios realizados, facilitando así el control y mantenimiento de los clientes dentro del sistema.  


**Modificar Usuario**

Esta imagen muestra la pestaña "Modificar Usuario". En esta sección, el sistema ha cargado los datos existentes de un cliente (Nikel Disbel, DNI 77778888), permitiendo al operador actualizar cualquier información, como el Teléfono o la Dirección (que ahora aparece como "Callao"). Es crucial notar que el campo DNI aparece bloqueado o en modo de solo lectura para evitar que se altere el identificador único del usuario, y el proceso de edición se confirma haciendo clic en el botón verde "Guardar Cambios".

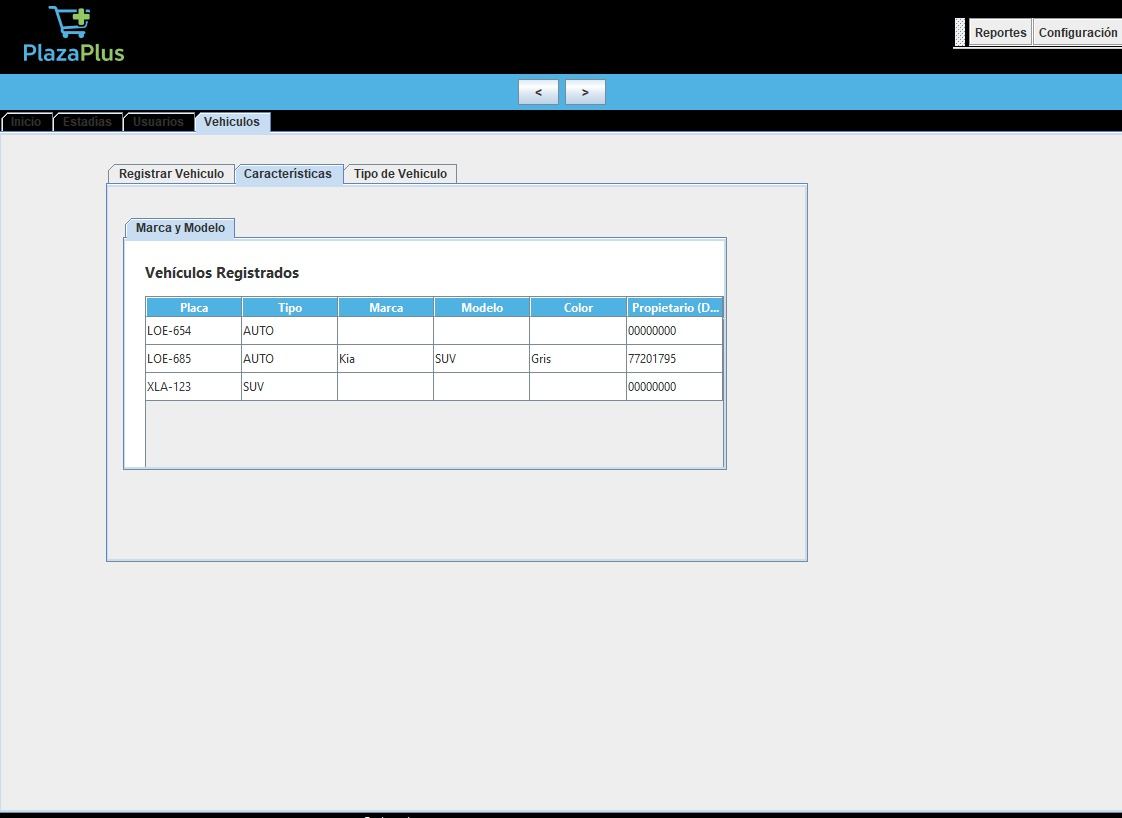


**Registro Vehículos**

En este apartado del sistema se permite registrar vehículos ingresando información esencial como el tipo de vehículo, marca, modelo, color, placa y el DNI del propietario. El usuario completa estos campos utilizando listas desplegables y cuadros de texto, y finalmente confirma el registro mediante el botón “Registrar Vehículo”, lo que facilita asociar correctamente cada vehículo con su dueño dentro de la plataforma PlazaPlus.  


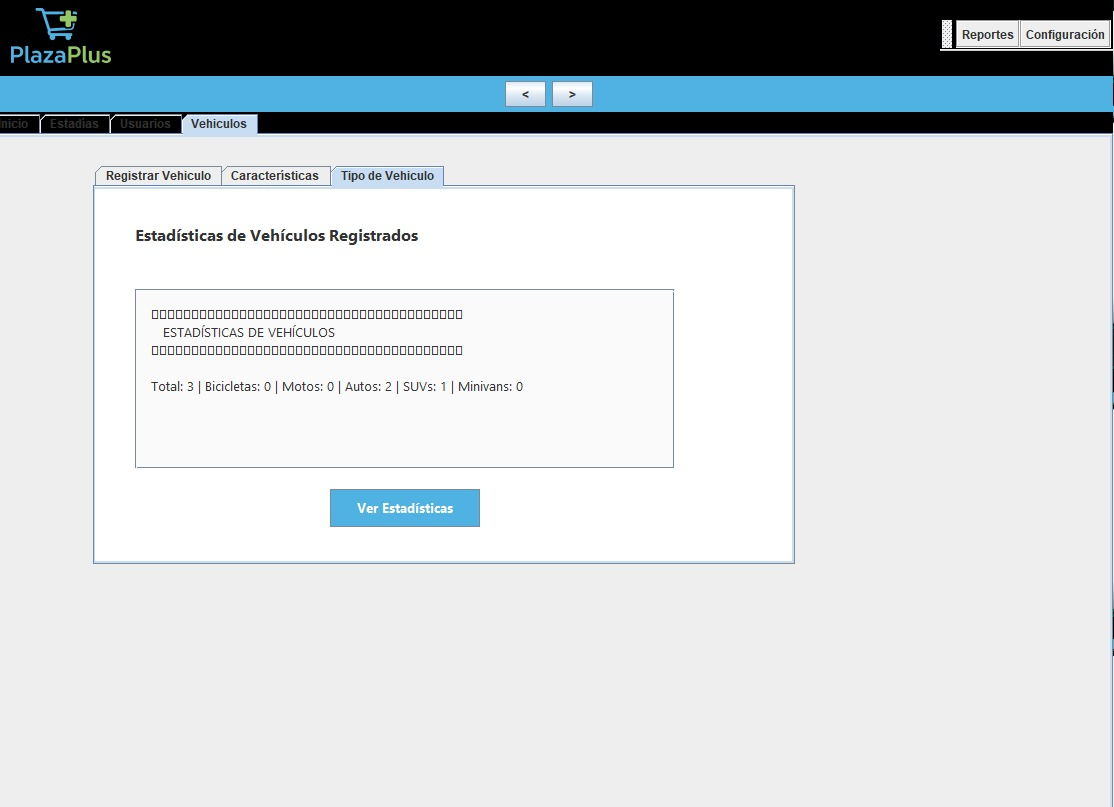
**Características**

Esta imagen muestra a Plaza Plus en la sección "Vehículos", pero en la pestaña "Características". El objetivo de esta pantalla es presentar un resumen de los vehículos que ya han sido registrados en el sistema, mostrando una tabla con los detalles clave de cada unidad, como la Placa, el Tipo, la Marca, el Modelo, el Color y, lo más importante, el ID del Propietario asociado, sirviendo como una herramienta de visualización y gestión del inventario vehicular.



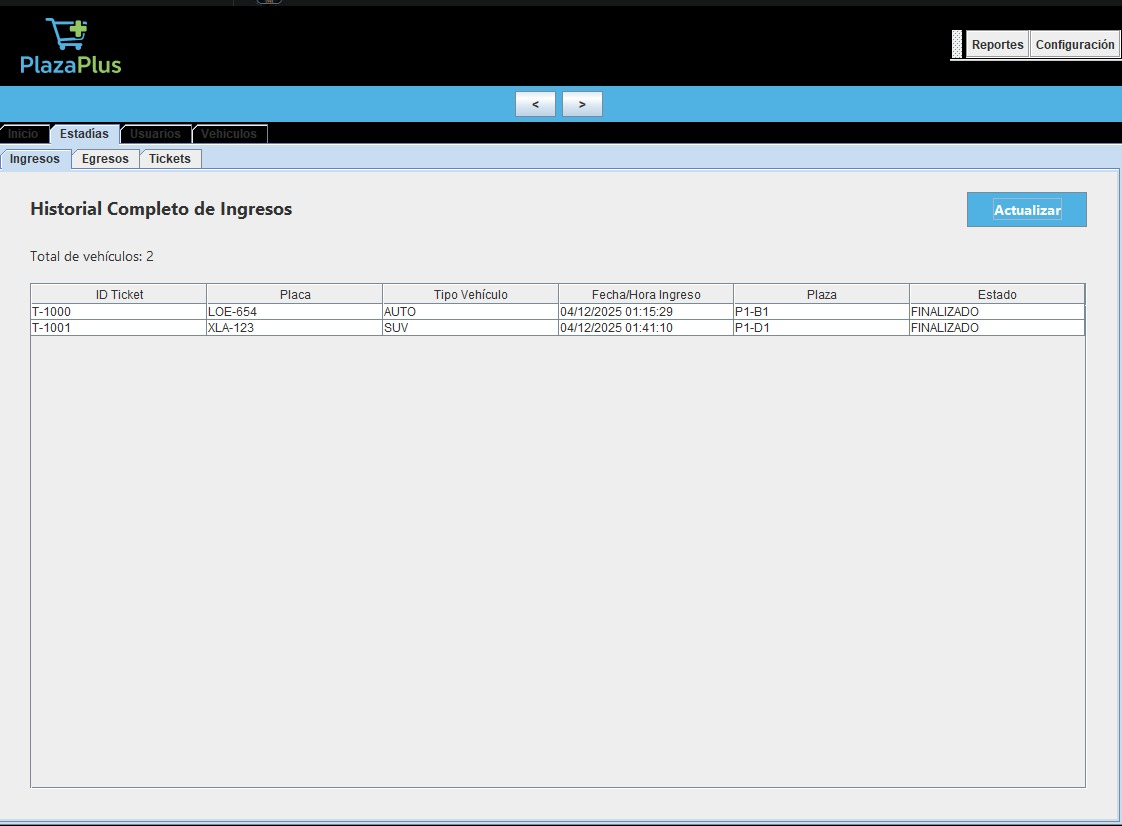
**Estadísticas de vehículos registrados**

Esta imagen muestra la interfaz del software PlazaPlus en la sección "Vehículos", específicamente en la pestaña "Tipo de Vehículo", la cual está dedicada a las "Estadísticas de Vehículos Registrados". El objetivo de esta pantalla es presentar un resumen cuantitativo del inventario, mostrando el Total: 3 vehículos y desglosándolo por tipo, indicando que hay 2 Autos y 1 SUV registrados, lo que proporciona una vista de alto nivel sobre la composición de la flota, con un botón para "Ver Estadísticas" más detalladas.



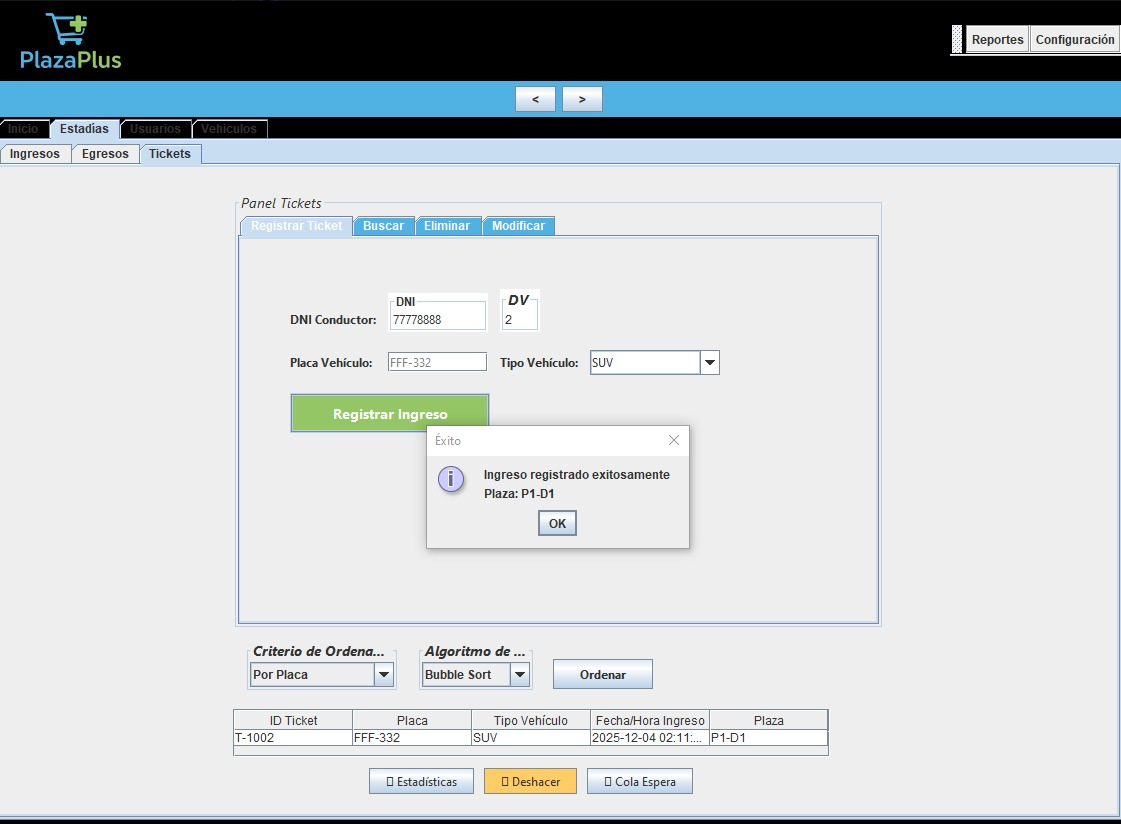
**Ingresos**

En este apartado se muestra el historial completo de ingresos al estacionamiento, donde se puede visualizar información detallada de cada ticket generado, como el ID, la placa del vehículo, el tipo de vehículo, la fecha y hora de ingreso, la plaza utilizada y el estado del registro. Esta sección permite al administrador llevar un control ordenado de todos los ingresos realizados y actualizar la información con el botón “Actualizar”.



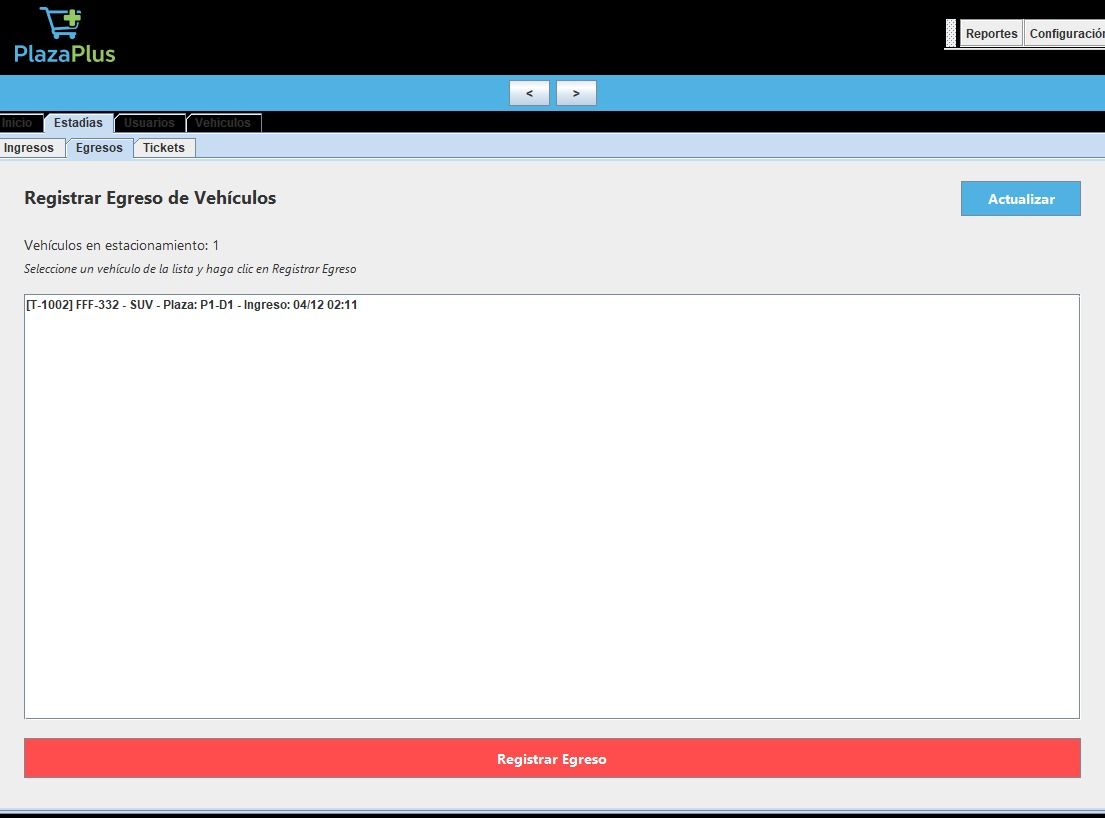
**Tickets**

En esta sección del sistema se realiza el registro de ingresos de vehículos al estacionamiento. El operador ingresa datos como el DNI del conductor, la placa y el tipo de vehículo, y luego confirma el proceso mediante el botón “Registrar Ingreso”. Al completarse el registro, el sistema asigna automáticamente una plaza disponible y muestra un mensaje de éxito con la ubicación asignada. Además, en la parte inferior se visualiza la lista de tickets generados, junto con opciones para ordenar, buscar, modificar o eliminar registros según sea necesario.



**Egresos**

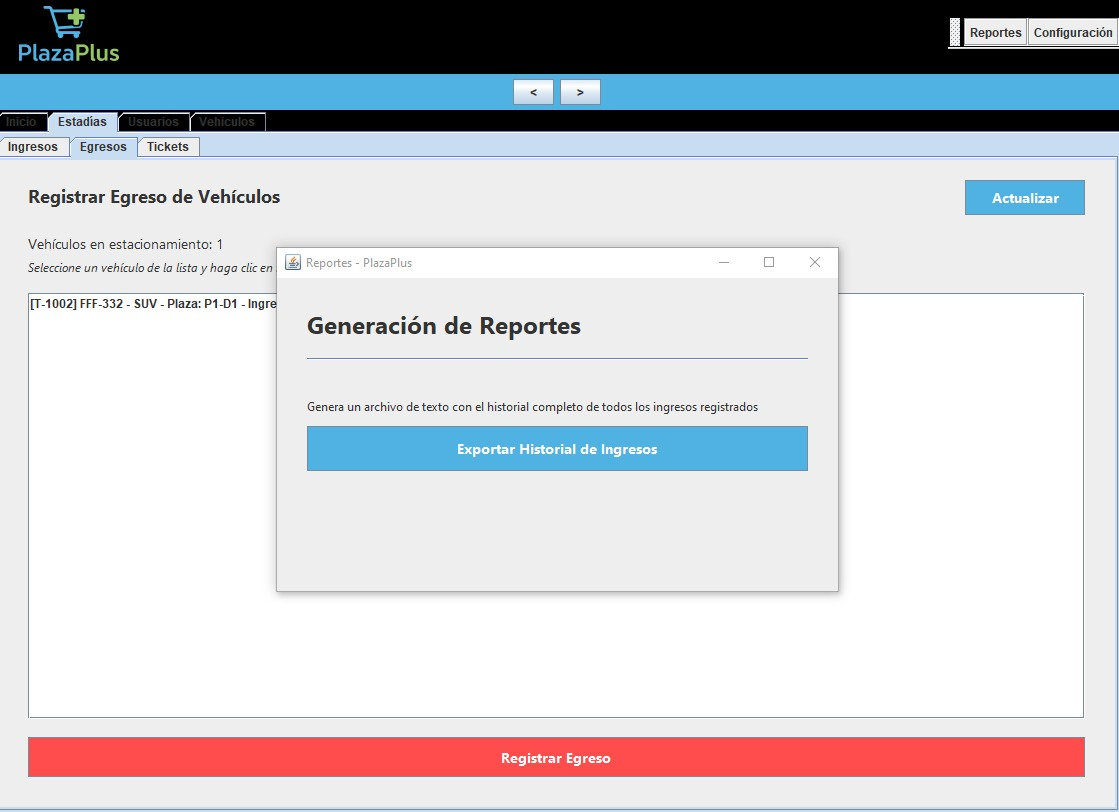
Esta imagen muestra el "Registrar Egreso de Vehículos". La pantalla lista los vehículos que están actualmente en el estacionamiento (mostrando 1), detallando información esencial como el número de ticket, la placa (FFF-332), el tipo, la plaza asignada y la hora de ingreso; el flujo de trabajo requiere que el operador seleccione el vehículo de la lista y presione el botón rojo "Registrar Egreso" para finalizar la sesión de estacionamiento.



**Reportes**

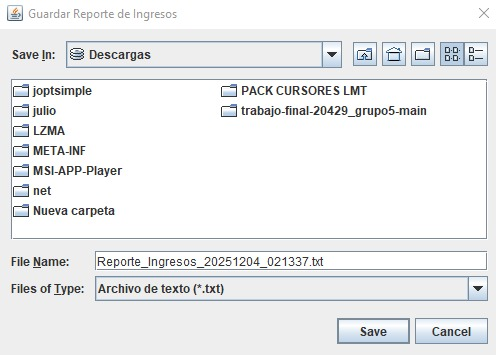
**Generación del reporte**

Esta imagen muestra la pestaña "Egresos" del módulo "Estadías", pero la acción principal es la ventana emergente de "Generación de Reportes". Esta funcionalidad permite al usuario generar y exportar un archivo de texto con el "historial completo de todos los ingresos registrados" en el sistema, lo cual es una herramienta de auditoría clave para revisar la actividad operativa del estacionamiento.



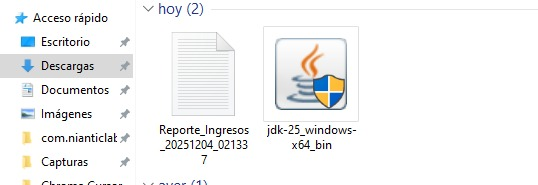
**Guardado de reportes de ingreso**

Esta imagen muestra un cuadro de diálogo del sistema operativo titulado "Guardar Reporte de Ingresos", el cual aparece como el paso final tras solicitar la exportación del historial de ingresos en la pantalla anterior. El objetivo de esta ventana es permitir al usuario elegir la ubicación donde se guardará el archivo (actualmente en "Descargas"), confirmar el nombre del archivo (que incluye la fecha y hora.



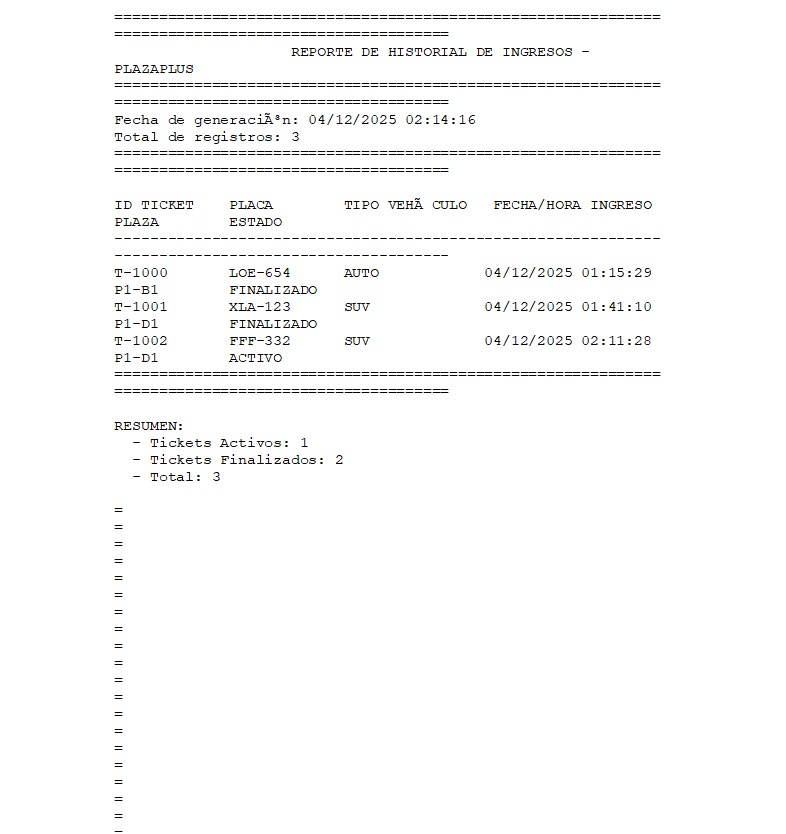
**Localizando la descarga**

Esta imagen muestra una vista del explorador de archivos de Windows, específicamente de la carpeta "Descargas", y sirve como confirmación visual de la exportación del reporte de ingresos. Se puede ver el archivo de texto generado por el software PlazaPlus, identificado con el nombre Reporte\_Ingresos\_20251204\_021337, que verifica que el proceso de auditoría y guardado de datos se realizó correctamente en el sistema local, junto a un archivo jdk-25... que sugiere que la aplicación PlazaPlus está desarrollada en Java.



**Reporte del historial del ingreso**

Esta imagen muestra el resultado del proceso de exportación de datos. El reporte ofrece un registro detallado de los 3 vehículos que han ingresado al sistema, incluyendo información clave como el ID del ticket, la placa, el estado (ACTIVO o FINALIZADO), el tipo de vehículo y la fecha/hora de ingreso, y se concluye con un resumen que indica el número de tickets activos y finalizados, sirviendo como el principal documento de auditoría y operación del estacionamiento.



## CAPITULO 4

## Diseño de la Aplicación

La aplicación PlazaPlus ha sido desarrollada utilizando la biblioteca gráfica Java Swing, implementando una estructura modular que facilita tanto la navegación como el mantenimiento del sistema. El diseño parte de una ventana principal de inicio de sesión (LoginVista), la cual permite autenticar a los usuarios y administradores antes de acceder a las funcionalidades del sistema.

El sistema cuenta con un controlador principal (ControladorPrincipal) que gestiona la lógica central del estacionamiento, incluyendo la asignación de plazas según el tipo de vehículo, la emisión de tickets, el registro de ingresos y egresos, así como la gestión de la cola de espera y el historial de operaciones. Cada módulo funcional se implementa en ventanas e interfaces independientes, manteniendo una estética coherente y una distribución uniforme de componentes.

Las interfaces gráficas, como LoginVista, Principal, TablaTickets y las ventanas de administración, integran campos de texto para el ingreso de datos, botones de acción (registrar, buscar, eliminar, deshacer), y tablas para visualizar la información procesada por el sistema, tales como vehículos activos, tickets emitidos y usuarios registrados.

Adicionalmente, se incorporaron elementos visuales orientados a mejorar la experiencia del usuario, tales como el uso de fuentes legibles y un encabezado en azul que resalta el título PlazaPlus, ofreciendo una interfaz profesional, intuitiva y amigable. Esta disposición facilita la gestión eficiente de los estacionamientos de la empresa, optimizando los ingresos, la disponibilidad de plazas y la atención al cliente.

**Funcionalidades del Aplicativo Implementadas**

El sistema desarrollado para PlazaPlus incluye múltiples funcionalidades distribuidas en módulos independientes, diseñados para facilitar la gestión eficiente de los estacionamientos y los procesos asociados. A continuación, se detallan las principales características implementadas por módulo:

**Gestión de usuarios (Usuario.java)**

* + - Registro de usuarios con email, contraseña, nombre y rol.
    - Modificación de datos de usuario (email, contraseña, nombre y rol).
    - Verificación de rol de usuario (administrador o usuario normal).
    - Método para verificar si un usuario es administrador (es administrador()).
    - Visualización de información básica mediante toString().

**Gestión de conductores (Conductor.java)**

* + - Registro de conductores con DNI, DV, nombres y apellidos.
    - Validación de datos:
      * DNI de 8 dígitos.
      * DV entre 0 y 9.
      * Nombres y apellidos solo con letras y espacios.
    - Asignación de ID único (no repetible) mediante un Set estático.
    - Edición de datos de conductores.
    - Visualización de información mediante toString ().

**Gestión de vehículos (TipoVehiculo.java)**

* + - Tipos de vehículos: Bicicleta, Moto, Auto, SUV y Minivan.
    - Cada tipo tiene un factor de tarifa y un código asociado.
    - Obtención de un arreglo con todos los códigos de vehículos ( getArregloCodigos()).

**Gestión de plazas (EstadoPlaza.java)**

* + - Estados posibles de plazas: Libre, Ocupada, Reservada, Mantenimiento.
    - Cada estado indica disponibilidad (true si está libre, false en otro caso).
    - Obtención de un arreglo con todos los códigos de estado (getArregloCodigos ()).

**Gestión de tickets (Ticket.java)**

* + - Registro de tickets: código, vehículo, plaza, hora de ingreso y hora de salida.
    - Cálculo y registro del monto a pagar.
    - Estado de pago del ticket (pagado o no).
    - Métodos para obtener y modificar toda la información del ticket (getters/setters).

**Gestión de solicitudes de espera (SolicitudEspera.java)**

* + - Registro de solicitudes de espera: placa, tipo de vehículo y fecha/hora de solicitud.
    - Edición y obtención de datos de solicitudes (getters/setters).
    - Visualización de la solicitud mediante toString ().

**Vistas**

* + - **VentanaInicio.java:**
* Interfaz inicial con botones para acceder a:
* VentanaAccesoAdministrador
* Configuración de layout y diseño de paneles, logo y eslogan.

Gestión de cierre de ventana (listeners windowClosed y windowClosing).

**VentanaAdministrador.java:**

* + - Panel con pestañas (Tickets).
    - Tabla de tickets para visualizar información.
    - Encabezado y pie de página en la interfaz.

**Controladores**

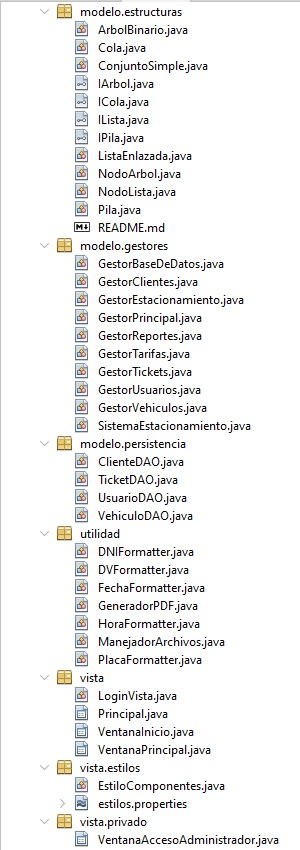
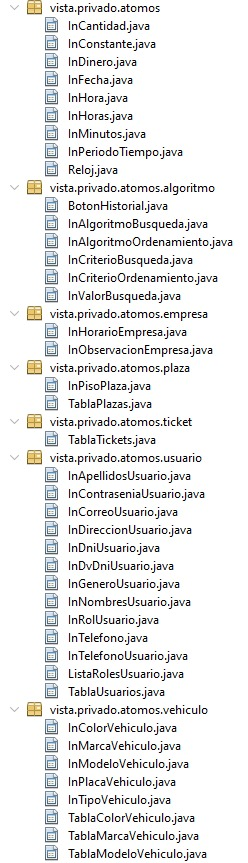
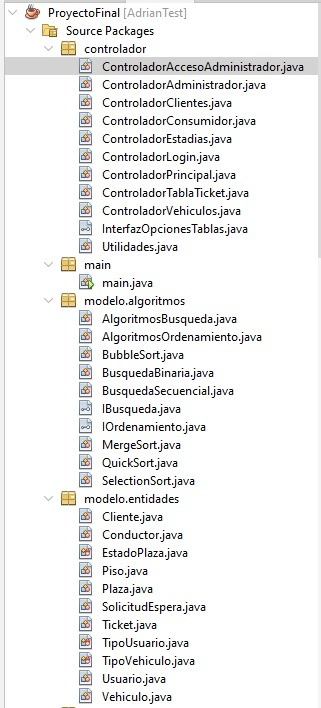
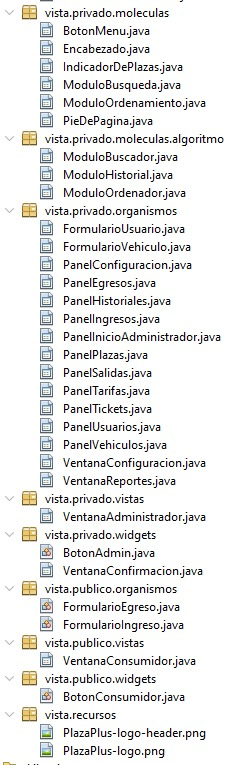
**ControladorAdministrador.java:**

* + - Controla la interacción entre la VentanaAdministrador y los modelos de tickets y plazas.

**Estructura de paquetes**

La estructura de paquetes del proyecto PlazaPlus organiza el sistema en módulos bien definidos, lo que facilita la comprensión, el mantenimiento y la escalabilidad del software. Esta organización permite separar responsabilidades entre vistas, controladores, gestores y modelos de entidades, asegurando un flujo de información eficiente y un diseño modular y reutilizable.

Esta estructura modular facilita la integración de nuevas funcionalidades y asegura que el proyecto sea escalable, mantenible y coherente con los principios de diseño orientado a objetos.

**Estructura del Paquete**

[Emisor] --(clave secreta K, plaintext)--> AES Encrypt => ciphertext --> [Canal]

[Receptor] --(recibe ciphertext, usa K)--> AES Decrypt => plantes.operaciones correspondientes en los modelos de datos. Cada controlador está asociado a su módulo, reforzando la cohesión y la mantenibilidad del sistema.

La estructura modular del proyecto permite la escalabilidad, la integración de nuevas funcionalidades y un mantenimiento más eficiente, garantizando que los distintos componentes trabajen de manera coordinada y consistente dentro de la aplicación.

En este proyecto, se ha decidido centrar el análisis en el paquete modelo.entidades, dado que constituye la base estructural y conceptual del sistema. Este paquete contiene las clases que representan las entidades fundamentales, como Usuario, Conductor, TipoVehiculo, EstadoPlaza, Ticket y SolicitudEspera. Cada clase encapsula atributos y métodos que permiten almacenar, validar y manipular la información de manera organizada, garantizando la integridad de los datos y la correcta operación de los módulos asociados.

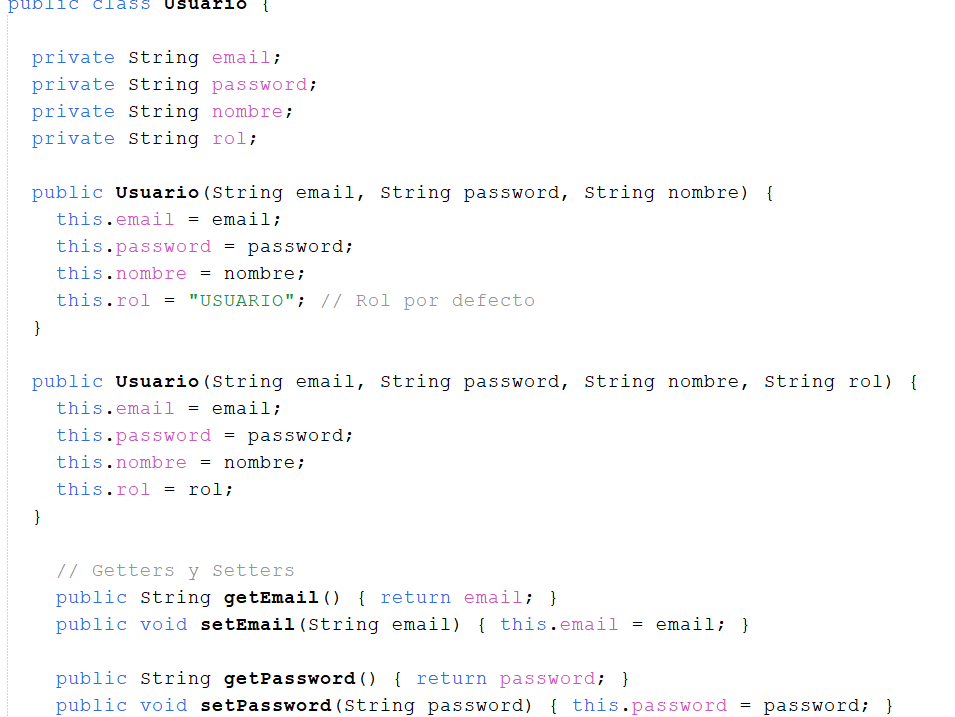
El paquete modelo constituye el núcleo lógico del sistema, ya que define cómo se representa y gestiona la información sobre usuarios, vehículos, plazas, tickets y solicitudes de espera. La implementación incluye validaciones esenciales, como verificación de roles de usuario, formatos de DNI, asignación de ID único y control de disponibilidad de plazas, así como métodos que facilitan la obtención y modificación de los datos, asegurando un flujo coherente de información.

Se ha optado por no incluir en este análisis los paquetes vista y controlador, dado que su función principal es la interacción con el usuario y el manejo de eventos. Si bien son cruciales para la operatividad completa de la aplicación, su descripción detallada no aporta información adicional al análisis estructural de las entidades y complicaría innecesariamente la explicación.

Esta decisión permite mantener el enfoque en los componentes esenciales del diseño, proporcionando una visión clara de cómo se organiza y manipula la información dentro del sistema.

## Clases usadas en la aplicación

**Usuario.java**  
La clase Usuario representa a los usuarios del sistema, incorporando atributos esenciales como correo electrónico, contraseña, nombre completo y rol. Esta clase permite gestionar la información de los usuarios mediante métodos que facilitan el registro, la modificación y la validación de credenciales. Adicionalmente, el método es Administrador() permite verificar si un usuario tiene privilegios administrativos, asegurando un control adecuado sobre las operaciones críticas del sistema (Figura1).

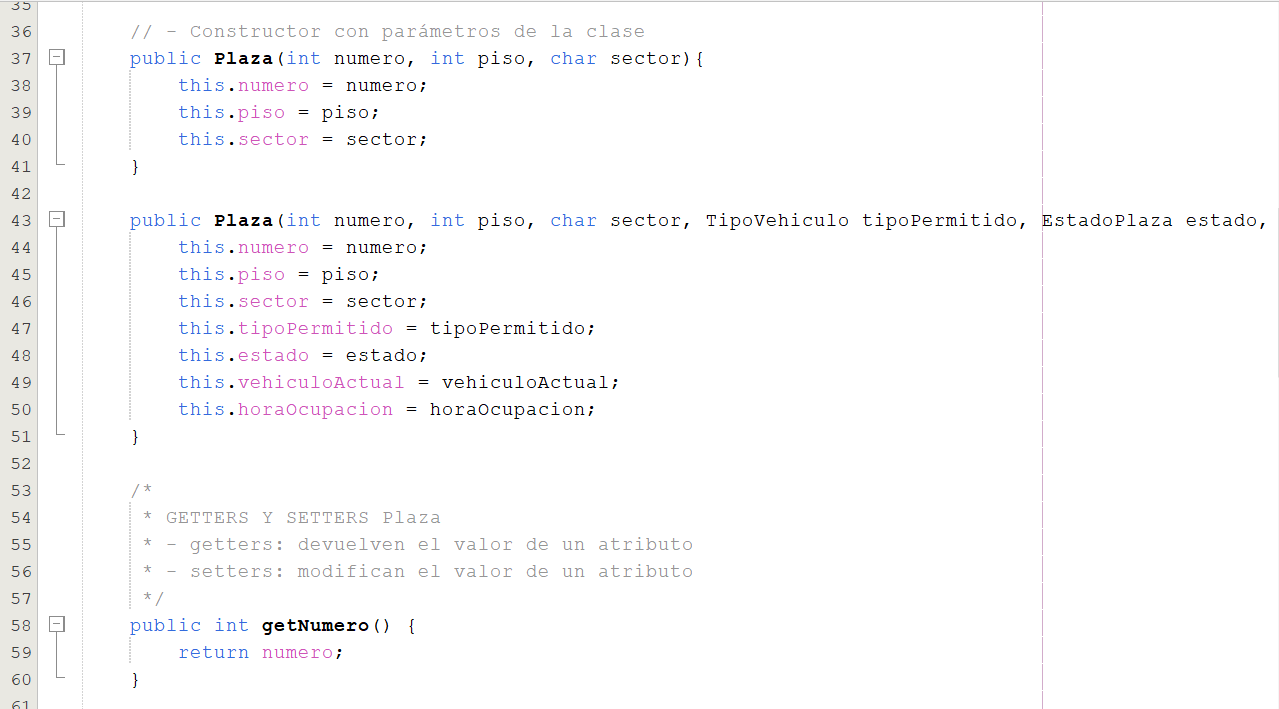
**Figura 1**  
Nota. Representación de la clase Usuario.java con sus atributos y métodos principales.

**Conductor.java**  
La clase Conductor define y administra la información de los conductores registrados, validando datos importantes como DNI, dígito verificador, nombres y apellidos, asegurando un ID único para cada conductor. Esta clase proporciona métodos para registrar, editar y visualizar datos, facilitando la organización de la información dentro del sistema (Figura 2).  
**Figura 2**  

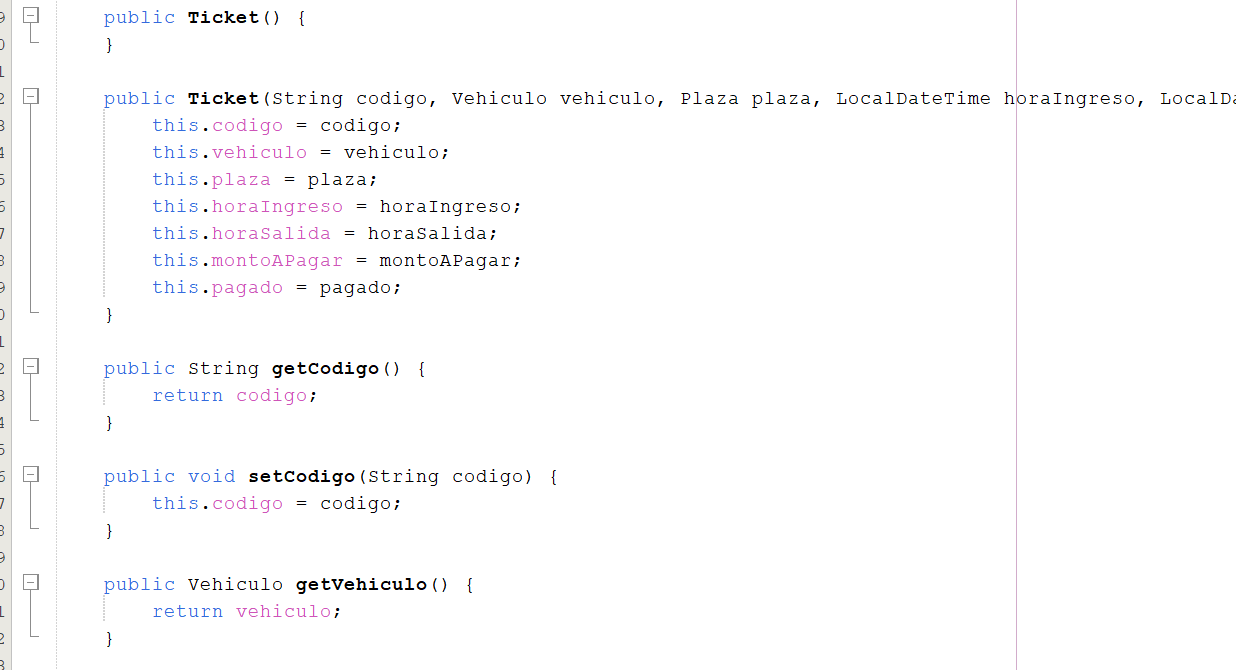

**Vehiculo.java y TipoVehiculo.java**  
Estas clases permiten definir y clasificar los vehículos que interactúan con el sistema. Cada tipo de vehículo posee un factor de tarifa y un código único, facilitando operaciones internas como la categorización eficiente de vehículos (Figura 3).  
**Figura 3**



**Plaza.java y EstadoPlaza.java**  
Estas clases gestionan las plazas de estacionamiento y su estado, indicando si están libres, ocupadas, reservadas o en mantenimiento. Esto permite un control efectivo de la disponibilidad y optimiza la gestión del flujo de vehículos en el sistema (Figura 4).  
**Figura 4**



**Ticket.java**  
La clase Ticket registra los tickets generados por el ingreso y salida de vehículos, incluyendo información como plaza utilizada, hora de entrada y salida, monto a pagar y estado de pago. Esto asegura un control preciso de las operaciones y la facturación dentro del sistema (Figura 5).  
**Figura 5**



**SolicitudEspera.java**  
La clase SolicitudEspera administra las solicitudes de espera cuando no hay plazas disponibles. Registra la placa del vehículo, el tipo de vehículo y la fecha y hora de la solicitud, garantizando que los usuarios sean atendidos en cuanto se libere un espacio (Figura 6).  
**Figura 6**



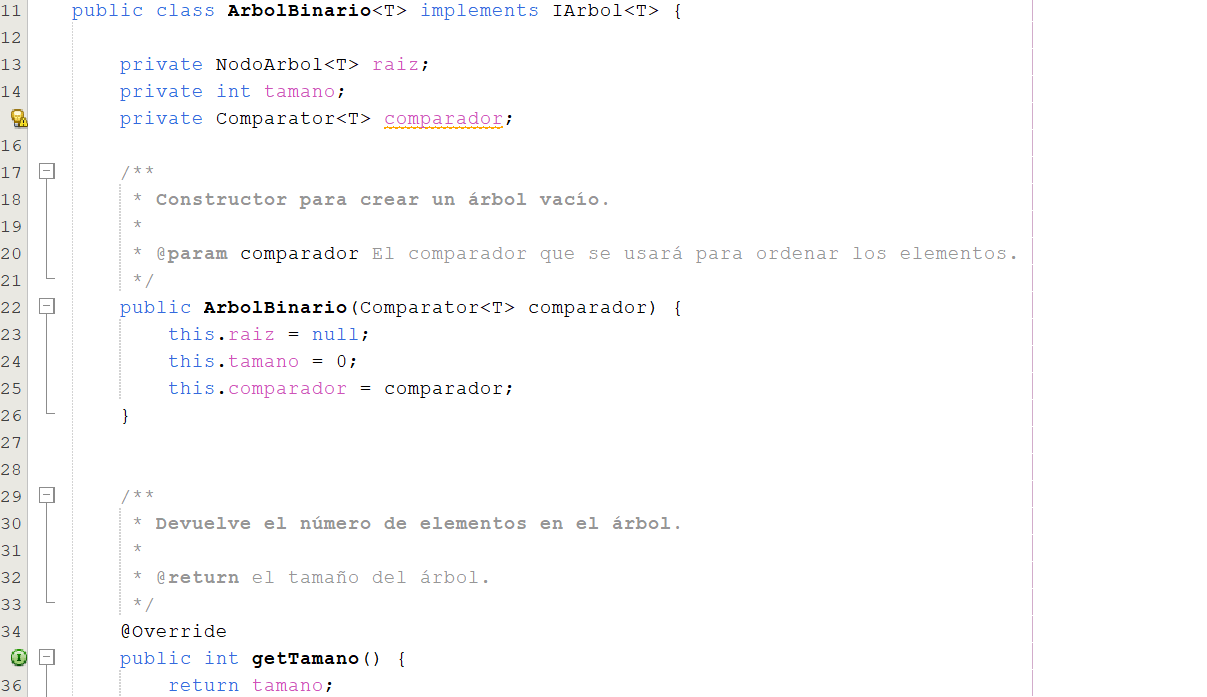
## Código Fuente

El código fuente del sistema PlazaPlus se encuentra organizado en una estructura modular de paquetes, lo que permite mantener una separación clara de responsabilidades y facilitar la extensibilidad del proyecto. En el paquete estructuras se implementan las colecciones dinámicas utilizadas por el sistema, tales como listas enlazadas, pilas, colas, conjuntos y árboles binarios, cada una construida desde cero para garantizar eficiencia asintótica y control total sobre sus operaciones internas. Por su parte, el paquete algoritmos incorpora las estrategias de ordenamiento y búsqueda empleadas para la generación de reportes y la recuperación optimizada de información, cumpliendo un rol fundamental en la velocidad de consulta del sistema.

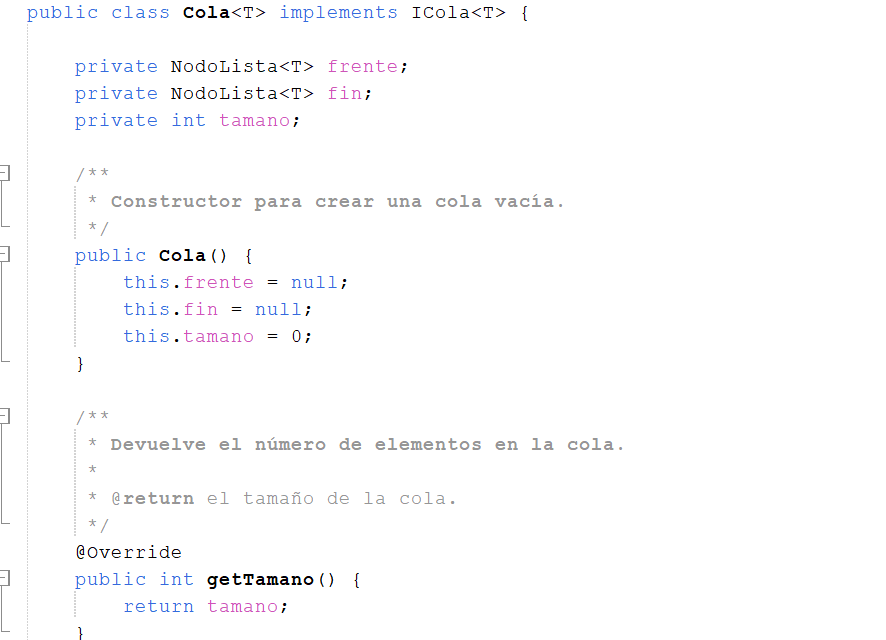
La lógica de negocio se concentra en el paquete gestores, donde clases como *GestorPrincipal*, *SistemaEstacionamiento* y *GestorUsuarios* coordinan el registro de clientes, vehículos, tickets y plazas, además de validar credenciales y administrar el flujo operacional del estacionamiento. Finalmente, el paquete modelos reúne las clases que representan las entidades del dominio incluyendo Cliente, Vehiculo, Plaza y Ticket asegurando una estructura clara y consistente para el manejo de datos.

Esta organización del código fuente permite mantener alta cohesión dentro de cada módulo, bajo acoplamiento entre componentes y una arquitectura limpia que favorece la mantenibilidad y escalabilidad del sistema.

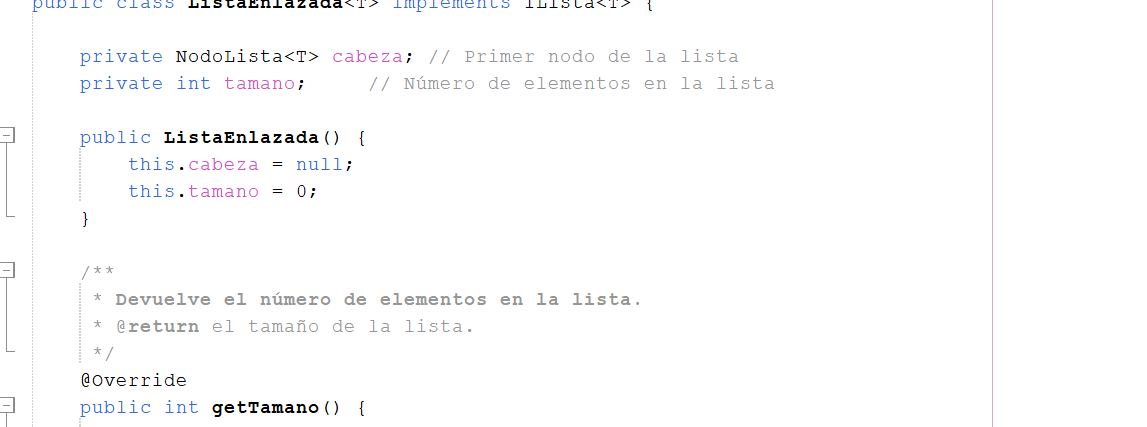
## Paquete Modelo

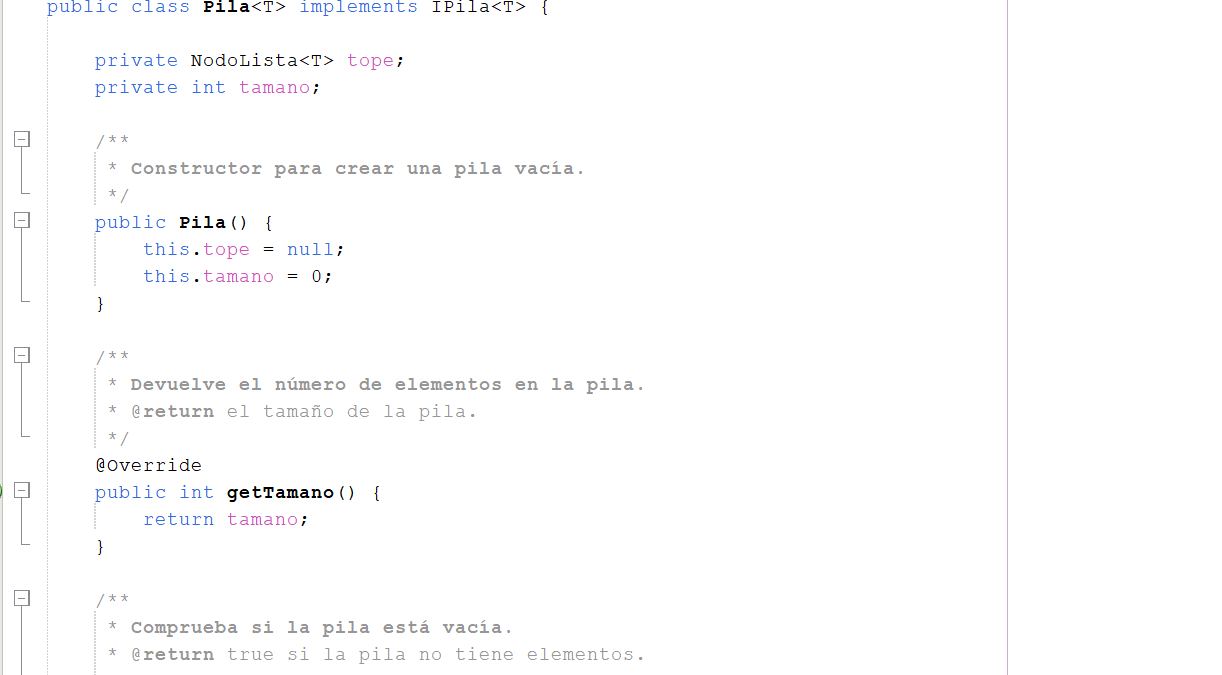
**ArbolBinario.java e IArbol.java**  
Estas clases implementan árboles binarios que permiten organizar y buscar datos de manera eficiente. La interfaz IArbol define los métodos esenciales para insertar, buscar y recorrer nodos, proporcionando una estructura jerárquica flexible (Figura 7).  
**Figura 7**

**Cola.java e ICola.java**  
Representan estructuras tipo cola que gestionan elementos bajo el principio FIFO (First In, First Out). Son utilizadas para solicitudes de espera, tickets pendientes u otros procesos que requieren atención en orden de llegada (Figura 8).

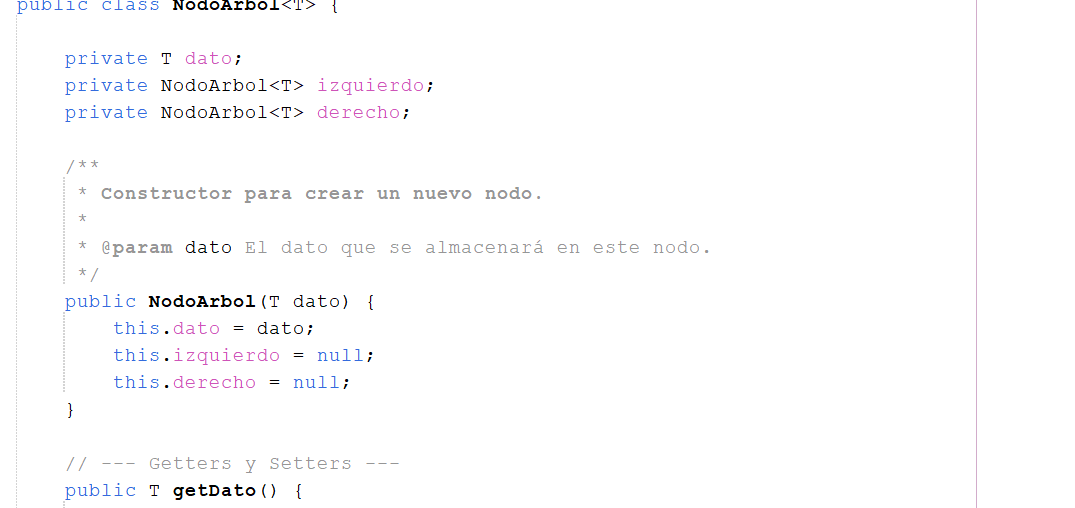
**Figura 8**  


**ListaEnlazada.java e ILista.java**  
Permiten la creación de listas enlazadas para almacenar objetos de manera dinámica. Facilitan la inserción, eliminación y búsqueda de elementos, siendo útiles en escenarios donde la cantidad de datos cambia constantemente (Figura 9).  
**Figura 9**



**Pila.java**  
Estas estructuras tipo LIFO (Last In, First Out) son utilizadas para gestionar elementos temporales, como historial de operaciones o acciones de deshacer, permitiendo revertir procesos de manera eficiente (Figura 10).  
Figura 10****

**NodoLista.java y NodoArbol.java**  
Son clases auxiliares que representan nodos individuales dentro de listas y árboles. Cada nodo contiene referencias a los datos y conexiones hacia otros nodos, constituyendo la base para la construcción de estructuras de datos complejas (Figura 11).

**Figura 11**

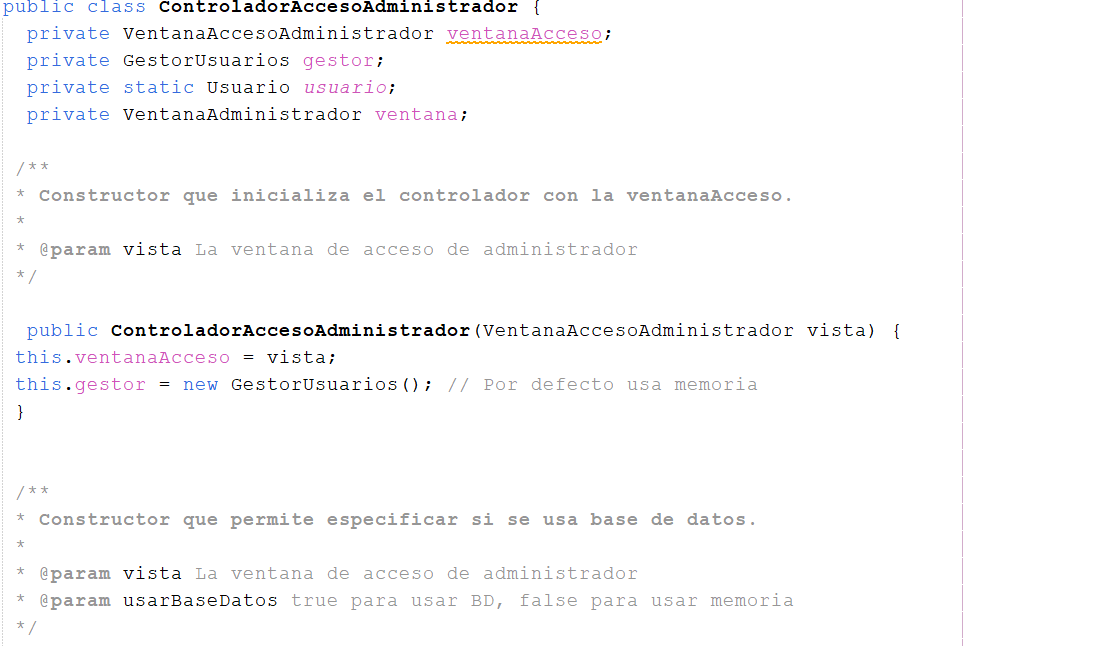
**Paquete Modelo.gestores**

**GestorPrincipal.java**  
Administra el estado de las plazas y regula el flujo de vehículos. Controla ocupación, reservas y mantenimientos, optimizando la disponibilidad y eficiencia del estacionamiento (Figura 13).  
**Figura 13**



**Paquete Controlador**

ControladorAccesoAdministrador.java  
coordina el acceso y la interacción del administrador con el sistema. Se encarga de procesar los eventos generados desde la interfaz de administración, validar las acciones realizadas y actualizar la información en tiempo real. Su implementación garantiza que únicamente los usuarios con permisos administrativos puedan ejecutar funciones críticas, manteniendo la seguridad, la integridad de los datos y un control adecuado sobre las operaciones internas del sistema (Figura 15).  
**Figura 15**



**Paquete Vista**

VentanaAdministrador.java  
Son interfaces gráficas que permiten interactuar con el sistema de forma intuitiva, mostrando información de plazas, tickets y solicitudes en tablas y paneles visuales. Estas interfaces facilitan la supervisión y gestión de operaciones, brindando herramientas claras a los usuarios y administradores (Figura 18)

**Figura 18  
VentanaInicio.java**

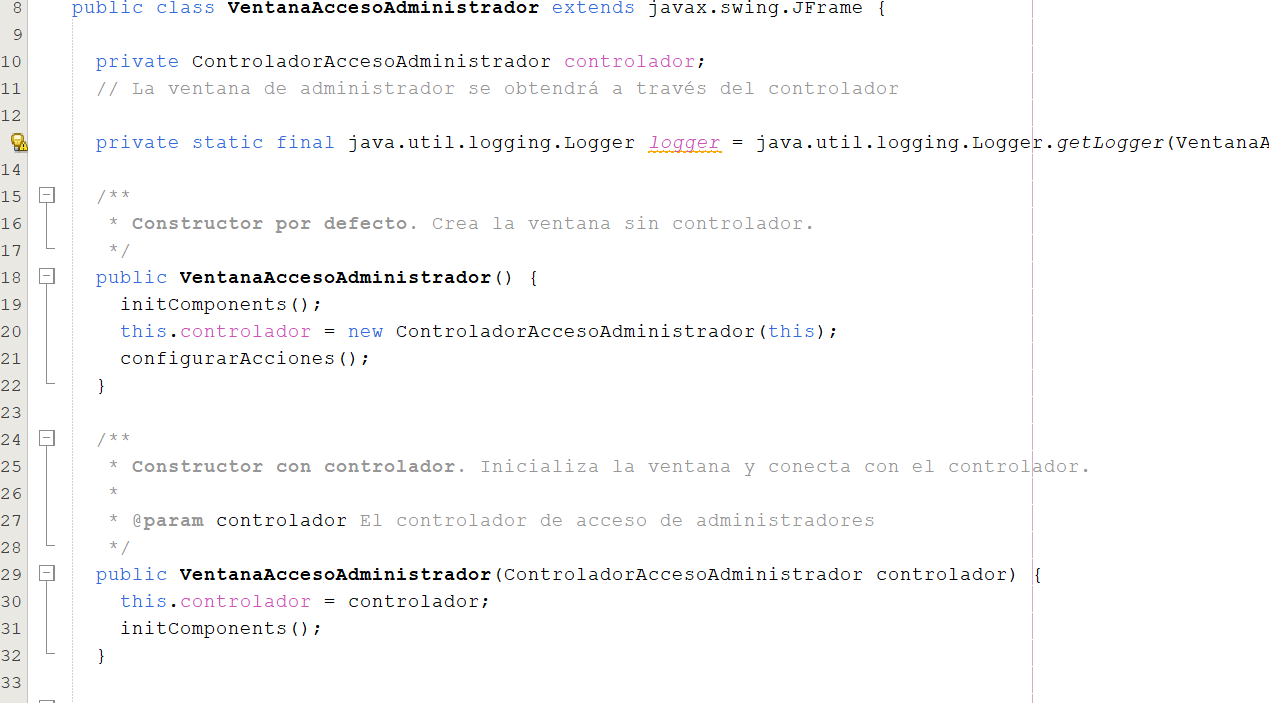
La clase VentanaInicio.java corresponde a la pantalla inicial del sistema PlazaPlus y forma parte de la capa Vista dentro del patrón MVC. Su función principal es actuar como punto de acceso para los dos tipos de usuarios del sistema: Administrador. Desde esta ventana se determina a qué interfaz debe dirigirse el usuario según la opción que seleccione (Figura 21).  
**Figura 21**



**VentanaAccesoAdministrador**

La clase VentanaAccesoAdministrador.java forma parte de la capa Vista dentro del patrón Modelo–Vista–Controlador (MVC) empleado por el sistema PlazaPlus. Su propósito principal es proporcionar una interfaz gráfica para que los administradores ingresen sus credenciales y puedan acceder al panel administrativo del sistema. Esta ventana se encarga de solicitar el correo y la contraseña, así como de comunicar estos datos al controlador correspondiente para verificar la autenticidad del usuario (Figura 22).

**Figura 22**



## Bibliografía

* Joyanes Aguilar, L. (2016). *Estructura de datos en Java* (1.ª ed.). McGraw-Hill/Interamericana de España, S.A.U.
* Universidad Autónoma Metropolitana, División de Ciencias Básicas e Ingeniería. (s. f.). *Algoritmos y estructuras de datos - Listas enlazadas*. <https://academicos.azc.uam.mx/franz/aed/docs/listas.pdf>
* Universidad Veracruzana, Facultad de Estadística e Informática. (s. f.). *Listas ligadas o enlazadas*. <https://www.uv.mx/personal/ermeneses/files/2021/08/Clase5-ListasEnlazadasFinal.pdf>
* Vaca, C. (2011). *Estructuras de datos y algoritmos. Tema 3: Arrays y listas enlazadas* [Apuntes de clase]. Departamento de Informática, Universidad de Valladolid. <https://www.infor.uva.es/~cvaca/asigs/doceda/tema3.pdf>

## Anexo

* Prototipo en figma: <https://www.figma.com/design/1GUEsJmpdsr6qLVlKPynzy/Prototipado?m=auto&t=h0b4uWhbDezG6jB6-1>
* Archivo de diagramas: <https://github.com/E-specter/DIAGRAMS-trabajo-final-20429_grupo5/invitations>