

DISEÑO Y DESARROLLO DE UN SISTEMA DE CONTROL DE ESTACIONAMIENTO INTELIGENTE PARA LA EMPRESA PLAZA PLUS, LIMA 2025

Integrantes:

Ramirez Rivera Adrian Jose Felix

Sihue Saavedra Julio César

Ortiz Angeles Eduard Frank

Velarde Ochoa Obed Victor Jesus

ÍNDICE

01. Aspectos Generales
02. Diseño de la Aplicación
03. Conclusiones y Recomendaciones

ASPECTOS GENERALES

Descripción del Problema

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 administración manual de plazas y 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.

Objetivos de la Solución

- Optimizar la gestión de los procesos internos del estacionamiento mediante un sistema automatizado que integre el control de plazas, ingresos, salidas y reportes.
- Reducir los errores humanos en la asignación de plazas, 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.

Alcance de la Solucion Tecnologica

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.

Funcionalidades de la Aplicación

Registro de Usuarios

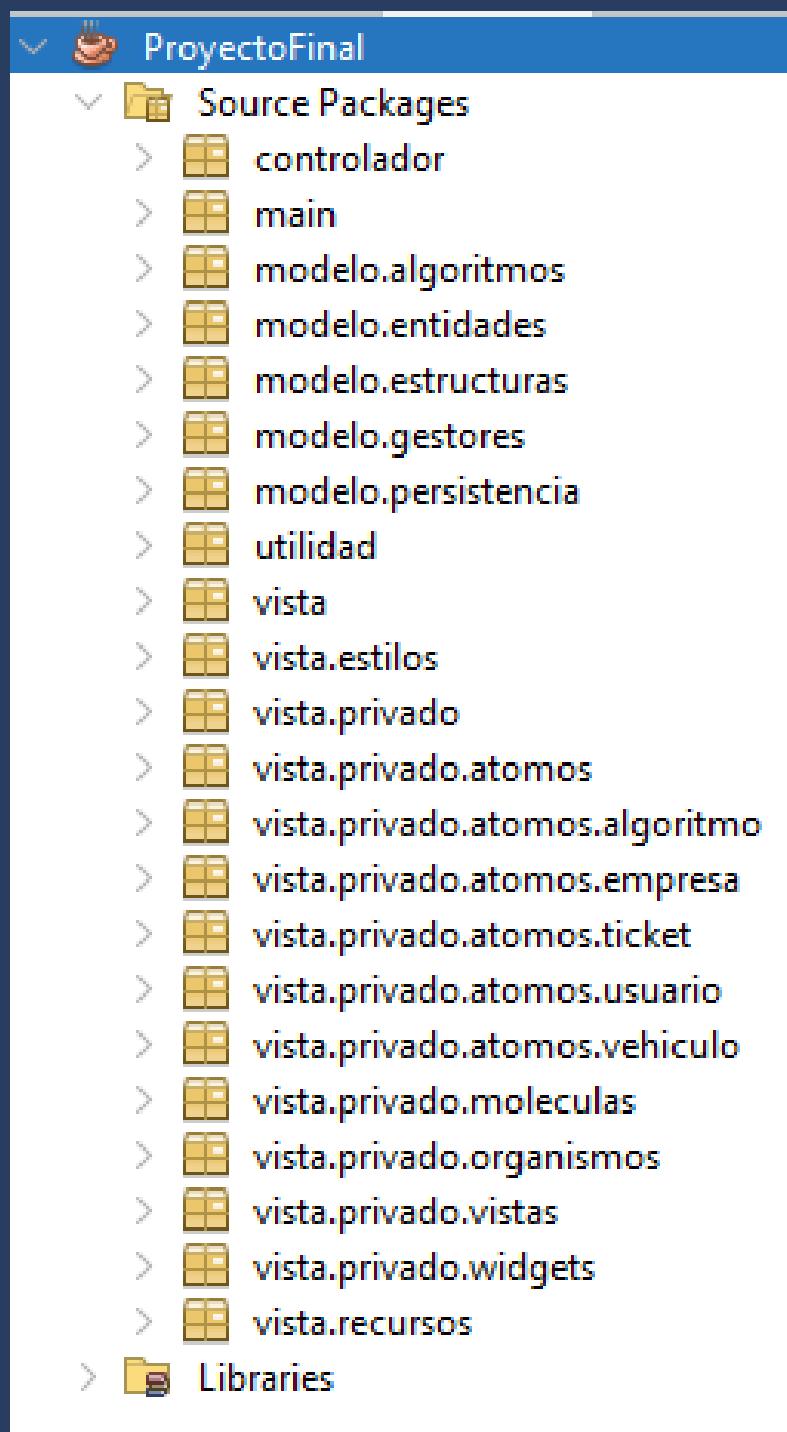
Estadias

Reportes

Registro de Vehículos

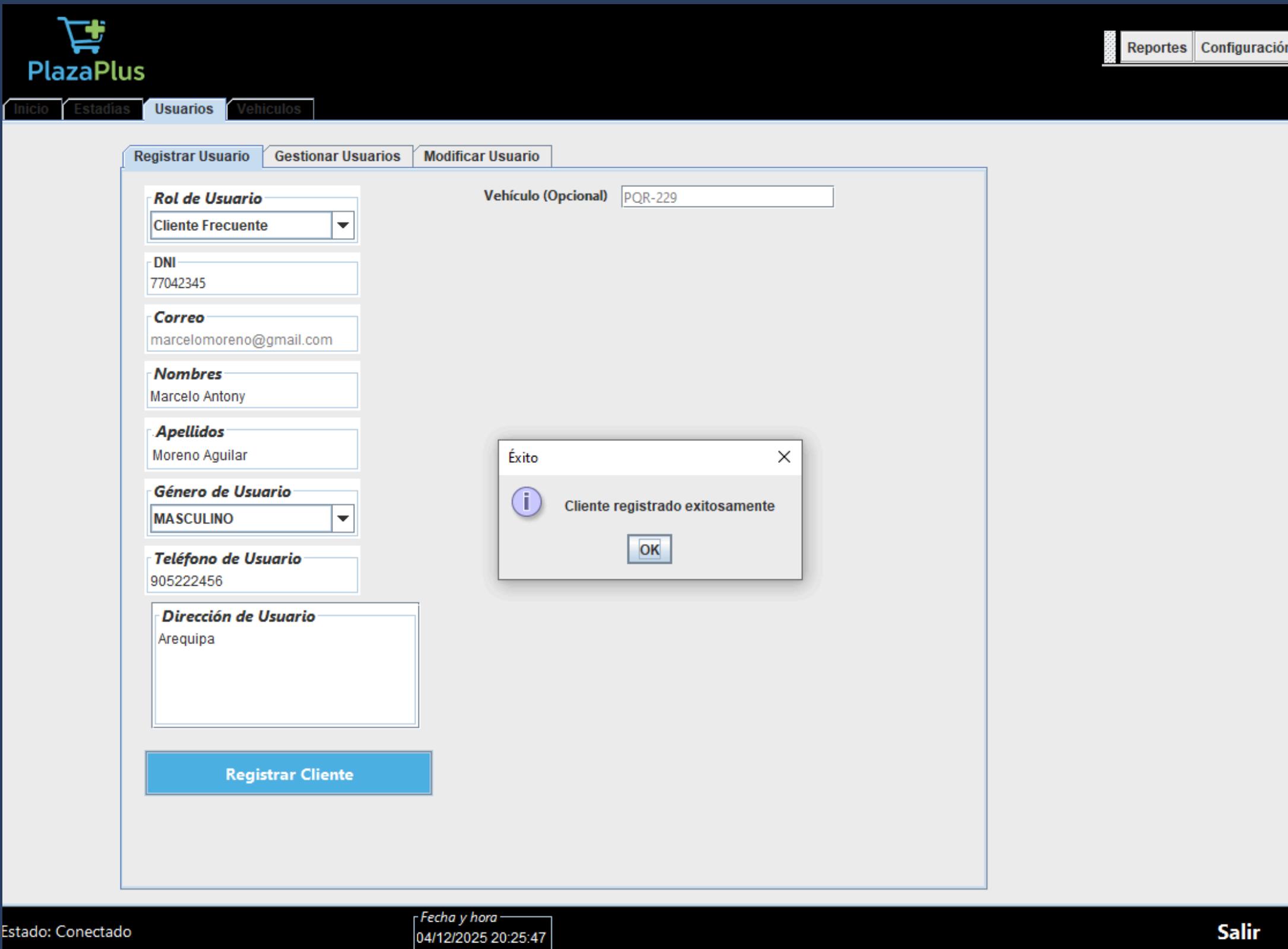
Estructura de Paquetes

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, mantenable y coherente con los principios de diseño orientado a objetos.



USUARIOS

Registrar Usuario



The screenshot shows the PlazaPlus user registration interface. The top navigation bar includes a logo, menu items (Inicio, Estadias, Usuarios, Vehículos), and links for Reports and Configuration. The main content area has tabs for Registering a User, Managing Users, and Modifying a User. The registration form fields are as follows:

- Rol de Usuario: Cliente Frecuente
- Vehículo (Opcional): PQR-229
- DNI: 77042345
- Correo: marcelomoreno@gmail.com
- Nombres: Marcelo Antony
- Apellidos: Moreno Aguilar
- Género de Usuario: MASCULINO
- Teléfono de Usuario: 905222456
- Dirección de Usuario: Arequipa

A blue "Registrar Cliente" button is at the bottom of the form. A modal window titled "Éxito" displays the message "Cliente registrado exitosamente" (Client registered successfully) with an "OK" button.

At the bottom, status messages show "Estado: Conectado" and the current date and time "04/12/2025 20:25:47". A "Salir" (Logout) button is also present.

```

private void registrarCliente() {
    try {
        // Obtener valores de los campos
        String dni = inDni1.getText().trim();
        String nombres = inNombresUsuario1.getText().trim();
        String apellidos = inApellidosUsuario1.getText().trim();
        String correo = inCorreo1.getText().trim();
        String telefono = inTelefonoUsuario1.getText().trim();
        String direccion = inDireccionUsuario1.getText().trim();

        // Obtener género
        String genero = "";
        if (inGeneroUsuario1.getSelectedItem() != null) {
            genero = inGeneroUsuario1.getSelectedItem().toString();
        }

        // Obtener tipo de usuario del rol
        TipoUsuario tipo = TipoUsuario.REGULAR;
        if (inRolUsuario1.getSelectedItem() != null) {
            String rolSeleccionado = inRolUsuario1.getSelectedItem().toString();
            if (rolSeleccionado.contains("Abonado")) {
                tipo = TipoUsuario.ABONADO;
            } else if (rolSeleccionado.contains("Frecuente")) {
                tipo = TipoUsuario.FRECUENTE;
            }
        }

        String placaInicial = inPlaca1.getText().trim();

        // Validaciones
        if (dni.isEmpty() || nombres.isEmpty() || apellidos.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                "DNI, Nombres y Apellidos son obligatorios",
                "Datos incompletos",
                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if (dni.length() != 8) {
            JOptionPane.showMessageDialog(this,
                "El DNI debe tener 8 dígitos",
                "DNI inválido",
                JOptionPane.WARNING_MESSAGE);
            return;
        }

        // Registrar
        boolean exito = controlador.registrarCliente(dni, nombres, apellidos, correo,
            telefono, direccion, genero, tipo, placaInicial);
    }
}

```

```

public boolean registrarCliente(String dni, String nombres, String apellidos,
                                String correo, String telefono, String direccion,
                                String genero, TipoUsuario tipo, String placaInicial) {
    // Validaciones básicas
    if (dni == null || dni.trim().isEmpty() || dni.length() != 8) {
        return false;
    }

    if (nombres == null || nombres.trim().isEmpty()) {
        return false;
    }

    if (apellidos == null || apellidos.trim().isEmpty()) {
        return false;
    }

    Cliente nuevoCliente = new Cliente(dni, nombres, apellidos, correo,
                                         telefono, direccion, genero, tipo);

    // Si proporcionó una placa, agregarla
    if (placaInicial != null && !placaInicial.trim().isEmpty()) {
        nuevoCliente.agregarVehiculo(placaInicial.trim().toUpperCase());
    }

    return gestorPrincipal.registrarCliente(nuevoCliente);
}

```

```

/**
 * Registra un nuevo cliente en el sistema.
 * Usa árbol binario para búsqueda rápida de duplicados.
 */
public boolean registrarCliente(Cliente cliente) {
    if (cliente == null || cliente.getDni() == null) {
        return false;
    }

    // Verificar duplicados usando árbol (O(log n))
    if (buscarCliente(cliente.getDni()) != null) {
        return false;
    }

    arbolClientes.insertar(cliente);

    // Guardar en archivo TXT
    ClienteDAO.guardarCliente(cliente);

    return true;
}

```

Gestionar Usuarios

PlazaPlus

Reportes Configuración

Inicio Estadías Usuarios Vehículos

Registrar Usuario Gestionar Usuarios Modificar Usuario

DNI	Nombre Compl...	Correo	Teléfono	Tipo	Vehículos	Estado
00000000	Cliente Temporal			Cliente Regular	4	ACTIVO
77042345	Marcelo Antony ...	marcelomore...	905222456	Cliente Frecuen...	1	ACTIVO
77201791	Aaron Burgos	aaron@gmail.c...	957356458	Cliente Regular	1	ACTIVO
77201794	Nicolas Ramirez	nicolas@gmail...	907621687	Cliente Regular	1	ACTIVO
77201795	Adrian Ramirez	adrian@gmail.c...	907657728	Cliente Regular	1	ACTIVO

Modificar Desactivar Actualizar

Estado: Conectado Fecha y hora: 04/12/2025 20:26:54

Salir

PlazaPlus

Reportes Configuración

Inicio Estadías Usuarios Vehículos

Registrar Usuario Gestionar Usuarios Modificar Usuario

DNI	Nombre Compl...	Correo	Teléfono	Tipo	Vehículos	Estado
00000000	Cliente Temporal			Cliente Regular	4	ACTIVO
77042345	Marcelo Antony ...	marcelomore...	905222456	Cliente Frecuen...	1	ACTIVO
77201791	Aaron Burgos	aaron@gmail.c...	957356458	Cliente Regular	1	ACTIVO
77201794	Nicolas Ramirez	nicolas@gmail...	907621687	Cliente Regular	1	ACTIVO
77201795	Adrian Ramirez	adrian@gmail.c...	907657728	Cliente Regular	1	ACTIVO

Confirmar eliminación X
¿Desactivar cliente?
Nicolas Ramirez
DNI: 77201794
Yes No

Modificar Desactivar Actualizar

Estado: Conectado Fecha y hora: 04/12/2025 20:28:08

Salir

PlazaPlus

Reportes Configuración

Inicio Estadías Usuarios Vehículos

Registrar Usuario Gestionar Usuarios Modificar Usuario

DNI	Nombre Compl...	Correo	Teléfono	Tipo	Vehículos	Estado
00000000	Cliente Temporal			Cliente Regular	4	ACTIVO
77042345	Marcelo Antony ...	marcelomore...	905222456	Cliente Frecuen...	1	ACTIVO
77201791	Aaron Burgos	aaron@gmail.c...	957356458	Cliente Regular	1	ACTIVO
77201794	Nicolas Ramirez	nicolas@gmail...	907621687	Cliente Regular	1	INACTIVO
77201795	Adrian Ramirez	adrian@gmail.c...	907657728	Cliente Regular	1	ACTIVO

Modificar Desactivar Actualizar

Estado: Conectado Fecha y hora: 04/12/2025 20:28:57

Salir

```
private void cargarClientes() {
    if (controlador == null) return;

    // 1. LIMPIAR TABLA
    modeloTabla.setRowCount(0);

    // 2. OBTENER TODOS LOS CLIENTES DEL ÁRBOL BINARIO (recorrido inorden)
    ListaEnlazada<Cliente> clientes = controlador.obtenerTodosLosClientes();

    // 3. POBLAR TABLA
    for (int i = 0; i < clientes.getTamano(); i++) {
        Cliente c = clientes.get(i);

        // Contar vehículos del cliente
        int cantVehiculos = c.getPlacasVehiculos().getTamano();

        Object[] fila = {
            c.getDni(), // Columna 0: DNI
            c.getNombreCompleto(), // Columna 1: Nombre Completo
            c.getCorreo(), // Columna 2: Correo
            c.getTelefono(), // Columna 3: Teléfono
            c.getTipoUsuario().toString(), // Columna 4: Tipo (REGULAR/FRECUENTE/ABONADO)
            cantVehiculos, // Columna 5: Cantidad de vehículos
            c.isActivo() ? "ACTIVO" : "INACTIVO" // Columna 6: Estado
        };

        modeloTabla.addRow(fila);
    }
}
```

```
private void seleccionarClienteParaModificar() {
    // 1. VERIFICAR QUE HAY FILA SELECCIONADA
    int filaSeleccionada = tablaUsuarios1.getSelectedRow();

    if (filaSeleccionada < 0) {
        JOptionPane.showMessageDialog(this, "Por favor, seleccione un cliente de la tabla");
        return;
    }

    // 2. OBTENER DNI DE LA FILA
    String dni = modeloTabla.getValueAt(filaSeleccionada, 0).toString();

    // 3. BUSCAR CLIENTE COMPLETO EN EL ÁRBOL (O(log n))
    clienteSeleccionado = controlador.buscarPorDni(dni);

    if (clienteSeleccionado != null) {
        // 4. CARGAR DATOS EN FORMULARIO DE MODIFICACIÓN
        cargarDatosParaModificar();

        // 5. CAMBIAR A PESTAÑA "Modificar Usuario"
        jTabbedPane1.setSelectedIndex(2);
    }
}
```

```
public boolean modificarCliente(String dni, String nombres, String apellidos,
                                String correo, String telefono, String direccion,
                                String genero, TipoUsuario tipo) {
    // 1. BUSCAR CLIENTE EN ÁRBOL (O(log n))
    Cliente cliente = gestorPrincipal.buscarCliente(dni);
    if (cliente == null) {
        return false;
    }

    // 2. ACTUALIZAR TODOS LOS CAMPOS
    cliente.setNombres(nombres);
    cliente.setApellidos(apellidos);
    cliente.setCorreo(correo);
    cliente.setTelefono(telefono);
    cliente.setDireccion(direccion);
    cliente.setGenero(genero);
    cliente.setTipoUsuario(tipo);

    // NOTA: Los cambios se propagan automáticamente porque el árbol
    //       contiene referencias a los objetos Cliente originales

    return true;
}
```

```
private void eliminarCliente() {
    // 1. VERIFICAR SELECCIÓN
    int filaSeleccionada = tablaUsuarios1.getSelectedRow();

    if (filaSeleccionada < 0) {
        JOptionPane.showMessageDialog(this, "Por favor, seleccione un cliente");
        return;
    }

    // 2. OBTENER DATOS DE LA FILA
    String dni = modeloTabla.getValueAt(filaSeleccionada, 0).toString();
    String nombre = modeloTabla.getValueAt(filaSeleccionada, 1).toString();

    // 3. CONFIRMAR ACCIÓN
    int confirmacion = JOptionPane.showConfirmDialog(this,
        "¿Desactivar cliente?\n\n" + nombre + "\nDNI: " + dni,
        "Confirmar eliminación",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.WARNING_MESSAGE);

    if (confirmacion == JOptionPane.YES_OPTION) {
        // 4. LLAMAR AL CONTROLADOR
        boolean exito = controlador.eliminarCliente(dni);

        if (exito) {
            JOptionPane.showMessageDialog(this, "Cliente desactivado exitosamente");
            cargarClientes(); // REFRESCAR TABLA (estado cambia de ACTIVO a INACTIVO)
        }
    }
}
```

ARBOL BINARIO USADO

```
/*
 * Busca un cliente por DNI usando el árbol binario (O(log n)).
 */
public Cliente buscarCliente(String dni) {
    return buscarClienteEnArbol(arbolClientes.getRaiz(), dni);
}

private Cliente buscarClienteEnArbol(NodoArbol<Cliente> nodo, String dni) {
    if (nodo == null) {
        return null;
    }

    int comparacion = dni.compareTo(nodo.getDate().getDni());
    if (comparacion == 0) {
        return nodo.getDate();
    } else if (comparacion < 0) {
        return buscarClienteEnArbol(nodo.getIzquierdo(), dni);
    } else {
        return buscarClienteEnArbol(nodo.getDerecho(), dni);
    }
}
```

```
/*
 * Obtiene todos los clientes (recorrido inorden del árbol).
 */
public ListaEnlazada<Cliente> obtenerTodosLosClientes() {
    ListaEnlazada<Cliente> lista = new ListaEnlazada<>();
    recorridoInordenClientes(arbolClientes.getRaiz(), lista);
    return lista;
}

private void recorridoInordenClientes(NodoArbol<Cliente> nodo, ListaEnlazada<Cliente> lista) {
    if (nodo != null) {
        recorridoInordenClientes(nodo.getIzquierdo(), lista);
        lista.agregarAlFinal(nodo.getDate());
        recorridoInordenClientes(nodo.getDerecho(), lista);
    }
}
```

Modificar Usuario

PlazaPlus

Reportes Configuración

Inicio Estadias Usuarios Vehículos

Registrar Usuario Gestionar Usuarios Modificar Usuario

DNI: 77042345 Teléfono: 955234789

Nombres: Marcelo Antony Género: MASCULINO

Apellidos: Moreno Aguilar Tipo de Usuario: Cliente Frecuente

Correo: marcelomoreno@gmail.com

Dirección: Arequipa

Éxito
Cliente modificado exitosamente OK

Guardar Cambios Cancelar

Estado: Conectado 04/12/2025 20:34:12

Salir

PlazaPlus

Reportes Configuración

Inicio Estadias Usuarios Vehículos

Registrar Usuario Gestionar Usuarios Modificar Usuario

DNI	Nombre Compl...	Correo	Teléfono	Tipo	Vehículos	Estado
00000000	Cliente Temporal			Cliente Regular	4	ACTIVO
77042345	Marcelo Antony ...	marcelomoreno...	955234789	Cliente Frecuen...	1	ACTIVO
77201791	Aaron Burgos	aaron@gmail.c...	957356458	Cliente Regular	1	ACTIVO
77201794	Nicolas Ramirez	nicolas@gmail....	907621687	Cliente Regular	1	INACTIVO
77201795	Adrian Ramirez	adrian@gmail.c...	907657728	Cliente Regular	1	ACTIVO

Modificar Desactivar Actualizar

Estado: Conectado 04/12/2025 20:35:24

Salir

```
● ● ●

private void guardarCambiosCliente() {
    if (clienteSeleccionado == null) {
        JOptionPane.showMessageDialog(this, "No hay cliente seleccionado");
        return;
    }

    // 1. CAPTURAR DATOS DEL FORMULARIO
    String nombres = txtNombresMod.getText().trim();
    String apellidos = txtApellidosMod.getText().trim();
    String correo = txtCorreoMod.getText().trim();
    String telefono = txtTelefonoMod.getText().trim();
    String direccion = txtDireccionMod.getText().trim();
    String genero = cboGeneroMod.getSelectedItem().toString();

    // 2. CONVERTIR TIPO DE USUARIO
    TipoUsuario tipo = TipoUsuario.REGULAR;
    String tipoSeleccionado = cboTipoMod.getSelectedItem().toString();
    if (tipoSeleccionado.contains("Abonado")) {
        tipo = TipoUsuario.ABONADO;
    } else if (tipoSeleccionado.contains("Frecuente")) {
        tipo = TipoUsuario.FRECUENTE;
    }

    // 3. VALIDACIONES
    if (nombres.isEmpty() || apellidos.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Nombres y Apellidos son obligatorios");
        return;
    }

    // 4. LLAMAR AL CONTROLADOR
    boolean exito = controlador.modificarCliente(
        clienteSeleccionado.getDni(),
        nombres, apellidos, correo,
        telefono, direccion, genero, tipo
    );

    if (exito) {
        JOptionPane.showMessageDialog(this, "Cliente modificado exitosamente");
        clienteSeleccionado = null;
        cargarClientes(); // REFRESCAR TABLA
        jTabbedPane1.setSelectedIndex(1); // VOLVER A "GESTIONAR"
    }
}
```

VEHICULOS

Registrar Vehiculo

The screenshot shows the PlazaPlus application interface. At the top, there is a dark header bar with the PlazaPlus logo (a shopping cart icon with a plus sign) and the text "PlazaPlus". Below the logo are navigation tabs: Inicio, Estadias, Usuarios, and Vehiculos (which is currently selected). On the right side of the header are links for Reportes and Configuración. The main content area has a title bar with three tabs: "Registrar Vehiculo" (selected), "Características", and "Tipo de Vehiculo". The "Registrar Vehiculo" tab contains several input fields: "Tipo de Vehículo:" dropdown set to "AUTO", "Placa:" text input "PQR-229", "Marca:" dropdown set to "Honda", "Modelo:" dropdown set to "Van", "Color:" dropdown set to "Rojo", and "Propietario (DNI):" text input "123456789". A blue button labeled "Registrar V" is visible. A modal dialog box titled "Éxito" is displayed, containing the message "Vehículo registrado exitosamente" and an "OK" button. At the bottom of the screen, there is a footer bar with the text "Estado: Conectado", the date and time "04/12/2025 20:39:59", and a "Salir" button.

```
private void registrarVehiculo() {
    try {
        String placa = inPlaca1.getText().trim().toUpperCase();
        TipoVehiculo tipo = (TipoVehiculo) inTipoVehiculo1.getSelectedItem();

        // Obtener marca, modelo, color del ComboBox
        String marca = inMarcaVehiculo1.getSelectedItem() != null ?
            inMarcaVehiculo1.getSelectedItem().toString().trim() : "";
        String modelo = inModeloVehiculo1.getSelectedItem() != null ?
            inModeloVehiculo1.getSelectedItem().toString().trim() : "";
        String color = inColorVehiculo1.getSelectedItem() != null ?
            inColorVehiculo1.getSelectedItem().toString().trim() : "";
        String dniPropietario = inDni1.getText().trim();

        // Validaciones
        if (placa.isEmpty() || marca.isEmpty() || modelo.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                "Placa, Marca y Modelo son obligatorios",
                "Datos incompletos",
                JOptionPane.WARNING_MESSAGE);
            return;
        }

        boolean exito = controlador.registrarVehiculo(placa, tipo, dniPropietario,
                                                       marca, modelo, color);

        if (exito) {
            JOptionPane.showMessageDialog(this,
                "Vehículo registrado exitosamente",
                "Éxito",
                JOptionPane.INFORMATION_MESSAGE);

            limpiarCamposRegistro();
            cargarVehiculos();
        } else {
            JOptionPane.showMessageDialog(this,
                "Error: La placa ya está registrada",
                "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this,
            "Error al registrar vehículo: " + ex.getMessage(),
            "Error",
            JOptionPane.ERROR_MESSAGE);
    }
}
```

```
public boolean registrarVehiculo(String placa, TipoVehiculo tipo, String dniPropietario,
                                 String marca, String modelo, String color) {
    // Validaciones básicas
    if (placa == null || placa.trim().isEmpty() || placa.length() < 6) {
        return false;
    }
    if (tipo == null) {
        return false;
    }
    if (dniPropietario == null || dniPropietario.trim().isEmpty()) {
        return false;
    }
    Vehiculo nuevoVehiculo = new Vehiculo(placa.trim().toUpperCase(), tipo,
                                            dniPropietario.trim(), marca, modelo, color);

    // GestorPrincipal valida que el propietario exista
    return gestorPrincipal.registrarVehiculo(nuevoVehiculo);
}
```

```
public boolean registrarVehiculo(Vehiculo vehiculo) {
    if (vehiculo == null || vehiculo.getPlaca() == null) {
        return false;
    }

    // Verificar que no exista la placa
    if (buscarVehiculo(vehiculo.getPlaca()) != null) {
        return false;
    }

    // Verificar que el cliente propietario exista
    Cliente propietario = buscarCliente(vehiculo.getPropietario());
    if (propietario == null) {
        return false; // No se puede registrar vehículo sin cliente
    }

    arbolVehiculos.insertar(vehiculo);
    propietario.agregarVehiculo(vehiculo.getPlaca());

    // Guardar vehículo en archivo TXT
    VehiculoDAO.guardarVehiculo(vehiculo);

    // Actualizar cliente en archivo TXT
    ClienteDAO.actualizarCliente(propietario);

    return true;
}
```



```
/**  
 * Busca un vehículo por placa usando el árbol binario ( $O(\log n)$ ).  
 */  
public Vehiculo buscarVehiculo(String placa) {  
    return buscarVehiculoEnArbol(arbolVehiculos.getRaiz(), placa);  
}  
  
private Vehiculo buscarVehiculoEnArbol(NodoArbol<Vehiculo> nodo, String placa) {  
    if (nodo == null) {  
        return null;  
    }  
  
    int comparacion = placa.toUpperCase().compareTo(nodo.getDato().getPlaca());  
    if (comparacion == 0) {  
        return nodo.getDato();  
    } else if (comparacion < 0) {  
        return buscarVehiculoEnArbol(nodo.getIzquierdo(), placa);  
    } else {  
        return buscarVehiculoEnArbol(nodo.getDerecho(), placa);  
    }  
}
```

Características

The screenshot displays the PlazaPlus software interface, specifically the 'Vehículos' (Vehicles) module. The top navigation bar includes links for Inicio, Estadías, Usuarios, Vehículos, Reportes, and Configuración. The main content area shows a table of registered vehicles:

Placa	Tipo	Marca	Modelo	Color	Propietario (D...)
LOE-654	AUTO				00000000
LOE-685	AUTO	Kia	SUV	Gris	77201795
LOL-777	AUTO	Toyota	Pickup	Gris	77201794
MAN-111	MINIVAN	Hyundai	Hatchback	Gris	77201791
PQR-229	AUTO	Honda	Van	Rojo	77042345
...

At the bottom of the screen, there is a status bar showing 'Estado: Conectado', the date and time '04/12/2025 20:41:09', and a 'Salir' (Logout) button.

Tipo de Vehiculo

The screenshot displays the PlazaPlus software interface, specifically the 'Vehiculos' (Vehicles) section. The top navigation bar includes links for Inicio, Estadías, Usuarios, Vehiculos (which is highlighted in blue), Reportes, and Configuración. Below the navigation is a sub-menu with options: Registrar Vehiculo, Características, and Tipo de Vehiculo. A main content area titled 'Estadísticas de Vehículos Registrados' (Registered Vehicle Statistics) shows the following data:
ESTADÍSTICAS DE VEHÍCULOS
Total: 6 | Bicicletas: 0 | Motos: 0 | Autos: 4 | SUVs: 1 | Minivans: 1
A blue button labeled 'Ver Estadísticas' (View Statistics) is located at the bottom of this section. At the very bottom of the screen, there is a status bar with the text 'Estado: Conectado' (Status: Connected), the date and time 'Fecha y hora 04/12/2025 20:41:47', and a 'Salir' (Exit) button.

ESTADÍAS

Tickets - Registrar Ticket

 PlazaPlus

Inicio Estadias Usuarios Vehículos

Ingresos Egresos Tickets

Reportes Configuración

Panel Tickets

Registrar Ticket Buscar Eliminar Modificar

DNI Conductor: DV

Placa Vehículo: Tipo Vehículo:

Registrar Ingreso

Éxito

i Ingreso registrado exitosamente
Plaza: P1-A1

OK

Criterio de Ordena... Algoritmo de ...

Por Placa Bubble Sort Ordenar

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza
T-1004	PQR-229	AUTO	2025-12-04 21:02:...	P1-A1

Estadísticas Deshacer Cola Espera

Estado: Conectado Fecha y hora 04/12/2025 21:02:23

Salir

Panel tickets

```
● ● ●  
private void btnRegistrarIngresoActionPerformed(java.awt.event.ActionEvent evt) {  
    // 1. CAPTURAR DATOS  
    String placa = inPlaca1.getText().trim().toUpperCase();  
    TipoVehiculo tipo = (TipoVehiculo) inTipoVehiculo1.getSelectedItem();  
  
    // 2. DELEGAR A MÉTODO DE NEGOCIO  
    manejarRegistroIngreso(placa, tipo);  
  
    // 3. LIMPIAR CAMPO  
    inPlaca1.setText("");  
}
```

```
● ● ●  
public void manejarRegistroIngreso(String placa, TipoVehiculo tipo) {  
    // 1. VALIDAR CONTROLADOR  
    if (controlador == null) {  
        JOptionPane.showMessageDialog(this, "Error: Controlador no inicializado");  
        return;  
    }  
  
    // 2. CASO ESPECIAL: BICICLETA SIN PLACA  
    if (tipo == TipoVehiculo.BICICLETA && placa.isEmpty()) {  
        placa = "BICI-" + System.currentTimeMillis() % 10000;  
    } else if (placa == null || placa.trim().isEmpty()) {  
        JOptionPane.showMessageDialog(this, "Debe ingresar una placa válida");  
        return;  
    }  
  
    // 3. LLAMAR AL CONTROLADOR  
    Ticket ticketGenerado = controlador.registrarIngresoVehiculo(placa.toUpperCase(), tipo);  
  
    // 4. PROCESAR RESULTADO  
    if (ticketGenerado != null) {  
        // HAY PLAZA DISPONIBLE - REGISTRAR EN TABLA  
        agregarFilaATabla(ticketGenerado);  
  
        JOptionPane.showMessageDialog(this,  
            "Ingreso registrado exitosamente\nPlaza: P"  
            + ticketGenerado.getPlaza().getPiso()  
            + "-" + ticketGenerado.getPlaza().getSector()  
            + ticketGenerado.getPlaza().getNumero(),  
            "Éxito");  
    } else {  
        // NO HAY PLAZA - VEHÍCULO EN COLA DE ESPERA  
        JOptionPane.showMessageDialog(this,  
            "No hay plazas disponibles. Vehículo agregado a lista de espera.",  
            "Plaza no disponible");  
    }  
}
```

Controlador

```
● ● ●  
public Ticket registrarIngresoVehiculo(String placa, TipoVehiculo tipo) {  
    // 1. VALIDACIÓN BÁSICA  
    if (placa == null || placa.trim().isEmpty()) {  
        System.err.println("La placa no puede estar vacía.");  
        return null;  
    }  
  
    // 2. DELEGAR A GESTOR PRINCIPAL  
    // Usa DNI "00000000" para registro rápido (cliente temporal)  
    return gestorPrincipal.registrarIngreso(placa, tipo, "00000000");  
}
```

DAO

```
● ● ●  
public static boolean guardarTicket(Ticket ticket) {  
    // Formato: código|placa|tipo|horaIngreso|horaSalida|plaza|monto|pagado|estado  
    String linea = String.format("%s|%s|%s|%s|%s|%s|%.2f|%b|%s",  
        ticket.getCodigo(),  
        ticket.getVehiculo().getPlaca(),  
        ticket.getVehiculo().getTipo(),  
        ticket.getHoraIngreso(),  
        ticket.getHoraSalida(),  
        "P" + ticket.getPlaza().getPiso() + "-" + ticket.getPlaza().getSector() +  
        ticket.getPlaza().getNumero(),  
        ticket.getMontoAPagar(),  
        ticket.isPagado(),  
        ticket.getEstado());  
    );  
  
    return ManejadorArchivos.agregarLinea("tickets.txt", linea);  
}
```

Gestor Principal



```
public Ticket registrarIngreso(String placa, TipoVehiculo tipo, String dniPropietario) {  
    placa = placa.toUpperCase();  
  
    // ===== PASO 1: BUSCAR O REGISTRAR VEHÍCULO =====  
    Vehiculo vehiculo = buscarVehiculo(placa); // Búsqueda en árbol O(log n)  
    if (vehiculo == null) {  
        // Vehículo NO EXISTE → crear uno temporal  
        vehiculo = new Vehiculo(placa, tipo, dniPropietario, LocalDateTime.now());  
        arbolVehiculos.insertar(vehiculo); // Insertar en árbol binario  
    }  
  
    // ===== PASO 2: BUSCAR O REGISTRAR CLIENTE =====  
    Cliente cliente = buscarCliente(dniPropietario); // Búsqueda en árbol O(log n)  
    if (cliente == null) {  
        // Cliente NO EXISTE → crear uno temporal (REGULAR)  
        cliente = new Cliente(dniPropietario, "Cliente", "Temporal",  
                               "", "", "", "", TipoUsuario.REGULAR);  
        arbolClientes.insertar(cliente); // Insertar en árbol binario  
    }  
  
    // Asociar vehículo al cliente  
    if (!cliente.tieneVehiculo(placa)) {  
        cliente.agregarVehiculo(vehiculo);  
    }  
  
    // ===== PASO 3: VERIFICAR QUE NO ESTÉ DENTRO =====  
    if (buscarTicketActivo(placa) != null) {  
        return null; // Ya tiene un ticket activo (está adentro)  
    }  
  
    // ===== PASO 4: BUSCAR PLAZA DISPONIBLE =====  
    Plaza plazaAsignada = buscarPlazaLibre(tipo);  
  
    if (plazaAsignada != null) {  
        // ✅ HAY PLAZA DISPONIBLE  
  
        // 4.1. Actualizar estado de la plaza  
        plazaAsignada.setVehiculoActual(vehiculo);  
        plazaAsignada.setEstado(EstadoPlaza.OCUPADA);  
        plazaAsignada.setHoraOcupacion(LocalDateTime.now());  
  
        // 4.2. Crear ticket  
        String codigoTicket = generarCodigoTicket(); // Genera "T-1004", "T-1005", etc.  
        Ticket ticket = new Ticket(codigoTicket, vehiculo, plazaAsignada,  
                                   LocalDateTime.now(), null, 0.0, false);  
    }  
}
```

```
// 4.3. Registrar en ESTRUCTURAS DE DATOS  
ticketsActivos.agregarAlFinal(ticket); // Lista enlazada  
historialTickets.apilar(ticket); // Pila (para deshacer)  
  
// 4.4. PERSISTIR EN ARCHIVO TXT  
TicketDAO.guardarTicket(ticket);  
VehiculoDAO.guardarVehiculo(vehiculo);  
ClienteDAO.guardarCliente(cliente);  
  
return ticket; // ✅ ÉXITO  
  
} else {  
    // ❌ NO HAY PLAZA DISPONIBLE  
  
    // Agregar a COLA DE ESPERA (FIFO)  
    SolicitudEspera solicitud = new SolicitudEspera(placa, tipo, LocalDateTime.now());  
    colaEspera.encolar(solicitud);  
  
    return null; // NULL indica que no hay plaza  
}  
}
```



```
private Plaza buscarPlazaLibre(TipoVehiculo tipo) {  
    // Recorre TODAS las plazas (ListaEnlazada<Plaza>)  
    for (int i = 0; i < plazas.getTamano(); i++) {  
        Plaza plaza = plazas.get(i);  
  
        // Busca la PRIMERA plaza con estado LIBRE  
        if (plaza.getEstado() == EstadoPlaza.LIBRE) {  
            return plaza; // Retorna inmediatamente  
        }  
    }  
  
    return null; // No hay plazas libres  
}
```

Ingresos



PlazaPlus

Reportes Configuración

Inicio Estadias Usuarios Vehículos

Ingresos Egresos Tickets

Historial Completo de Ingresos

Total de vehículos: 5

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza	Estado
T-1000	LOE-654	AUTO	04/12/2025 01:15:29	P1-B1	FINALIZADO
T-1001	XLA-123	SUV	04/12/2025 01:41:10	P1-D1	FINALIZADO
T-1002	LOL-777	AUTO	04/12/2025 16:47:50	P1-A1	FINALIZADO
T-1003	MAN-111	MINIVAN	04/12/2025 17:28:22	P1-A1	FINALIZADO
T-1004	PQR-229	AUTO	04/12/2025 21:02:15	P1-A1	ACTIVO

Actualizar

Estado: Conectado

Fecha y hora
04/12/2025 21:04:24

Salir

● ● ●

```
private void inicializarTabla() {
    modeloTabla = new DefaultTableModel(
        new Object[][]{},
        new String[]{"ID Ticket", "Placa", "Tipo Veh\u00edculo", "Fecha/Hora Ingreso", "Plaza", "Estado"}
    );
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
    tablaHistorial.setModel(modeloTabla);
}
```

● ● ●

```
private void actualizarManualmente() {
    cargarDatos();
    javax.swing.JOptionPane.showMessageDialog(this,
        "Datos actualizados correctamente",
        "Actualizado",
        javax.swing.JOptionPane.INFORMATION_MESSAGE);
}
```

● ● ●

```
private void cargarDatos() {
    if (controlador == null) {
        return;
    }
    modeloTabla.setRowCount(0);

    // Obtener TODOS los tickets (historial completo)
    ListaEnlazada<modelo.entidades.Ticket> todosLosTickets = controlador.getTodosLosTickets();

    for (int i = 0; i < todosLosTickets.getTamano(); i++) {
        Ticket ticket = todosLosTickets.get(i);

        String fechaHora = ticket.getHoraIngreso().format(
            java.time.format.DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss")
        );

        String plaza = "P" + ticket.getPlaza().getPiso() +
            "-" + ticket.getPlaza().getSector() +
            ticket.getPlaza().getNumero();

        String estado = ticket.getEstado() == modelo.entidades.Ticket.EstadoTicket.ACTIVO
            ? "ACTIVO" : "FINALIZADO";

        modeloTabla.addRow(new Object[]{
            ticket.getCodigo(),
            ticket.getVehiculo().getPlaca(),
            ticket.getVehiculo().getTipo(),
            fechaHora,
            plaza,
            estado
        });
    }

    lblCantidad.setText("Total de veh\u00edculos: " + todosLosTickets.getTamano());
}
```

```
Private void cargarDatosDesdeArchivos() {
    System.out.println("Cargando datos desde archivos TXT...");

    // Cargar clientes
    ListaEnlazada<Cliente> clientes = ClienteDAO.cargarTodos();
    for (int i = 0; i < clientes.getTamano(); i++) {
        arbolClientes.insertar(clientes.get(i));
    }
    System.out.println("Clientes cargados: " + clientes.getTamano());

    // Cargar vehículos
    ListaEnlazada<Vehiculo> vehiculos = VehiculoDAO.cargarTodos();
    for (int i = 0; i < vehiculos.getTamano(); i++) {
        arbolVehiculos.insertar(vehiculos.get(i));
    }
    System.out.println("Vehículos cargados: " + vehiculos.getTamano());

    // Cargar tickets
    ListaEnlazada<Ticket> tickets = TicketDAO.cargarTodos();
    for (int i = 0; i < tickets.getTamano(); i++) {
        Ticket ticket = tickets.get(i);
        ticketsActivos.agregarAlFinal(ticket);

        // Si el ticket está activo, ocupar la plaza
        if (ticket.getEstado() == EstadoTicket.ACTIVO) {
            Plaza plaza = ticket.getPlaza();
            plaza.setEstado(EstadoPlaza.OCUPADA);
            plaza.setVehiculoActual(ticket.getVehiculo());
            plaza.setHoraOcupacion(ticket.getHoraIngreso());
        }
    }
    System.out.println("Tickets cargados: " + tickets.getTamano());

    // Ajustar contador de tickets
    if (tickets.getTamano() > 0) {
        // Buscar el código más alto
        int maxCodigo = 1000;
        for (int i = 0; i < tickets.getTamano(); i++) {
            String codigo = tickets.get(i).getCodigo();
            if (codigo.startsWith("T-")) {
                try {
                    int num = Integer.parseInt(codigo.substring(2));
                    if (num >= maxCodigo) {
                        maxCodigo = num + 1;
                    }
                } catch (NumberFormatException e) {
                    // Ignorar códigos inválidos
                }
            }
        }
        contadorTickets = maxCodigo;
    }
    System.out.println("Datos cargados exitosamente desde archivos TXT.");
}
```

Egresos

The screenshot shows the PlazaPlus software interface with the following details:

- Header:** PlazaPlus logo, Reports, Configuration.
- Top Navigation:** Inicio, Estadias, Usuarios, Vehículos, Ingresos, Egresos, Tickets.
- Main Title:** Registrar Egreso de Vehículos.
- Message:** Vehículos en estacionamiento: 1. Seleccione un vehículo de la lista y haga clic en Registrar Egreso.
- List:** [T-1004] PQR-229 - AUTO - Plaza: P1-A1 - Ingreso: 04/12 21:02
- Buttons:** Actualizar, Registrar Egreso (highlighted in red).
- Status Bar:** Estado: Conectado, Fecha y hora: 04/12/2025 21:05:58.
- Alert:** Actualizado: Lista actualizada correctamente (OK button).

The screenshot shows the PlazaPlus software interface with the following details:

- Header:** PlazaPlus logo, Reports, Configuration.
- Top Navigation:** Inicio, Estadias, Usuarios, Vehículos, Ingresos, Egresos, Tickets.
- Title:** Registrar Egreso de Vehículos.
- Message:** Vehículos en estacionamiento: 1. Seleccione un vehículo de la lista y haga clic en Registrar Egreso.
- List:** [T-1004] PQR-229 - AUTO - Plaza: P1-A1 - Ingreso: 04/12 21:02
- Modal:** Confirmar Egreso
¿Registrar egreso del vehículo?
Placa: PQR-229
Tipo: AUTO
Plaza: P1-A1
Yes No
- Buttons:** Registrar Egreso (highlighted in red).
- Status Bar:** Estado: Conectado, Fecha y hora: 04/12/2025 21:06:39.

The screenshot shows the PlazaPlus software interface with the following details:

- Header:** PlazaPlus logo, Reports, Configuration.
- Top Navigation:** Inicio, Estadias, Usuarios, Vehículos, Ingresos, Egresos, Tickets.
- Title:** Historial Completo de Ingresos.
- Message:** Total de vehículos: 5
- Table:** A table listing vehicle entries with columns: ID Ticket, Placa, Tipo Vehículo, Fecha/Hora Ingreso, Plaza, and Estado. The data is as follows:

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza	Estado
T-1000	LOE-654	AUTO	04/12/2025 01:15:29	P1-B1	FINALIZADO
T-1001	XLA-123	SUV	04/12/2025 01:41:10	P1-D1	FINALIZADO
T-1002	LOL-777	AUTO	04/12/2025 16:47:50	P1-A1	FINALIZADO
T-1003	MAN-111	MINIVAN	04/12/2025 17:28:22	P1-A1	FINALIZADO
T-1004	PQR-229	AUTO	04/12/2025 21:02:15	P1-A1	FINALIZADO
- Buttons:** Actualizar, Salir.
- Status Bar:** Estado: Conectado, Fecha y hora: 04/12/2025 21:08:38.



```
private void cargarVehiculosActivos() {
    if (controlador == null) return;

    // 1. LIMPIAR LISTA VISUAL
    modeloLista.clear();

    // 2. OBTENER TICKETS ACTIVOS DEL CONTROLADOR
    ticketsActivos = controlador.getTicketsActivos();

    // 3. VERIFICAR SI HAY VEHÍCULOS
    if (ticketsActivos.getTamano() == 0) {
        modeloLista.addElement("No hay vehículos en el estacionamiento");
        btnRegistrarEgreso.setEnabled(false); // Deshabilitar botón
        return;
    }

    // 4. POBLAR LISTA CON TICKETS ACTIVOS
    btnRegistrarEgreso.setEnabled(true);

    for (int i = 0; i < ticketsActivos.getTamano(); i++) {
        Ticket ticket = ticketsActivos.get(i);

        // Formatear plaza (P1-A5)
        String plaza = "P" + ticket.getPlaza().getPiso() +
                      "-" + ticket.getPlaza().getSector() +
                      ticket.getPlaza().getNumero();

        // Formatear fecha de ingreso (04/12 21:02)
        String fechaIngreso = ticket.getHoraIngreso().format(
            java.time.format.DateTimeFormatter.ofPattern("dd/MM HH:mm"))
    };

    // Crear ítem de la lista
    String item = String.format("[%s] %s - %s - Plaza: %s - Ingreso: %s",
        ticket.getCodigo(), // T-1004
        ticket.getVehiculo().getPlaca(), // PQB-229
        ticket.getVehiculo().getTipo(), // AUTO
        plaza, // P1-A3
        fechaIngreso // 04/12 21:02
    );

    modeloLista.addElement(item);
}

// 5. ACTUALIZAR CONTADOR
lblCantidad.setText("Vehículos en estacionamiento: " + ticketsActivos.getTamano());
}
```

Panel Egresos

```
private void manejarRegistroEgreso() {
    // 1. VERIFICAR QUE HAY SELECCIÓN
    int indiceSeleccionado = listaVehiculos.getSelectedIndex();

    if (indiceSeleccionado < 0) {
        JOptionPane.showMessageDialog(this, "Por favor, seleccione un vehículo de la lista");
        return;
    }

    // 2. OBTENER TICKET SELECCIONADO
    Ticket ticketSeleccionado = ticketsActivos.get(indiceSeleccionado);

    // 3. CALCULAR TIEMPO DE ESTANCIA (PREVIEW)
    java.time.Duration duracion = java.time.Duration.between(
        ticketSeleccionado.getHoraIngreso(),
        java.time.LocalDateTime.now()
    );
    long horas = duracion.toHours();
    long minutos = duracion.toMinutes() % 60;
    String tiempoEstancia = String.format("%d hora(s) %d minuto(s)", horas, minutos);

    // 4. MOSTRAR DIÁLOGO DE CONFIRMACIÓN
    int confirmacion = JOptionPane.showConfirmDialog(this,
        String.format("¿Registrar egreso del vehículo?\n\n" +
            "Placa: %s\n" +
            "Tipo: %s\n" +
            "Plaza: P%d-%c%d\n" +
            "Tiempo: %s",
            ticketSeleccionado.getVehiculo().getPlaca(),
            ticketSeleccionado.getVehiculo().getTipo(),
            ticketSeleccionado.getPlaza().getPiso(),
            ticketSeleccionado.getPlaza().getSector(),
            ticketSeleccionado.getPlaza().getNumero(),
            tiempoEstancia),
        "Confirmar Egreso",
        JOptionPane.YES_NO_OPTION);

    // 5. SI CONFIRMA, REGISTRAR EGRESO
    if (confirmacion == JOptionPane.YES_OPTION) {
        Ticket ticketProcesado = controlador.registrarEgreso(ticketSeleccionado);

        if (ticketProcesado != null) {
            // ÉXITO - Mostrar monto calculado
            JOptionPane.showMessageDialog(this,
                String.format("Egreso registrado exitosamente\n\n" +
                    "Plaza liberada: P%d-%c%d\n" +
                    "Tiempo total: %s\n" +
                    "Monto a pagar: S/. %.2f",
                    ticketProcesado.getPlaza().getPiso(),
                    ticketProcesado.getPlaza().getSector(),
                    ticketProcesado.getPlaza().getNumero(),
                    tiempoEstancia,
                    ticketProcesado.getMontoAPagar()), // 💰 MONTO CALCULADO
                "Éxito");
        }
    }

    // 6. REFRESCAR LISTA (quitar vehículo que salió)
    cargarVehiculosActivos();
} else {
    // ERROR
    JOptionPane.showMessageDialog(this, "Error al registrar el egreso", "Error");
}
}
```

Controlador

```
● ● ●  
public Ticket registrarEgreso(Ticket ticket) {  
    // 1. VALIDAR TICKET  
    if (ticket == null) {  
        return null;  
    }  
  
    try {  
        // 2. DELEGAR A GESTOR PRINCIPAL  
        // Pasa la placa del vehículo  
        Ticket ticketSalida = gestorPrincipal.registrarSalida(  
            ticket.getVehiculo().getPlaca());  
        // 3. RETORNAR TICKET PROCESADO (con monto calculado)  
        return ticketSalida;  
  
    } catch (Exception e) {  
        System.err.println("Error al registrar egreso: " + e.getMessage());  
        return null;  
    }  
}
```

Gestor Principal

```
● ● ●  
private Ticket buscarTicketActivo(String placa) {  
    // Recorre la lista de tickets activos  
    for (int i = 0; i < ticketsActivos.getTamano(); i++) {  
        Ticket t = ticketsActivos.get(i);  
  
        // Busca por placa (case-insensitive)  
        if (t.getVehiculo().getPlaca().equalsIgnoreCase(placa)) {  
            return t;  
        }  
    }  
  
    return null; // No encontrado  
}
```

Gestor Principal

```
● ● ●  
public Ticket registrarSalida(String placa) {  
    placa = placa.toUpperCase();  
  
    // ===== PASO 1: BUSCAR TICKET ACTIVO =====  
    Ticket ticket = buscarTicketActivo(placa);  
    if (ticket == null) {  
        return null; // No existe ticket activo para esa placa  
    }  
  
    // ===== PASO 2: REGISTRAR HORA DE SALIDA =====  
    ticket.setHoraSalida(LocalDateTime.now());  
  
    // ===== PASO 3: CALCULAR MONTO =====  
    // Usa: tiempo × tarifa_tipo_vehículo × descuento_tipo_usuario  
    double monto = calcularMonto(ticket);  
    ticket.setMontoAPagar(monto);  
  
    // ===== PASO 4: CAMBIAR ESTADO A FINALIZADO =====  
    ticket.setEstado(EstadoTicket.FINALIZADO);  
  
    // ===== PASO 5: PERSISTIR EN ARCHIVO TXT =====  
    TicketDAO.actualizarTicket(ticket);  
  
    // ===== PASO 6: LIBERAR PLAZA =====  
    Plaza plaza = ticket.getPlaza();  
    if (plaza != null) {  
        plaza.setEstado(EstadoPlaza.LIBRE); // Estado: OCUPADA → LIBRE  
        plaza.setVehiculoActual(null); // Quitar vehículo  
        plaza.setHoraOcupacion(null); // Limpiar hora  
    }  
  
    // ===== PASO 7: PROCESAR COLA DE ESPERA =====  
    // Si hay vehículos esperando, desencolar el siguiente (FIFO)  
    if (!colaEspera.estaVacia()) {  
        SolicitudEspera siguiente = colaEspera.desencolar();  
        System.out.println("Plaza liberada. Siguiente en espera: " + siguiente.getPlaca());  
        // Aquí se podría auto-asignar la plaza al siguiente  
    }  
  
    return ticket; // Retorna ticket con monto calculado  
}
```

Gestor Principal

```
● ● ●

private double calcularMonto(Ticket ticket) {
    LocalDateTime ingreso = ticket.getHoraIngreso();
    LocalDateTime salida = ticket.getHoraSalida();

    // 1. CALCULAR TIEMPO EN MINUTOS
    long minutos = Duration.between(ingreso, salida).toMinutes();

    // 2. CONVERTIR A HORAS (redondear hacia arriba, mínimo 1 hora)
    double horas = Math.max(1, Math.ceil(minutos / 60.0));

    // 3. OBTENER TARIFA BASE SEGÚN TIPO DE VEHÍCULO
    TipoVehiculo tipo = ticket.getVehiculo().getTipo();
    double tarifaPorHora = obtenerTarifa(tipo);
    /*
        BICICLETA: S/. 1.0/hora
        MOTO:      S/. 2.0/hora
        AUTO:      S/. 3.5/hora
        SUV:       S/. 4.0/hora
        MINIVAN:   S/. 4.5/hora
    */

    // 4. APLICAR DESCUENTO SEGÚN TIPO DE USUARIO
    Vehiculo vehiculo = ticket.getVehiculo();
    Cliente cliente = buscarCliente(vehiculo.getPropietario());

    if (cliente != null) {
        switch (cliente.getTipoUsuario()) {
            case ABONADO:
                tarifaPorHora *= 0.8; // 20% descuento
                break;
            case FRECUENTE:
                tarifaPorHora *= 0.9; // 10% descuento
                break;
            case REGULAR:
                // Sin descuento
                break;
        }
    }

    // 5. CALCULAR MONTO FINAL (redondear a 2 decimales)
    return Math.round(horas * tarifaPorHora * 100.0) / 100.0;
}
```

Tickets - Buscar

PlazaPlus

Panel Tickets

Algoritmo de Búsqueda: Búsqueda en Árbol

Criterio de Búsqueda: Por Placa

Buscar

Resultado de Búsqueda

✓ Ticket encontrado:

Código: T-1005
Placa: OFA-223
Tipo: MOTO
Fecha/Hora: 2025-12-04T21:22:56.986563200
Plaza: P1-A1

Tiempos de búsqueda:
• Árbol Binario: 0.007200 ms
• Búsqueda Secuencial: 0.004700 ms

Diferencia: -0.002500 ms (0.65x más rápido)

OK

Criterio de Ordenamiento: Por Placa

Algoritmo de Ordenamiento: Bubble Sort

ID Ticket | Placa | Tipo Vehículo | Fecha/Hora Ingreso | Plaza

T-1004	PQR-229	AUTO	2025-12-04 21:02:...	P1-A1
T-1005	OFA-223	MOTO	2025-12-04 21:22:...	P1-A1

Estadísticas **Deshacer** **Cola Espera**

Estado: Conectado **Fecha y hora:** 04/12/2025 21:23:41

Salir

Criterio de Ordenamiento: Por Placa

Algoritmo de Ordenamiento: Bubble Sort

Ordenar

Por Placa

Por Tipo de Vehículo

Por Fecha/Hora

ID Ticket | Placa | Tipo Vehículo | Fecha/Hora Ingreso | Plaza

T-1004	PQR-229	MOTO	2025-12-04 21:22:...	P1-A1
T-1005	OFA-223	AUTO	2025-12-04 21:02:...	P1-A1

Estadísticas **Deshacer** **Cola Espera**

Criterio de Ordenamiento: Por Placa

Algoritmo de Ordenamiento: Bubble Sort

Ordenar

Bubble Sort

Selection Sort

Quick Sort

Merge Sort

ID Ticket | Placa | Tipo Vehículo | Fecha/Hora Ingreso | Plaza

T-1005	OFA-223	MOTO	2025-12-04 21:22:...	P1-A1
T-1004	PQR-229	AUTO	2025-12-04 21:02:...	P1-A1

Estadísticas **Deshacer** **Cola Espera**

Fecha y hora: 04/12/2025 21:32:09

PlazaPlus

Panel Tickets

Algoritmo de Búsqueda: Búsqueda en Árbol

Criterio de Búsqueda: Por ID Ticket

Buscar

Ordenamiento Completado

Registros ordenados usando Bubble Sort
Criterio: Por Placa
Tiempo: 0.010 ms
Registros: 2

OK

Criterio de Ordenamiento: Por Placa

Algoritmo de Ordenamiento: Bubble Sort

Ordenar

ID Ticket | Placa | Tipo Vehículo | Fecha/Hora Ingreso | Plaza

T-1005	OFA-223	MOTO	2025-12-04 21:22:...	P1-A1
T-1004	PQR-229	AUTO	2025-12-04 21:02:...	P1-A1

Estadísticas **Deshacer** **Cola Espera**

Estado: Conectado **Fecha y hora:** 04/12/2025 21:32:47

Salir

Tickets - Eliminar

PlazaPlus

Reportes Configuración

Inicio Estadias Usuarios Vehículos

Ingresos Egresos Tickets

Panel Tickets

Registrar Ticket Buscar Eliminar Modificar

Seleccione un registro en la tabla y haga clic en Eliminar

Eliminar Ticket Seleccionado

Confirmar eliminación

¿Está seguro que desea eliminar el ticket?

ID: T-1005
Placa: OFA-223

Yes No

Criterio de Ordenamiento: Por Placa

Algoritmo de Ordenamiento: Bubble Sort

Ordenar

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza
T-1005	OFA-223	MOTO	2025-12-04 21:22:00	P1-A1
T-1004	PQR-229	AUTO	2025-12-04 21:02:00	P1-A1

Estado: Conectado Fecha y hora: 04/12/2025 21:42:29

Salir

PlazaPlus

Reportes Configuración

Inicio Estadias Usuarios Vehículos

Ingresos Egresos Tickets

Panel Tickets

Registrar Ticket Buscar Eliminar Modificar

Seleccione un registro en la tabla y haga clic en Eliminar

Eliminar Ticket Seleccionado

Éxito

Ticket eliminado exitosamente

OK

Criterio de Ordenamiento: Por Placa

Algoritmo de Ordenamiento: Bubble Sort

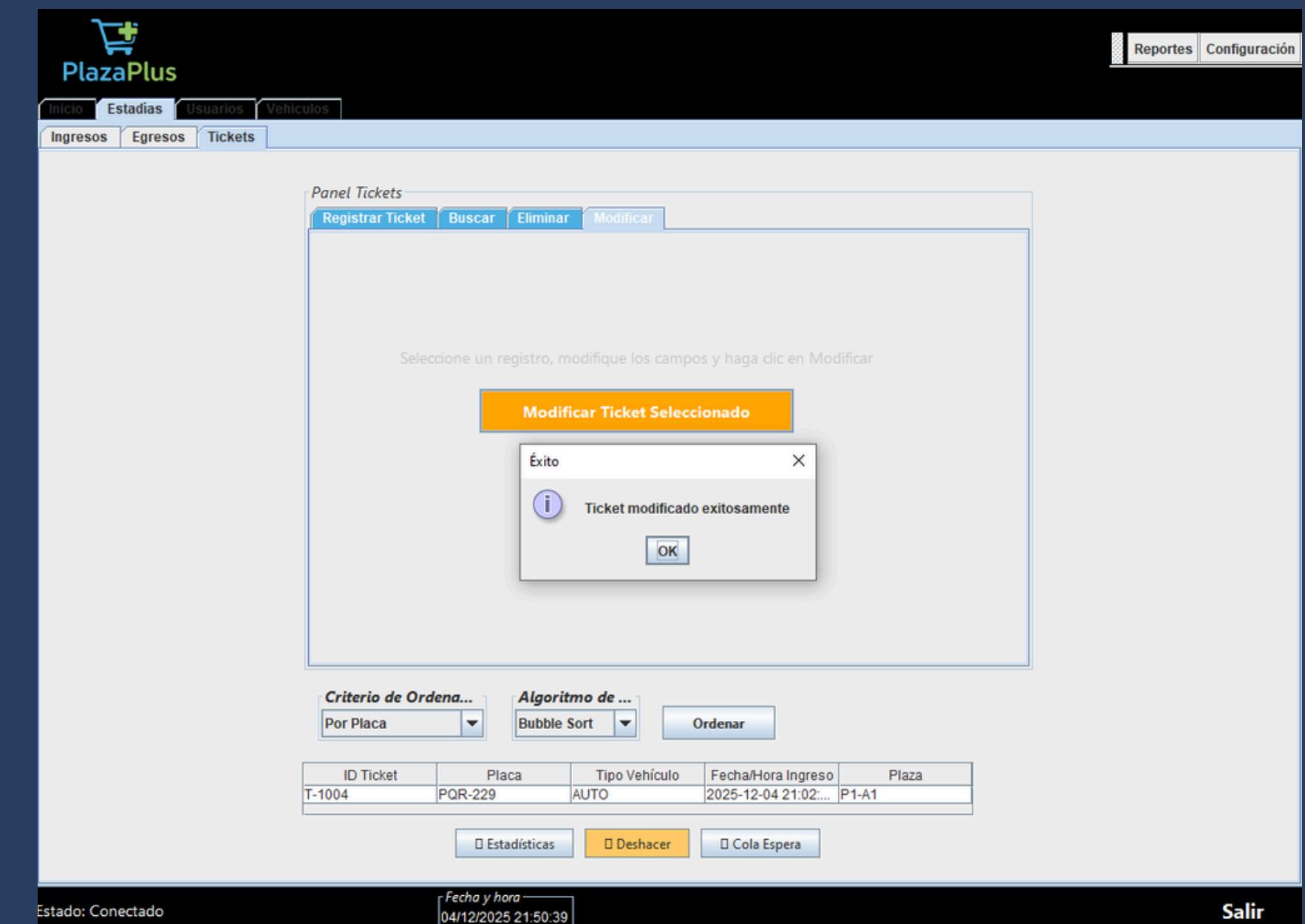
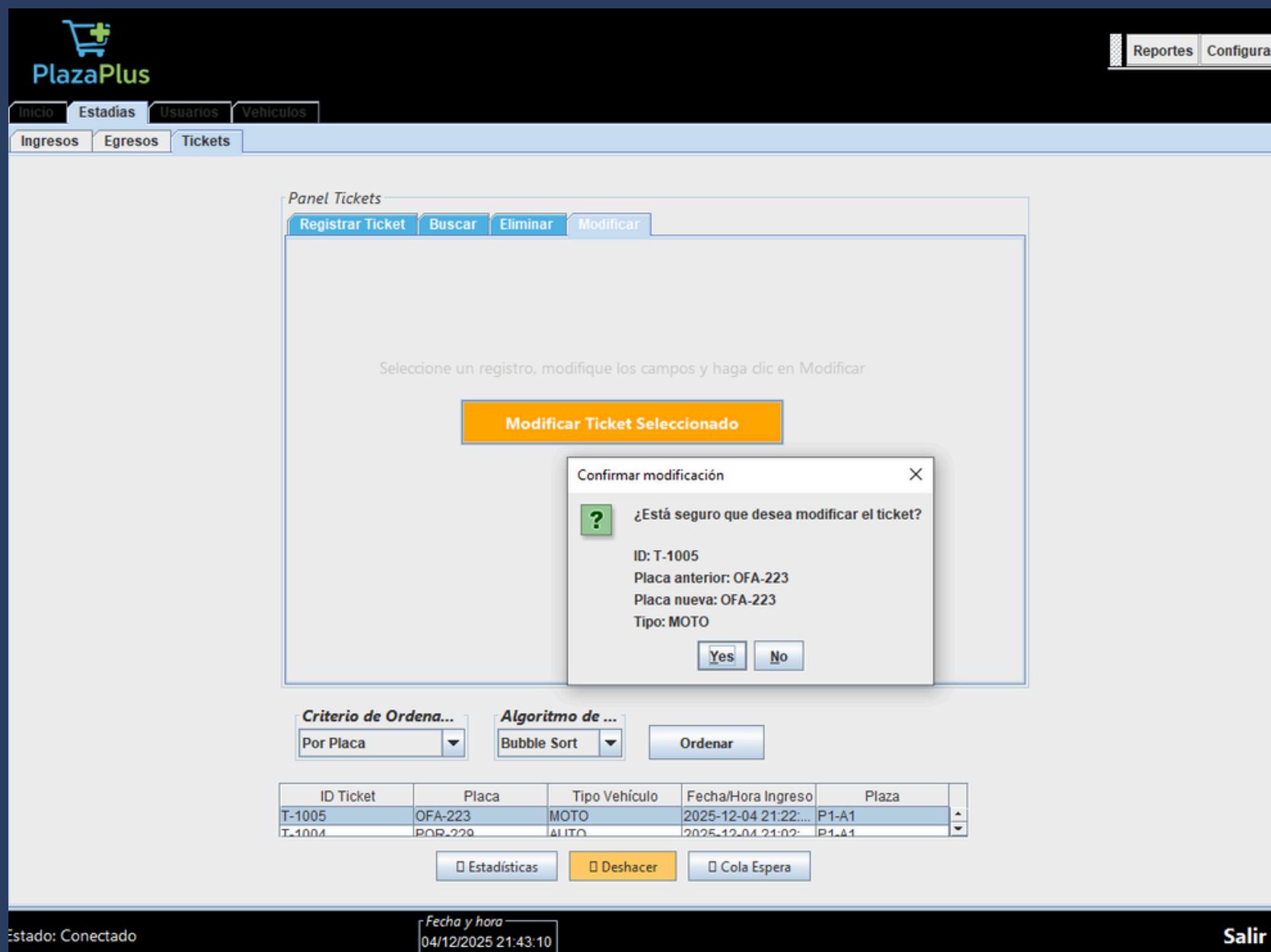
Ordenar

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza
T-1004	PQR-229	AUTO	2025-12-04 21:02:00	P1-A1

Estado: Conectado Fecha y hora: 04/12/2025 21:43:51

Salir

Tickets - Modificar



Estadísticas de Estructuras

Estado de Estructuras de Datos:

- ListaEnlazada (Tickets Activos): 2 elementos
- ÁrbolBinario (Índice por Placa): 2 elementos
- Pila (Historial): 2 operaciones
- Cola (Lista de Espera): 0 solicitudes

Criterio de Ordena... **Algoritmo de ...** **Ordenar**

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza
T-1005	OFA-223	MOTO	2025-12-04 21:22:...	P1-A1
T-1004	POR-229	AUTO	2025-12-04 21:02...	P1-A1

PlazaPlus

Panel Tickets

Registrar Ticket **Buscar** **Eliminar** **Modificar**

Algoritmo de Búsqueda... **Criterio de Busqueda**

Búsqueda en Árbol **Por ID Ticket** **Buscar**

Confirmar deshacer

¿Está seguro que desea deshacer la última operación?
Esta acción liberará la plaza y eliminará el ticket de la lista.

Criterio de Ordena... **Algoritmo de ...** **Ordenar**

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza
T-1005	OFA-223	MOTO	2025-12-04 21:22:...	P1-A1
T-1004	POR-229	AUTO	2025-12-04 21:02...	P1-A1

Estado: Conectado

Fecha y hora: 04/12/2025 21:40:36

Salir

Estadísticas

Deshacer

PlazaPlus

Panel Tickets

Registrar Ticket **Buscar** **Eliminar** **Modificar**

Algoritmo de Búsqueda... **Criterio de Busqueda**

Búsqueda en Árbol **Por ID Ticket** **Buscar**

Cola vacía

No hay vehículos en lista de espera

Criterio de Ordena... **Algoritmo de ...** **Ordenar**

ID Ticket	Placa	Tipo Vehículo	Fecha/Hora Ingreso	Plaza
T-1004	POR-229	AUTO	2025-12-04 21:02...	P1-A1

Estado: Conectado

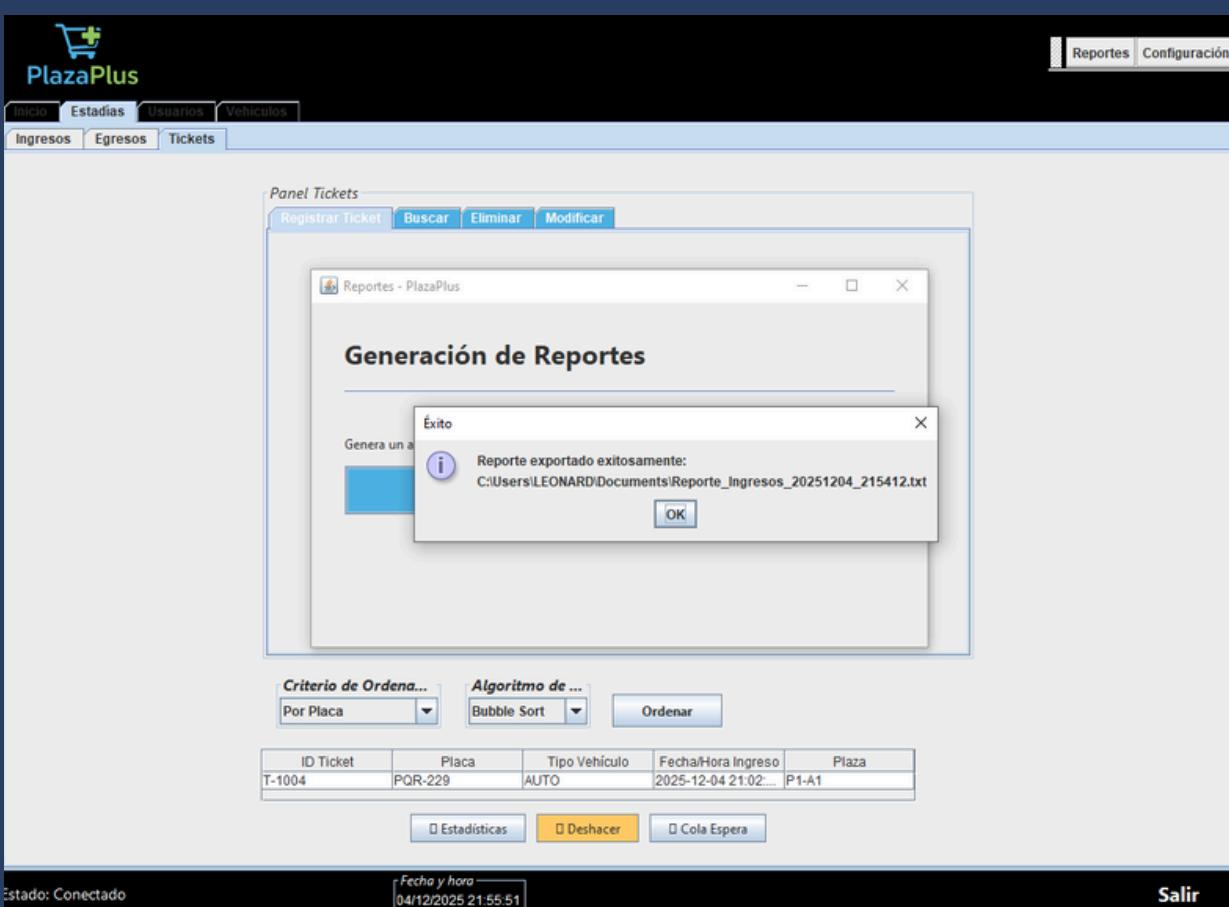
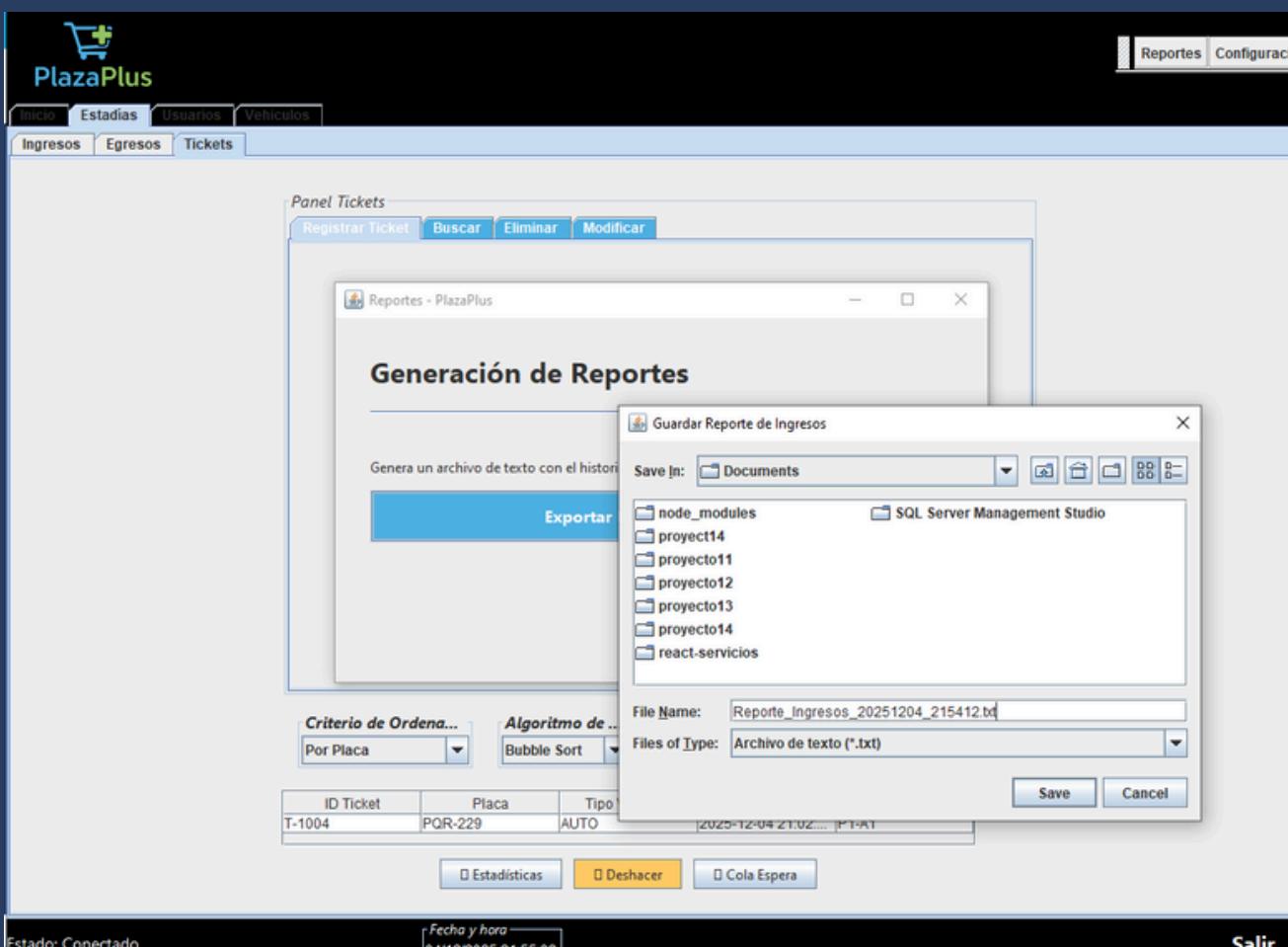
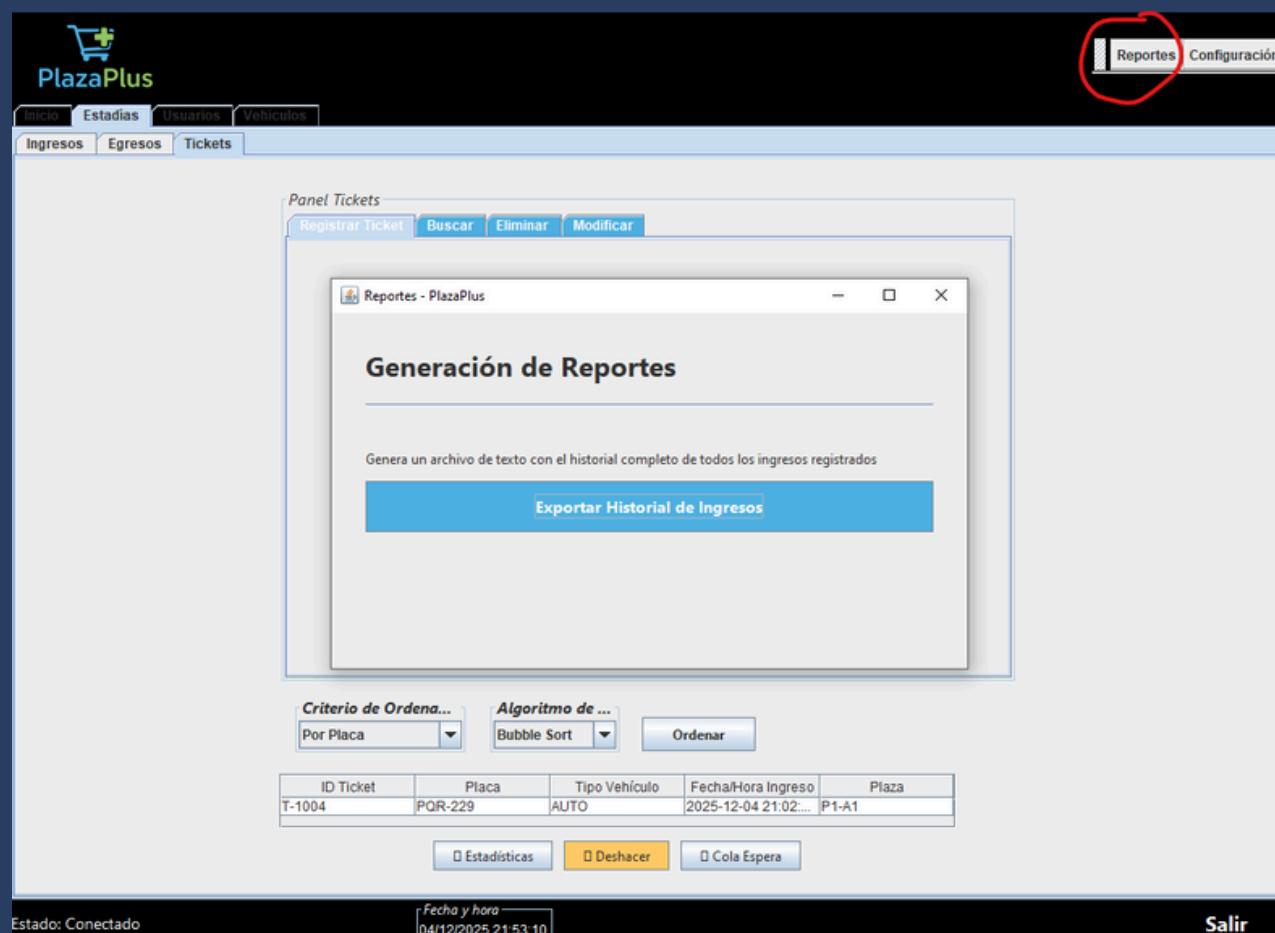
Fecha y hora: 04/12/2025 21:47:18

Salir

Cola Espera

REPORTES

Boton - Reportes



	Nombre	Fecha de modificación	Tipo	Tamaño
Universidad Tecnic	node_modules	13/11/2025 22:08	Carpeta de archivos	
nts	proyect14	30/11/2025 18:37	Carpeta de archivos	
tos	proyecto11	24/10/2025 15:43	Carpeta de archivos	
IVU_ObedVelarde	proyecto12	12/11/2025 20:38	Carpeta de archivos	
tos	proyecto13	13/11/2025 00:12	Carpeta de archivos	
	proyecto14	13/11/2025 19:27	Carpeta de archivos	
	react-servicios	13/11/2025 22:07	Carpeta de archivos	
	SQL Server Management Studio	17/3/2025 16:15	Carpeta de archivos	
	package.json	13/11/2025 22:08	Archivo de origen ...	1 KB
	package-lock.json	13/11/2025 22:08	Archivo de origen ...	11 KB
	Reporte_Ingresos_20251204_215412.txt	4/12/2025 21:55	Documento de te...	2 KB

PlazaPlus

Reporte_Ingresos_20251204_215412.txt: Bloc de notas

REPORTE DE HISTORIAL DE INGRESOS - PLAZAPLUS

Fecha de generación: 04/12/2025 21:55:18

Total de registros: 6

ID TICKET	PLACA	TIPO VEHÍCULO	FECHA/HORA INGRESO	PLAZA	ESTADO
T-1000	LOE-654	AUTO	04/12/2025 01:15:29	P1-B1	FINALIZADO
T-1001	XLA-123	SUV	04/12/2025 01:41:10	P1-D1	FINALIZADO
T-1002	LOL-777	AUTO	04/12/2025 16:47:50	P1-A1	FINALIZADO
T-1003	MAN-111	MINIVAN	04/12/2025 17:28:22	P1-A1	FINALIZADO
T-1004	PQR-229	AUTO	04/12/2025 21:02:15	P1-A1	FINALIZADO
T-1005	OFA-223	MOTO	04/12/2025 21:22:56	P1-A1	ACTIVO

RESUMEN:

- Tickets Activos: 1
- Tickets Finalizados: 5
- Total: 6

Estado: Conectado [04/12/2025 22:00:08] Salir

Conclusiones

1. El sistema desarrollado permitió optimizar la gestión interna del estacionamiento, ya que automatiza procesos como el registro de vehículos, control de ingresos y salidas, y la asignación de plazas.
2. Se logró reducir significativamente los errores humanos en la asignación de plazas, gracias al uso de estructuras de datos dinámicas y algoritmos de búsqueda que controlan la disponibilidad en tiempo real.
3. La implementación de un módulo de reportes permitió disponer de información estructurada y verificable sobre las operaciones del estacionamiento, facilitando el análisis y la supervisión administrativa, incluso sin depender de una base de datos completa.
4. La implementación bajo el patrón MVC permitió obtener un sistema modular, escalable y fácil de mantener, lo cual asegura que la solución pueda integrarse o ampliarse en el futuro, por ejemplo, incorporando conexiones a bases de datos o módulos avanzados de control.

¡Gracias!