MANUAL DE CÓDIGO - PARKPLUS

1. INTRODUCCIÓN

Este manual describe la estructura interna, lógica, organización y funcionamiento del código del sistema ParkPlus, desarrollado en Java con Swing y MySQL. Su propósito es servir como referencia técnica para futuros mantenimientos, ampliaciones o auditorías.

2. ARQUITECTURA DEL SISTEMA

El proyecto sigue una arquitectura por capas:

- Capa de presentación (JFrames)
- Capa de lógica/subclases (SubClases)
- Capa de acceso a datos (dbConnection)
- Capa de base de datos (MySQL)

Cada módulo está diseñado para una responsabilidad específica.

3. CONEXIÓN A BASE DE DATOS

La clase dbConnection administra la conexión a MySQL mediante JDBC.
Método principal:
- conectar(): retorna Connection.
- Manejo de excepciones SQLException.

4. SUBCLASES PRINCIPALES

4.1 SubTicket

Responsable de:
- Crear tickets.
- Obtener ticket activo por placa.
- Registrar salida temporal.
- Calcular montos.
- Actualizar estado del ticket.

Métodos clave:
- crearTicket(...)
- obtenerTicketIdActivo(...)
- registrarSalidaTemporal(...)
- actualizarMonto(...)

4.2 SubParqueo

Gestiona los SPOTS dentro del parqueo:
- obtenerSpotPorTicket(ticket_id)
- marcarTemporal(spot)
- marcarOcupado(spot)
- liberarSpot(spot)

Estados de un spot:
• FREE → verde
• OCCUPIED → rojo
• TEMPORAL → amarillo

4.3 SubPersona

Registra y administra usuarios:
• CrearPersona(...)
• ValidarExistencia(...)
• ActualizarSaldo(...)
Atributos: placa, carnet, nombres, apellidos, teléfono, saldo, tipoPersona.

4.4 SubReingreso

Lógica para validar reingresos:
• validarReingreso(placa)
• obtenerTicketActivo(placa)
• permitirReingreso(id)

Reglas:
• FLAT → reingreso permitido.
• VARIABLE → solo si no han pasado más de 2 horas desde la salida temporal.

4.5 SubReportes

Genera reportes diarios:
• obtenerTicketsDelDia()
• obtenerTotalDelDia()

Estructura interna:
TicketReporte { id, placa, modo, ingreso, salida, monto }

5. INTERFACES GRÁFICAS (FRAMES)

5.1 PagoV

Función: Selección del método de pago y registro de ticket.
Recibe: placa, total, modo.
Acciones:
• Crear ticket.
• Mostrar confirmación.
• Regresar al menú.

5.2 RegistrarP

Registra personas:
• Evento para mayúsculas en placa.
• Cambios automáticos si es VISITANTE.
Valida saldo solo para estudiantes y catedráticos.

5.3 Reingreso

Valida:
• Existencia de ticket.
• Salida temporal.
• Tiempo permitido.
• Tipo de ticket.

Mensajes según estado:
• NO_TICKET
• NO_SALIDA_TEMP
• CADUCADO
• OK

5.4 Reportes

Muestra en JTable todos los tickets del día.
Calcula total en Q.
Interfaz simple con tabla y campo TOTAL.

5.5 Salidas

Permite salida total o temporal.

Salida total:
• Calcula monto variable si corresponde.
• Cierra ticket.
• Libera spot → FREE.

Salida temporal:
• Marca spot como TEMPORAL.
• Registra timestamp de salida temporal.

5.6 VParqueo

Genera el mapa dinámico del parqueo por áreas.

Colores:
• Verde → libre
• Rojo → ocupado
• Amarillo → salida temporal

Panel organizado por áreas y grid por spots.

6. BASE DE DATOS

Tablas principales:

TABLE area
• area_id
• nombre

• tipo_vehiculo

TABLE spots
• spot_id
• area_id
• tipo_vehiculo
• status

TABLE persona
• placa
• carnet
• nombres
• apellidos
• telefono
• saldo
• tipo

TABLE ticket
• ticket_id
• placa
• modo
• fecha_ingreso
• fecha_salida
• fecha_salida_temp
• spot_id
• monto

7. FLUJOS PRINCIPALES DEL SISTEMA

7.1 Ingreso

1. Ingreso placa y modo.
2. Se asigna spot libre compatible con el tipo de vehículo.
3. Se marca como OCCUPIED.
4. Se crea ticket.

7.2 Salida Total

1. Buscar ticket activo.
2. Calcular monto.
3. Actualizar ticket.
4. Liberar spot → FREE.
5. Mostrar total.

7.3 Salida Temporal

1. Validar ticket activo.
2. Registrar fecha_salida_temp.
3. Marcar spot → TEMPORAL.

7.4 Reingreso

---
1. Validar ticket.
2. Verificar tiempo de salida temporal.
3. Cambiar spot → OCCUPIED.

8. ESTRUCTURA DE PAQUETES

• ClaseVentanas (interfaces)
• SubClases (lógica interna)
• claseprin (main, conexión)
• recursos (PDF, CSV)

9. POSIBLES MEJORAS FUTURAS

• Implementar roles de usuario.
• Historial automático en CSV.
• Notificaciones de expiración.
• Optimización de búsqueda por spot.

10. CONCLUSIÓN

El código está organizado modularmente y permite un mantenimiento claro. La separación entre interfaz, lógica y datos garantiza escalabilidad y facilidad de implementación para futuras funciones.