# EASYPARK CONTROL: GESTIÓN DE PARQUEADERO

## RINCONCITO RUEDAS

Mariana Alvarez Betancur
Jhon Andres Taimal Fuelantala
Isabela Velez Buritica

Universidad de Antioquia
Facultad de Ingeniería

Algoritmia y Programación

Catalina Maria Maya Iregui

Lunes, 24 de Marzo de 2025

## A. REPORTE DE VISIÓN

### Objetivos

-Reducir el tiempo de espera en procesos de ingreso , ubicación y pago para mejorar la experiencia del cliente.

-Garantizar la operación eficiente mediante la supervisión del personal, el control de ingresos y egresos.

### Beneficios

El software a realizar busca suplir las necesidades presentadas en la problemática, sus principales beneficiarios son:

1. La creación de este software elimina los errores que pueden surgir al utilizar sistemas manuales y en papel.
2. Este software ayudará y facilitará el control de entrada y salida de los vehículos.
3. Al usar un software que realiza los cálculos de manera automática acelera el proceso de cobro.
4. Habrá una atención más rápida.
5. Habrá una visualización en tiempo real de las celdas disponibles y esto optimizará el uso del espacio
6. Se podrá evitar el sobrecupo del parqueadero o el desperdicio de espacio.

## B. ESPECIFICACIÓN DE REQUISITOS

El software presenta un sistema de requisitos funcionales como permitir la gestión a usuarios como:

- Registrar usuarios (nombre, apellidos, documento de identidad, placa del vehículo y teléfono)
- Registrar solamente a automóviles.
- Evitar el registro de motocicletas.
- Control al ingreso y salida de vehículos.

## C. LIBRERÍAS

1. **DATETIME:** Es una librería que permite trabajar con fechas y horas

**Funciones:**

- datetime.datetime.now(): Devuelve la fecha y hora actual.
- datetime.datetime.today(): Igual que now(), también la fecha y hora actual.
- hora_salida - hora_estadar: Resta dos fechas y te da la diferencia de tiempo (un timedelta)
- .strftime(formato): Convierte fecha/hora en texto con el formato a elección.

%d: día
%m: mes
%Y: año
%H: hora (24h)

%M: minutos
%S: segundos

2. **RE:** Esta librería es una forma de buscar patrones dentro de un texto.

- re.match(patrón, texto): ¿El texto empieza cumpliendo el patrón?
- re.fullmatch(patrón, texto): ¿Todo el texto cumple el patrón exactamente?
- re.search( patrón, texto): ¿En algún lugar del texto aparece el patrón?
- re.findall(patrón, texto): Encuentra todas las coincidencias.

**Tabla 1. Notas importantes para la librería "re"**

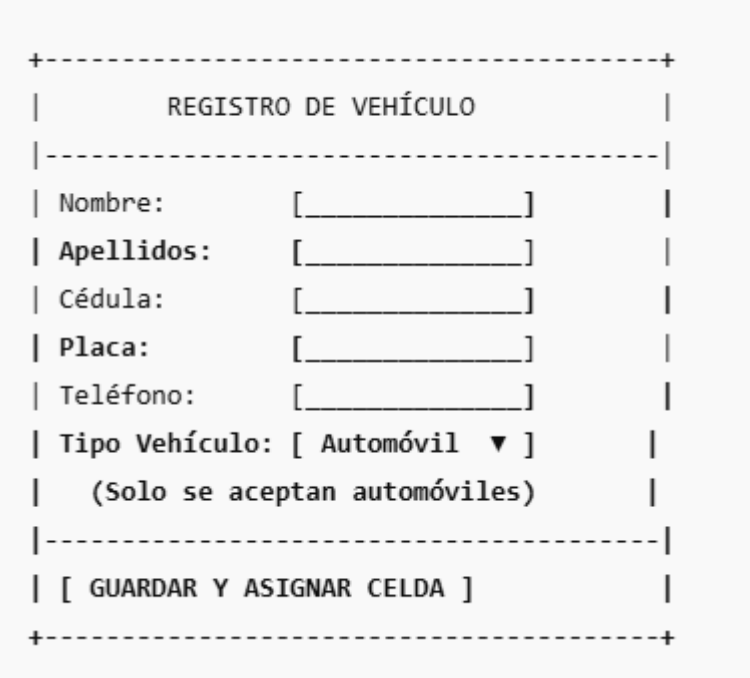
| SÍMBOLO | SIGNIFICADO |
|---|---|
| \d | Un número (0–9) |
| \D | Un carácter que NO es número |
| [A-Z] | Una letra mayúscula |
| [a-z] | Una letra minúscula |
| {n} | Exactamente n repeticiones |
| {n,m} | Entre n y m repeticiones |
| . | Cualquier carácter |
| ^ | Comienzo del texto |
| $ | Final del texto |

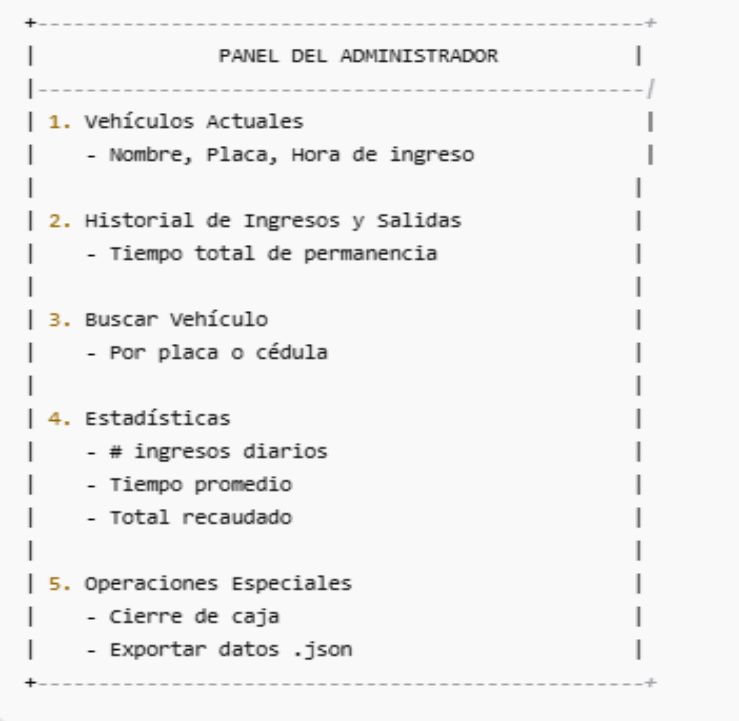
3. **.JSON:** Es un formato de intercambio de datos muy usado (parecido a un diccionario de Python pero en texto)

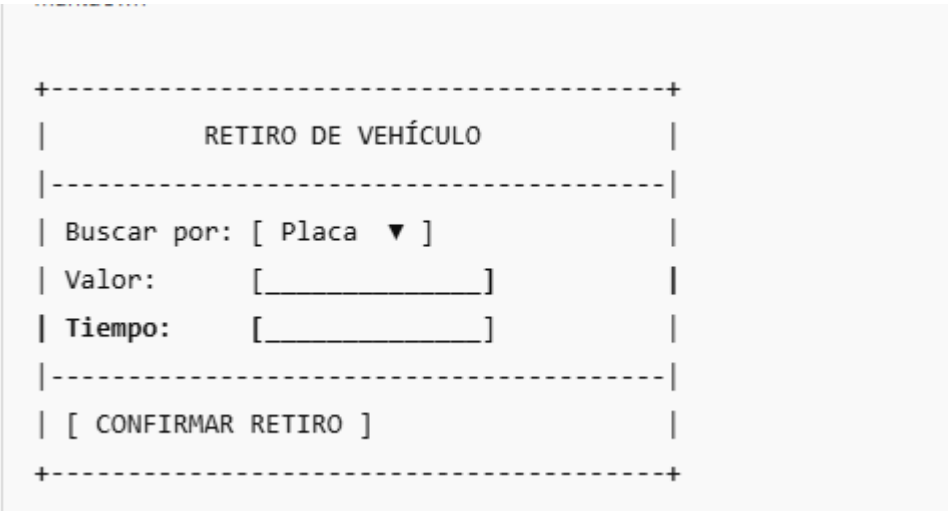
**Funciones:**

- Convertir un objeto de Python (como un dict) en una cadena JSON → (json.dumps()).
- Convertir una cadena JSON en un objeto Python → (json.loads()).
- Leer datos desde un archivo .json → (json.load()).
- Guardar datos en un archivo .json → (json.dump()).

## D. VISUAL

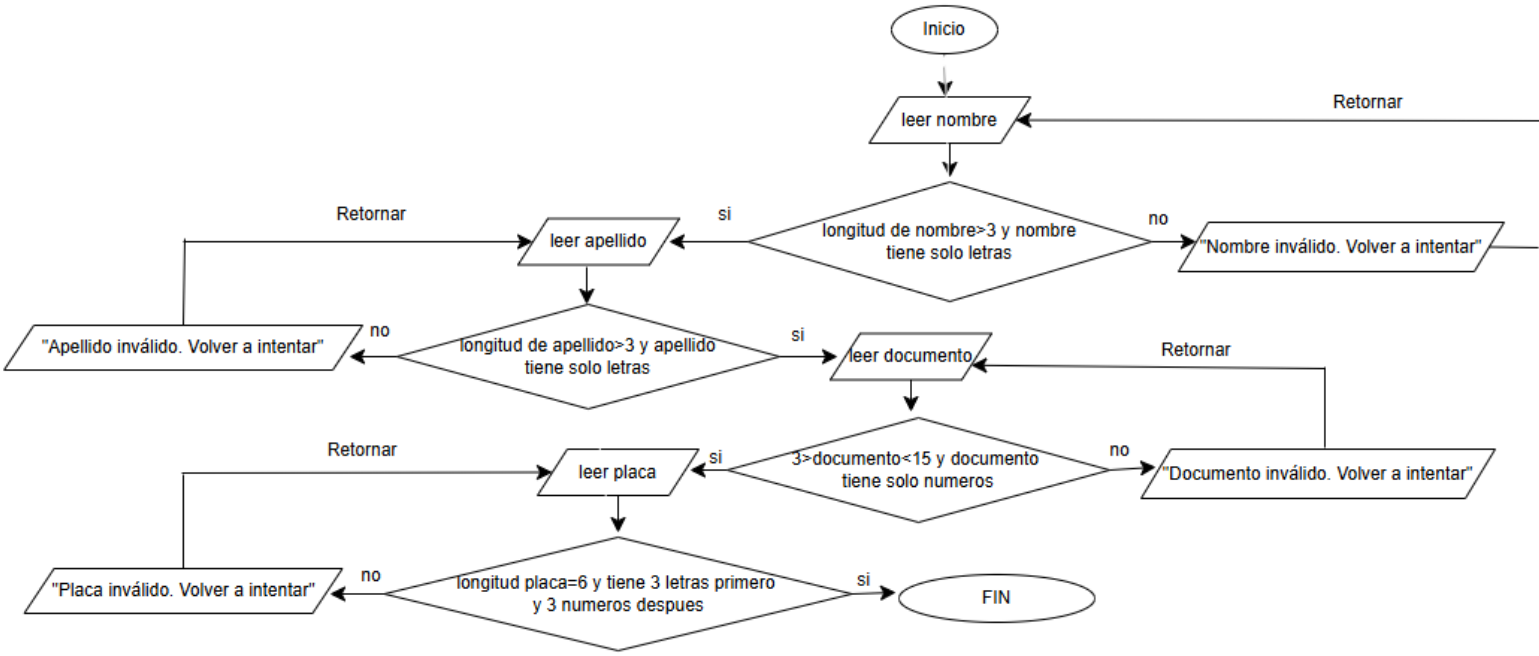

REGISTRO DE VEHÍCULO
Nombre: [_____]
Apellidos: [_____]
Cédula: [_____]
Placa: [_____]
Teléfono: [_____]
Tipo Vehículo: [ Automóvil ▼ ]
(Solo se aceptan automóviles)
[ GUARDAR Y ASIGNAR CELDA ]


PANEL DEL ADMINISTRADOR
1. Vehículos Actuales
- Nombre, Placa, Hora de ingreso
2. Historial de Ingresos y Salidas
- Tiempo total de permanencia
3. Buscar Vehículo
- Por placa o cédula
4. Estadísticas
- # ingresos diarios
- Tiempo promedio
- Total recaudado
5. Operaciones Especiales
- Cierre de caja
- Exportar datos .json

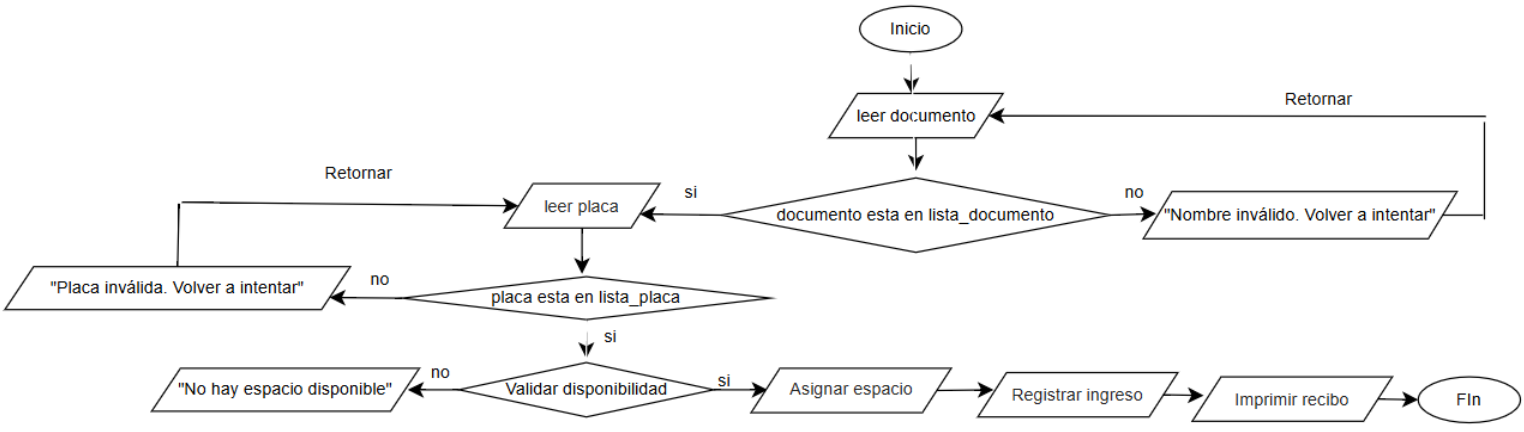

RETIRO DE VEHÍCULO
Buscar por: [ Placa ▼ ]
Valor: [_____]
Tiempo: [_____]
[ CONFIRMAR RETIRO ]

## E. ALGORITMOS

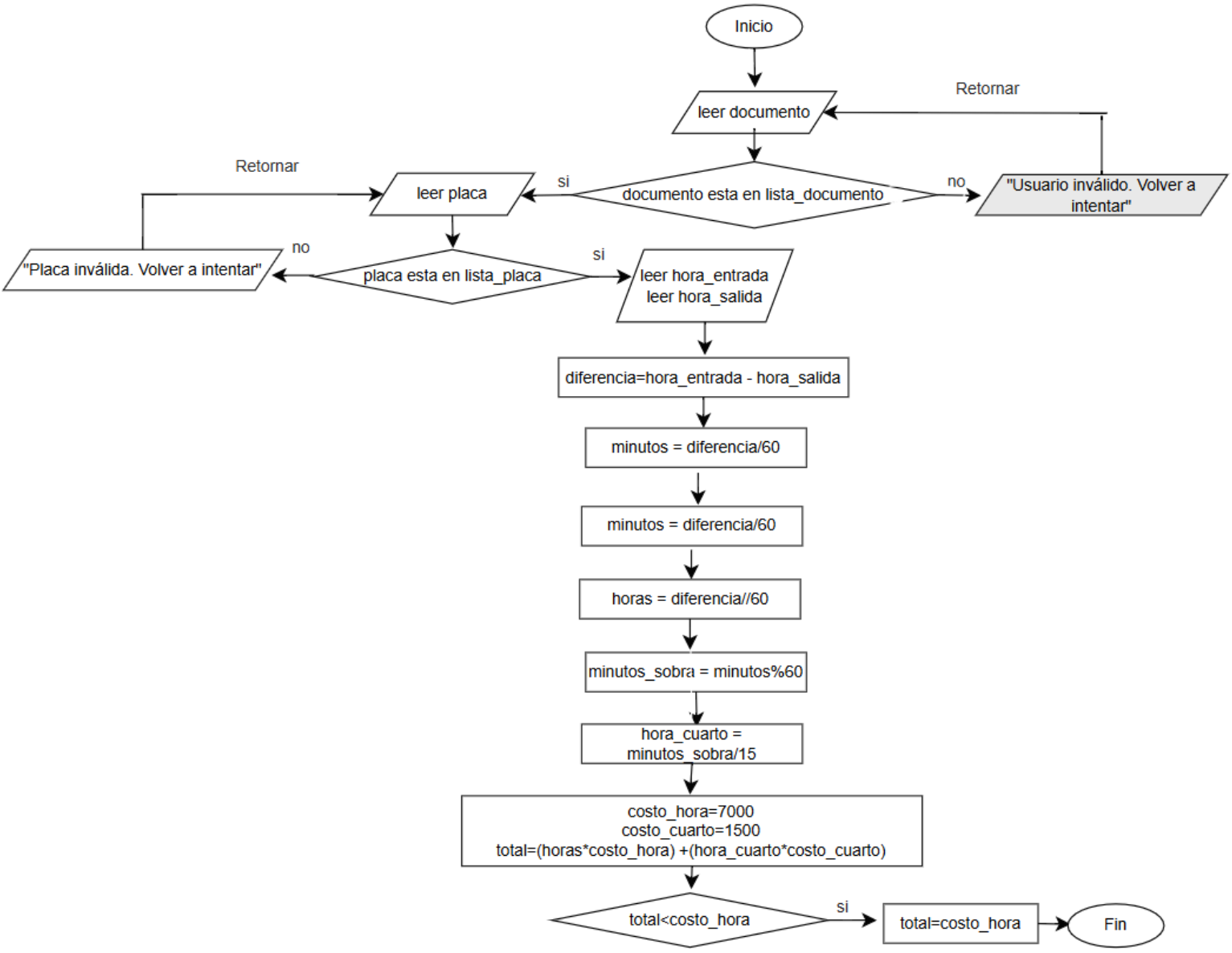
- **Registrar Usuario:**



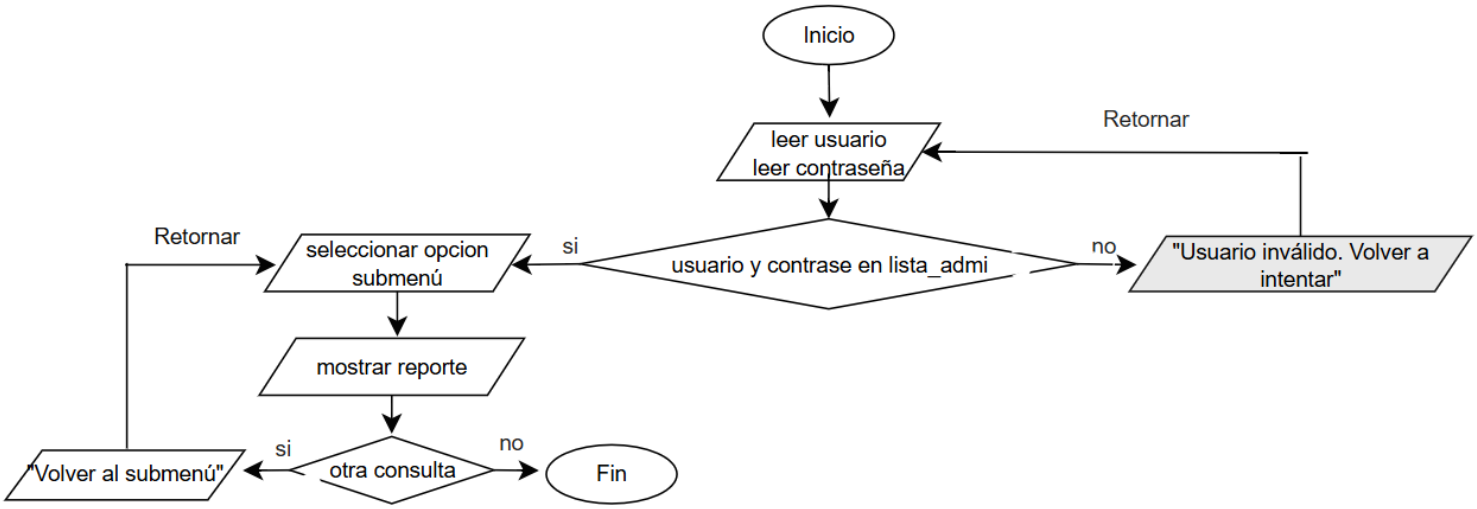

- **Ingresar Vehículo:**



- **Retirar Vehículo:**



- **Ingresar al administrador:**




## F. ESTRUCTURAS DE DATOS.

El software para la gestión de parqueadero se propone las siguientes estructuras de datos:

**1.Diccionarios:** Estos nos permiten acceso rápido mediante identificadores únicos, por ejemplo, diccionarios de documentos, placas que facilitan la organización de datos relacionados.

· Nos permite almacenar la información de los usuarios

· Nos permite el registro de los vehículos estacionados

**2. Listas:** Permiten almacenar varios diccionarios como usuarios o vehículos de forma secuencial, logrando fácilmente todos los registros que nos permiten mirar los registros de reportes o hacer búsquedas.

**3.Tuplas:** Se puede utilizarse para valores constantes o registros que no se pueden modificar.

· Permitir el tipo de vehículos aceptados

· Coordenadas de ubicación

**4.Archivos:** Para esto vamos a utilizar JSON ya que es el más ideal para el almacenamiento y fácil intercambio de estructuras.

Datos que almacenar:

· Información de usuarios

· Registro de vehículos

· Historial de todos los movimientos

## G. PRESENTACIÓN DE RESULTADOS

El diseño preliminar del módulo de administrador:

· **Ver lista de vehículos actuales en el parqueadero**

- Indicando el nombre, placa, hora de ingreso, tipo transcurrido

· **El historial de ingresos y salidas**

- Una tabla con los datos de ingreso y retiro, incluyendo tiempo total de permanencia

- **Buscar vehículo por placa o documento**
- Mostrar todos los datos detallados del usuario y registros históricos.
- **Estadísticas**
- Número de ingresos diarios
- Tiempo de permanencia
- Ingresos
- **Operaciones especiales**
- Cierre de caja
- Exportar datos.