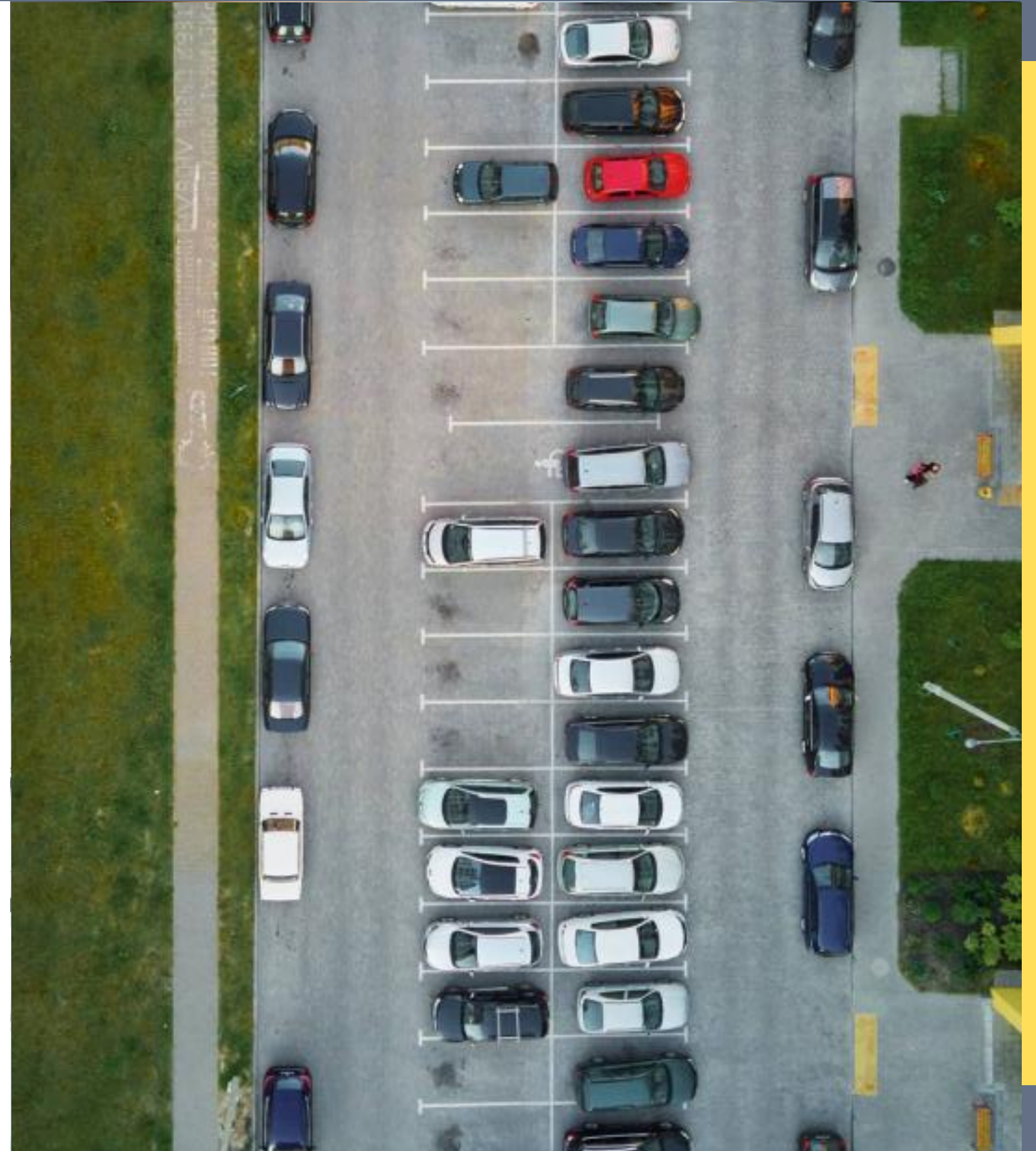


# SISTEMA DE CONTROL DE PARQUEA DERO

Parking Center S.A.S





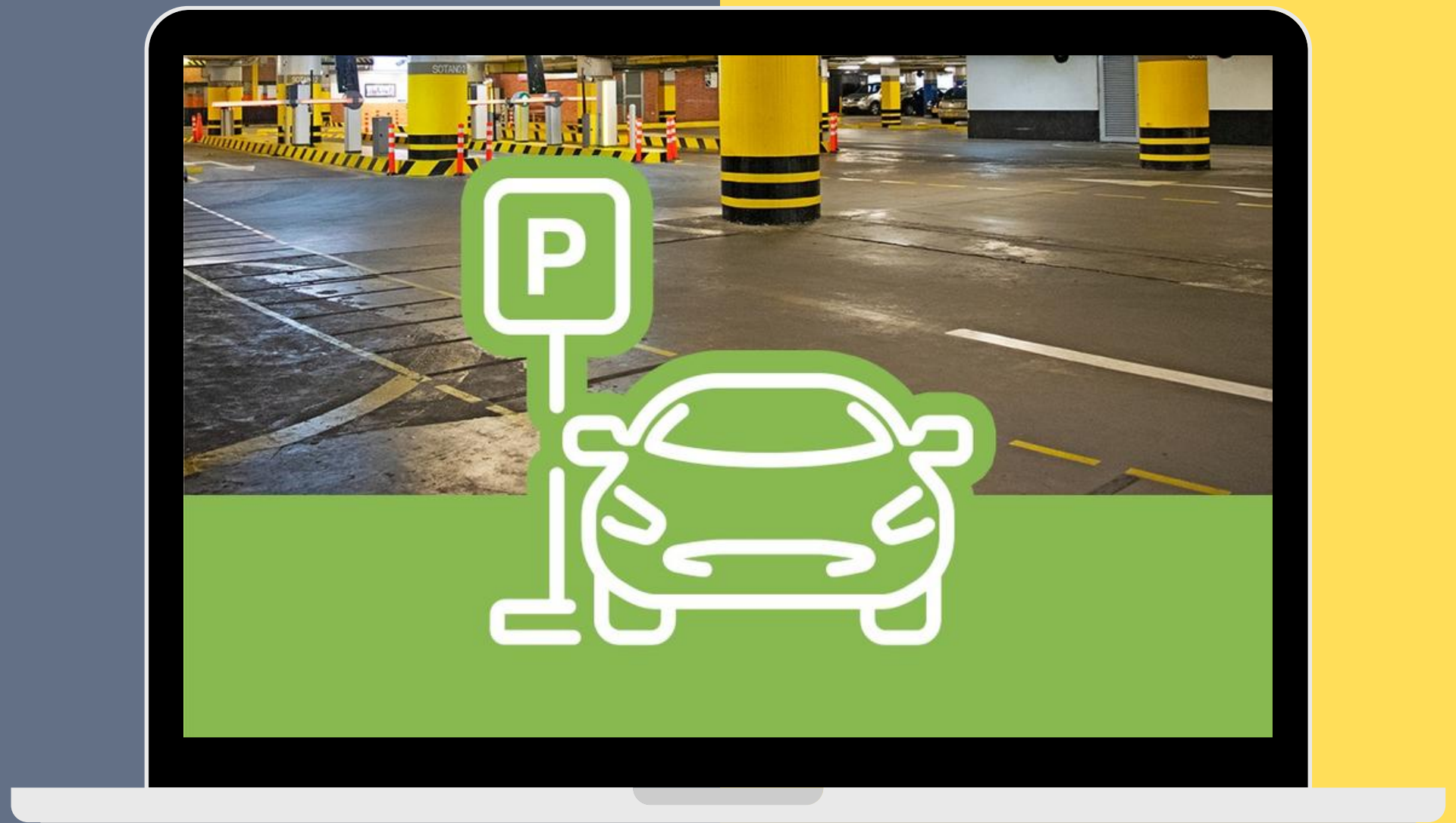
# Índice

- 1. Introducción**
- 2. Objetivos**
- 3. Requerimientos**
- 4. Primeros Avances y Mockup**
- 5. Estructura del Código**
- 6. Entradas y Salidas**
- 7. Conclusiones**



# Introducción

En busca del desarrollo de un prototipo para un sistema de registro y cobro de parqueadero, la empresa Parking Center S.A.S., encargada de operar varios parqueaderos, desea desarrollar un sistema básico de control para registrar el ingreso y salida de vehículos, y calcular el cobro de acuerdo con el tiempo de permanencia.





# Objetivos

**Desarrollar un prototipo funcional:** Queremos mostrar un sistema de parqueadero que pueda registrar ingresos, salidas y calcular cobros de forma básica.

**Implementar requerimientos técnicos:** Buscamos demostrar cómo a partir de tuplas, se calculan tarifas y horas ficticias para que el sistema funcione.

**Validar la solución propuesta:** Nuestro objetivo es probar que el sistema maneja bien los errores (como placas repetidas) y que es una solución práctica y efectiva para la gestión del parqueadero.

**Reforzar habilidades de programación:** El proyecto nos permitió aplicar y mejorar nuestra programación, uso de datos y lógica en un problema real.



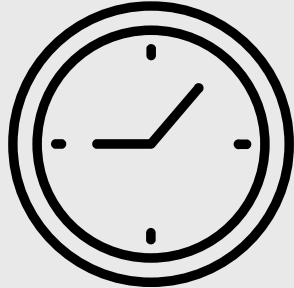






# Requerimientos

**El  
sistema  
permite:**



# Detalles técnicos:

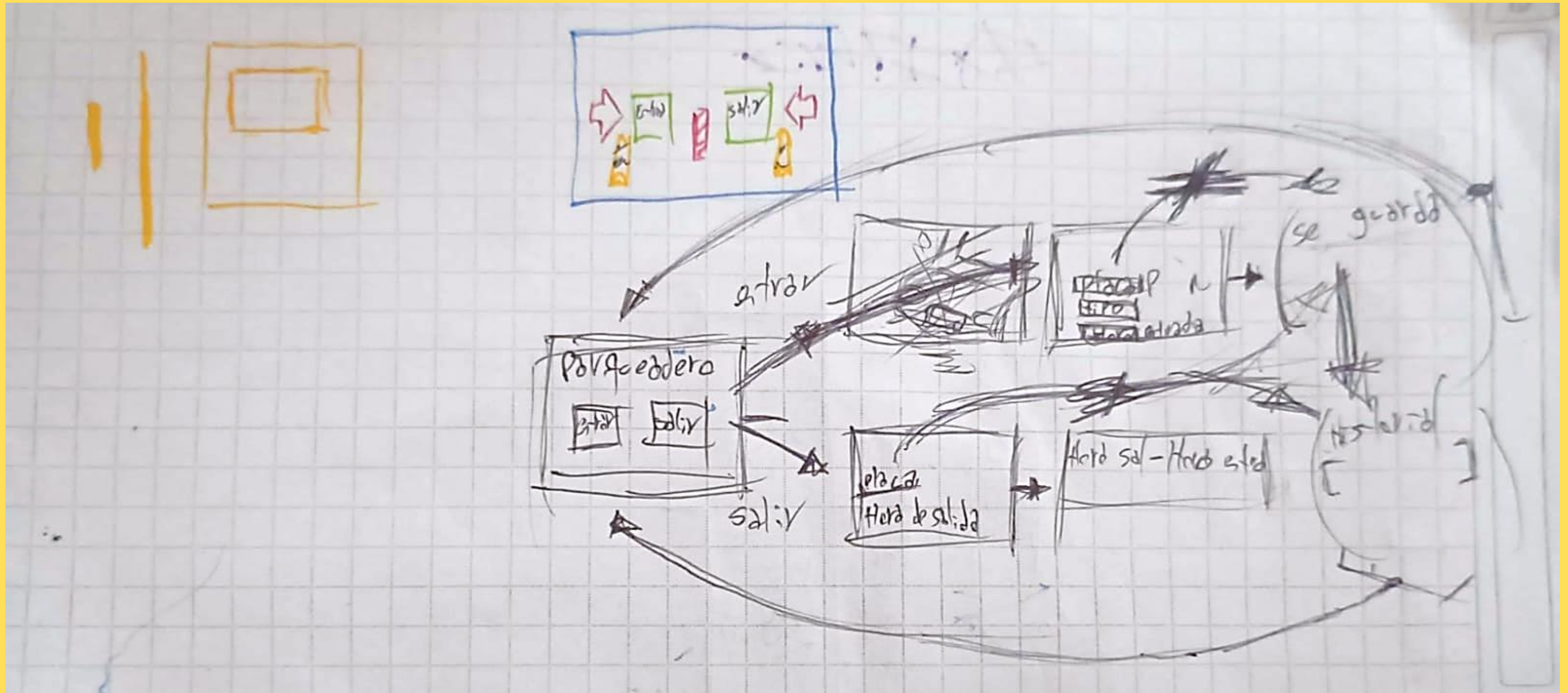
# Requerimientos

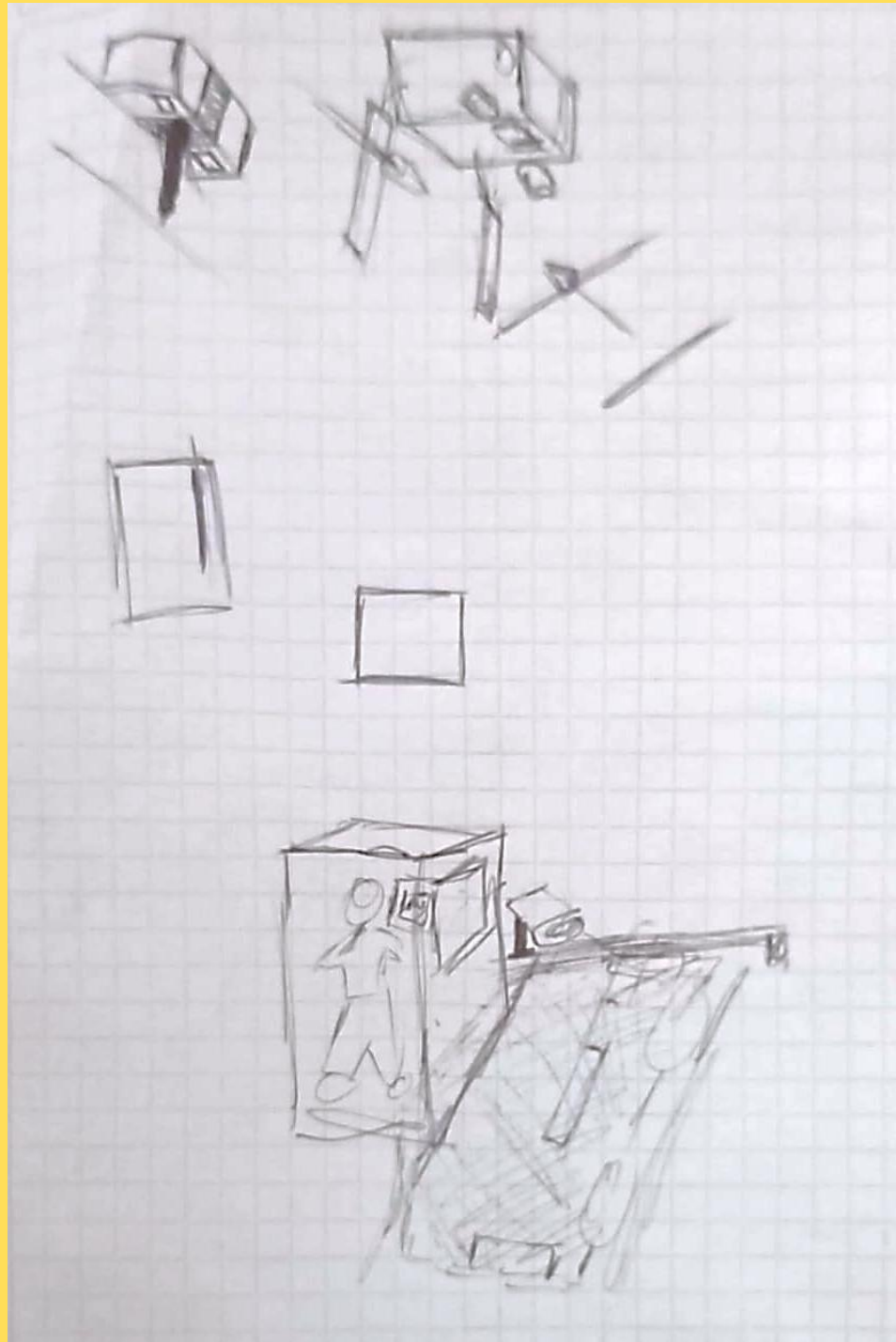
1. Datos de los vehículos	2. Tipos de vehículos y tarifas	3. Menú interactivo con bucles y condicionales	4. Fórmulas matemáticas	5. F-strings
<p>Cada vehículo es representado con una tupla, los cuales si están activos son guardados en una lista y al salir, se calcula el valor y se guarda en el historial</p> <div></div>	<div><div>\$1.000 por hora </div><div>\$2.000 por hora </div><div>\$2.500 por hora </div></div>	<div>Registrar ingreso</div> <div>Registrar salida</div> <div>Consulta parqueadero actual</div> <div>Ver total recaudado</div> <div>Salir</div>	<p>Se calculan las horas de permanencia y multiplicar por tarifa. Además, se simula el ingreso/salida con horas ficticias (números enteros entre 0 y 24)</p>	<div><div><div><div></div><div></div><div></div></div><div></div><div><div>Tipo de vehículo:</div><div>PLACA:</div></div><div>XX-XX01X0010X</div><div><div></div><div>Hora de Entrada:</div><div>HH/MM</div></div><div><div></div><div>Hora de Salida:</div><div>HH/MM</div></div><div><div></div><div>Tarifa por hora:</div><div>\$</div></div><div><div><div></div></div><div>Horas:</div><div>Total a pagar:</div></div></div></div>





# Primeros Avances





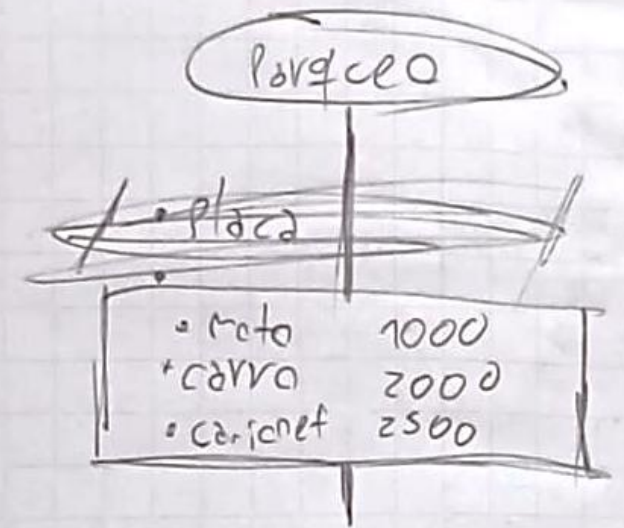
```

1 print("PARQUEADERO")
# while True:
2 entrada = input("Ingresar 'entrada' o 'salida': ")
3 if entrada == "entrar":
4     print("Ingrese los datos:")
5     # placa = input
6     # tipo_Vehiculo =
7     # Hora_entrada =
8     # while True:
9         tipo_Vehiculo = input("Ingrese su tipo de vehiculo: \n moto \n carro \n camioneta \n")
10    # while
11    if len(tipo_vehiculo) > 6 or len(tipo_vehiculo) < 6:
12        print("Placa Invalida")
13        # continue

```

Datos Vehiculo  $\rightarrow$  tuple [(placa), tipo, hora]

≠  
placa





# Mockup

Este sistema te permite llevar el control de los vehículos que entran y salen del parqueadero de manera fácil y rápida. Puedes registrar la entrada y salida de motos, carros y camionetas, calcular automáticamente el tiempo que estuvieron y el valor a pagar, y consultar reportes de ingresos diarios.





# Parking Center

**Registrar ingreso**

**Registrar salida**

**Otras opciones**





**Consulta  
parqueadero actual**

**Ver total diario**

**Salir**



**Parking Center S.A.S.**



**Registrar ingreso**

**Placa:**

AAA111

**Tipo de  
Vehículo:**

(Moto, Carro,  
Camioneta)

**Hora:**

HH/MM

**Fecha:**

DD/MM

**Parking Center S.A.S.**





**Ticket**

		
		
Tipo de vehículo:		PLACA:
XX-XX01X0010X		
	Hora de Entrada: HH/MM	
	Hora de Salida: HH/MM	
		Tarifa por hora:
		\$
	Horas:	Total a pagar: _____ _____

**Parking Center S.A.S.**



**Salida**

**Placa:**

AAA111

**Total a pagar:**

**Hora:**

HH/MM

**Fecha:**

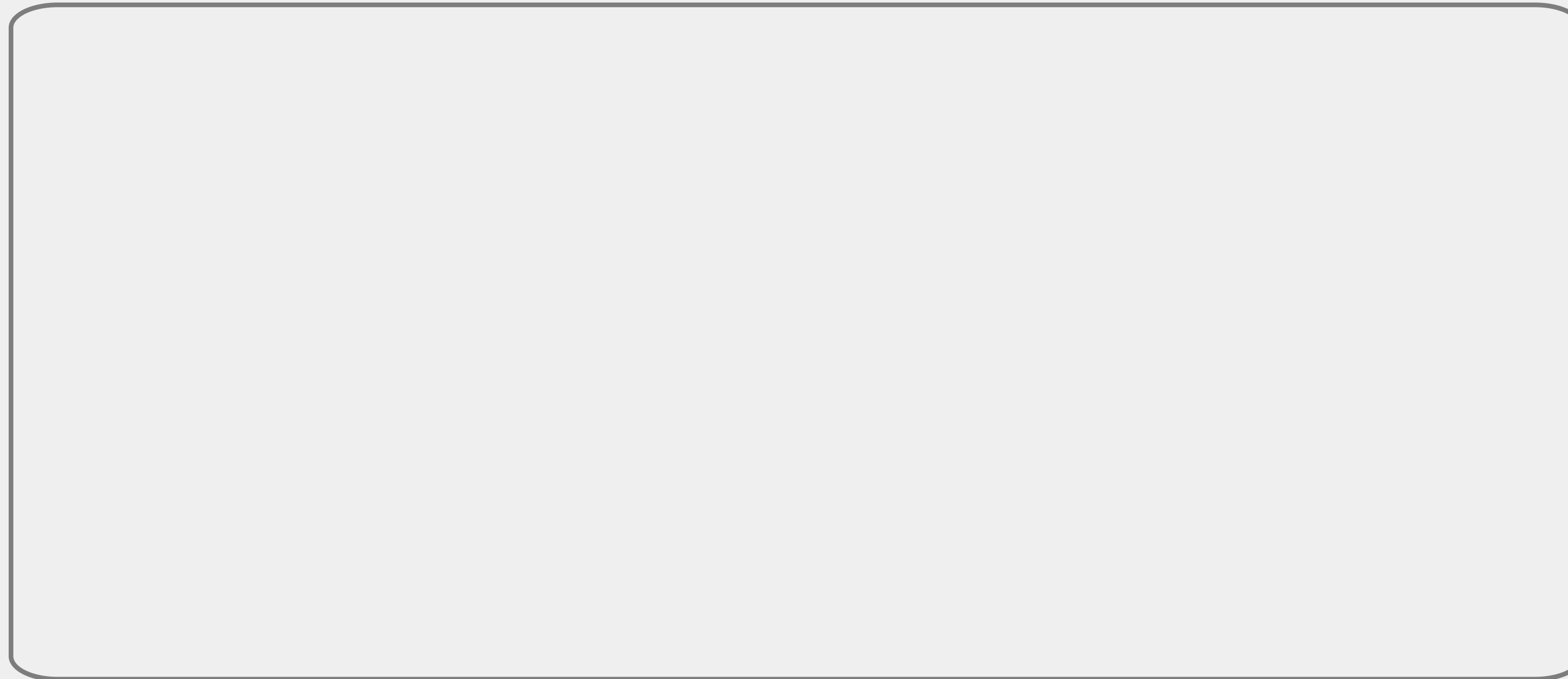
DD/MM

**Parking Center S.A.S.**





## Consulta Parquadero Actual



**Total Vehículos:**

**Parking Center S.A.S.**



**Total diario**

**Ingrese la fecha para  
el reporte de ingresos  
diario:**

DD/MM

**Para el día DD/MM**


**Parking Center S.A.S.**





# Estructura del Código

Este programa es un sistema de control para un parqueadero. Permite registrar los vehículos que entran y salen, calcular cuánto tiempo estuvieron y cuánto deben pagar, y además lleva un historial de todos los movimientos. El usuario puede ver los vehículos que están adentro, consultar los ingresos de un día específico y cerrar el programa cuando quiera. Todo esto se hace desde la consola, donde el usuario va eligiendo opciones y escribiendo los datos que se le piden.

Parqueadero > Sistema de Control de Parqueadero.py

Parqueadero > Sistema de Control de Parqueadero.py > ...

```
1 historial=[] #una lista para el historial de los vehiculos ingresados y sus
2 vehiculos=[] #una lista para los vehiculos ingresados en el parqueadero actu
3 precios={"MOTO":1000,"CARRO":2000,"CAMIONETA":2500} #precios de los vehiculo
4 placas={"01":31,"02":29,"03":31,"04":30,"05":31,"06":30,"07":31,"08":31,"09":
5
6 while True:
7     print("\n\n")
8     print("PARQUEADERO".center(100,"="))
9     print(f"\nEl precio por hora para cada vehiculo es:\n{precios}\n")
10    print("\n Para entrar un nuevo vehiculo al parqueadero ingrese 'E'\n Par
11    entrada=input("\nIngrese la accion que va a realizar: ").upper() #ingres
12
13    if entrada=="E": #si el usuario desea ingresar un vehiculo
14
15
16        #ingresar y verificar la placa
17        comprobador_placa=0 #Variable que ayuda a comprobar si la placa ya e
18        while True:
19            if comprobador_placa==1: #si se detecto la placa repetida
20                break #Si la placa ya está registrada, se rompe el bucle y s
21            placa=input("\nIngrese la placa del vehiculo (AAA000): ").upper()
22            #detectar errores en la placa
23            if len(placa)!=6:#si la longitud de la placa es incorrecta
24                print("Placa no valida vuelva a intentarlo\n")
25                continue #si detecta un error vuelve al inicio del bucle a p
26            elif placa[0:3].isalpha()==False or placa[3:5].isdigit()==False:
27                print("Placa no valida vuelva a intentarlo\n")
28                continue #si detecta un error vuelve al inicio del bucle a p
29            else: #si no hay ningun error al escribir la placa
30                contador_placa=0 #Contador que recorre la lista d
```



## Variables principales

- a. historial: Lista con el historial de vehículos y sus datos de salida.
- b. vehiculos: Lista con los vehículos actualmente en el parqueadero.
- c. precios: Diccionario con el precio por hora según el tipo de vehículo.
- d. meses: Diccionario con la cantidad de días de cada mes.

## Bucle principal (while True)

- a. Muestra el menú principal y pide al usuario una acción:
  - i. E: Ingresar un vehículo.
  - ii. S: Sacar un vehículo.
  - iii. X: Opciones adicionales (reportes, salir, etc).

## Ingreso de vehículo

- Pide y valida la placa (formato y que no esté repetida).
- Pide y valida el tipo de vehículo.
- Pide y valida la hora de entrada.
- Pide y valida la fecha de entrada.
- Guarda los datos en la lista vehiculos y muestra confirmación.



## Salida de vehículo

- Pide y valida la placa.
- Busca el vehículo en la lista.
- Pide y valida la hora y fecha de salida.
- Calcula el tiempo total y el valor a pagar.
- Muestra el resultado, actualiza el historial y elimina el vehículo de la lista.

## Opciones del programa (X)

- 1: Muestra los vehículos adentro.
- 2: Reporte de ingresos por día (pide fecha y suma pagos del historial).
- 3: Vuelve al menú principal.
- 4: Cierra el programa.
- Opción inválida: muestra mensaje de error.

## Validaciones

- En cada entrada de datos (placa, tipo, hora, fecha) hay validaciones y mensajes de error para asegurar que los datos sean correctos antes de continuar.





# Estructura

Parqueadero > Sistema de Control de Parqueadero.py > ...

```
1 historial=[] #una lista para el historial de los vehiculos ingresados y sus datos de salida
2 vehiculos=[] #una lista para los vehiculos ingresados en el parqueadero actualmente
3 precios={"MOTO":1000,"CARRO":2000,"CAMIONETA":2500} #precios de los vehiculos por hora
4 meses={"01":31,"02":29,"03":31,"04":30,"05":31,"06":30,"07":31,"08":31,"09":30,"10":31,"11":30,"12":31} #cantidad de dias en cada mes
```

- historial: Guarda los datos de los vehículos que ya salieron, junto con lo que pagaron y cuándo salieron.
- vehiculos: Guarda los vehículos que están adentro del parqueadero en este momento.
- precios: Indica cuánto cuesta la hora para cada tipo de vehículo.
- meses: Sirve para validar fechas y calcular cuántos días tiene cada mes.

```
6 while True:
7     print("\n\n")
8     print("PARQUEADERO".center(100,"="))
9     print(f"\nEl precio por hora para cada vehiculo es:\n{precios}\n")
10    print("\n Para entrar un nuevo vehiculo al parqueadero ingrese 'E'\n Para sacar un vehiculo del parqueadero ingrese 'S'\n Para ver mas
    opciones del programa ingrese 'X'\n")
11    entrada=input("\nIngrese la accion que va a realizar: ").upper() #ingresar que desea hacer el usuario
```

- El programa muestra el menú principal y espera que el usuario escriba si quiere entrar un vehículo, sacar uno, o ver opciones.



```

13 if entrada=="E": #si el usuario desea ingresar un vehiculo
14
15
16 #ingresar y verificar la placa
17 comprobador_placa=0 #Variable que ayuda a comprobar si la placa ya está registrada
18 while True:
19     if comprobador_placa==1: #si se detecto la placa repetida
20         break #Sí la placa ya está registrada, se rompe el bucle y se continúa
21     placa=input("\nIngrese la placa del vehiculo (AAA000): ").upper() #Le pide al ususario que ingrese la placa del vehiculo y la coloca en mayousculas
22     #detectar errores en la placa
23     if len(placa)!=6:#si la longitud de la placa es incorrecta
24         print("Placa no valida vuelva a intentarlo\n")
25         continue #si detecta un error vuelve al inicio del bucle a pedir el codigo
26     elif placa[0:3].isalpha()==False or placa[3:5].isdigit()==False: #si los caracteres de la placan son incorrectos
27         print("Placa no valida vuelva a intentarlo\n")
28         continue #si detecta un error vuelve al inicio del bucle a pedir el codigo
29     else: #si no hay ningun error al escribir la placa
30         contador_placa=0 #Contador que recorre la lista de vehículos
31         while True:
32             if len(vehiculos)==0: #Sí no hay vehículos en el parqueadero
33                 break # se rompe el bucle y se continúa con el siguiente
34
35             if placa==vehiculos[contador_placa][0]: #busca en el indice que da el contador placa para ver si la placa ingresada ya esta en una de las placas ya
               ingresadas
36                 print("El vehiculo ya esta ingresado en el parqueadero, no lo puede ingresar de nuevo\n")
37                 comprobador_placa=1 #comprobador placa pasa a ser verdadero indicando que la placa esta repetida
38                 break #Sí la placa ya está registrada, se rompe el bucle y se continúa
39             else: #si la placa del indice actual no es la misma que la ingresada
40                 contador_placa+=1 #Incrementa el contador para seguir buscando en la lista de vehículos
41                 if contador_placa>len(vehiculos): #si el indice que da el contador_placa es mayor o igual que la cantidad de vehiculos registrados
42                     break #sale del bucle de la placa y continua con el siguiente
43         if comprobador_placa==1: #si se detecto una placa repetida vuelve al inicio del bucle
44             continue #Sí la placa ya está, se vuelve a iniciar el bucle para pedir que se ingrese una placa nueva
45         else: #Sí no se detecta una placa repetida
46             break #se rompe este bucle y continua con el siguiente
47 if comprobador_placa==1: #Sí se detecta una placa repetida
48     continue #se vuelve a iniciar el bucle inicial para preguntar si desea ingresar o retirar un vehiculo

```

Esta parte del código se encarga de registrar la entrada de un vehículo al parqueadero y validar que la placa sea correcta y no esté repetida.



1. Pide la placa del vehículo al usuario y la convierte a mayúsculas.
2. Valida la placa:
  - a. Debe tener exactamente 6 caracteres.
  - b. Los primeros 3 deben ser letras y los últimos 3 deben ser números.
  - c. Si la placa no cumple con esto, muestra un mensaje de error y vuelve a pedir la placa.
3. Verifica que la placa no esté repetida:
  - a. Recorre la lista de vehículos que ya están en el parqueadero.
  - b. Si encuentra la misma placa, muestra un mensaje diciendo que ese vehículo ya está adentro y vuelve a pedir la placa.
4. Si la placa es válida y no está repetida, sale del bucle y continúa con el registro del tipo de vehículo.

```
52 #tipo de vehiculo
53 while True:
54     tipo=input("\nIngrese el tipo de vehiculo (Moto, Carro, Camioneta): ").upper()#pide el tipo de vehiculo y lo pone en mayusculas
55     #detectar errores en el tipo de vehiculo
56     if tipo=="MOTO" or tipo=="CARRO" or tipo=="CAMIONETA": #verifica si el tipo que se ingreso es uno de los 3 validos
57         break #si es valido sale del bucle de pedir el tipo del vehiculo
58     else: #detectar si el vehiculo que se ingreso no esta entre los vehiculos validos
59         print("Tipo de vehiculo no valido, vuelva a intentar\n")
60         continue #regresa al inicio del bucle a pedir el tipo de vehiculo
```

1. Pide la hora de entrada al usuario en formato HH:MM y separa la hora y los minutos.
2. Valida que la hora y los minutos sean correctos:
  - a. La hora debe estar entre 0 y 24.
  - b. Los minutos deben estar entre 0 y 59.
  - c. Si hay un error, muestra un mensaje y vuelve a pedir la hora.
3. Valida la cantidad de dígitos:
  - a. La hora y los minutos deben tener 1 o 2 dígitos.
  - b. Si no, muestra un mensaje y vuelve a pedir la hora.
4. Si la hora o los minutos tienen solo un dígito, les agrega un 0 adelante (por ejemplo, 3:5 se convierte en 03:05).





```

53 while True:
54     tipo=input("\nIngresa el tipo de vehiculo (Moto, Carro, Camioneta): ").upper()#pide el tipo de vehiculo y lo pone en mayusculas
55     #detectar errores en el tipo de vehiculo
56     if tipo=="MOTO" or tipo=="CARRO" or tipo=="CAMIONETA": #verifica si el tipo que se ingreso es uno de los 3 validos
57         break #si es valido sale del bucle de pedir el tipo del vehiculo
58     else: #detectar si el vehiculo que se ingreso no esta entre los vehiculos validos
59         print("Tipo de vehiculo no valido, vuelva a intentar\n")
60         continue #regresa al inicio del bucle a pedir el tipo de vehiculo
61
62 #hora de entrada
63 while True:
64     while True:#bucle para pedir la hora
65         try:#Intentar el bloque de codigo
66             hora_entrada,minutos_entrada=input("\nIngresa la hora de entrada (HH:MM): ").split(":")#se le asignan 2 variables a un input() y se divide la cadena de
              caracteres que da el usuario en el input() en dos datos según el caracter que se le asignó al .split()
67             if hora_entrada.isdigit() and minutos_entrada.isdigit():#si las variables son un numero
68                 break#sale del bucle para pedir la hora
69             else:
70                 print("Por favor ingrese una hora valida")
71         except ValueError:#si en alguna parte del bloque de codigo en el try da un ValueError se ejecuta el bloque de codigo del except
72             print("Por favor ingrese la informacion en el formato pedido")#indica el error y se repite el bucle
73     if int(hora_entrada)>24 or int(hora_entrada)<0 or int(minutos_entrada)>59 or int(minutos_entrada)<0: #si se encuentra un error en la cantidad de horas de la entrada
74         print("Hora mal ingresada, tiene que ser en formato 24h, vuelva a intentarlo\n")
75         continue #se volverá a repetir el bucle para pedir otra vez la hora por el continue
76     elif len(hora_entrada)>2 or len(hora_entrada)<1 or len(minutos_entrada)>2 or len(minutos_entrada)<1: #Cuenta los dígitos de la hora y minutos de entrada para asegurarse
              de que estén bien ingresados los datos
77         print("Hora no valida, vuelva a intentarlo\n")
78         continue
79     else:
80         if len(hora_entrada)==1:
81             hora_entrada=f"{0}{hora_entrada}" #Sí la hora que ingresa el usuario sólo tiene 1 dígito, se le agregará un 0 antes de imprimir la hora (ej: pasa la hora 3:30 ->
              03:30)
82         if len(minutos_entrada)==1:
83             minutos_entrada=f"{0}{minutos_entrada}" #Sí los minutos que ingresa el usuario sólo tienen 1 dígito, se le agregará un 0 antes de imprimir los minutos (ej: 11:05)
84         hora_entrada_=(hora_entrada,minutos_entrada)
85         break

```

Pide la hora de entrada al usuario en formato HH:MM. Valida el formato:

Si el usuario no ingresa el formato correcto, muestra un mensaje y vuelve a pedir la hora.

Valida los valores: La hora debe estar entre 0 y 24. Los minutos deben estar entre 0 y 59.

Si no cumplen, muestra un mensaje y vuelve a pedir la hora.

Valida la cantidad de dígitos:

La hora y los minutos deben tener 1 o 2 dígitos.

Si no cumplen, muestra un mensaje y vuelve a pedir la hora.

Formatea los valores: Si la hora o los minutos tienen solo un dígito, les agrega un 0 adelante (por ejemplo, 3:5 se convierte en 03:05).

Guarda la hora de entrada como una tupla y sale del bucle cuando la hora es válida.



```

87 while True:
88     while True:#bucle para pedir el mes
89         try:#Intenta un bloque de codigo
90             fecha_entrada_dia,fecha_entrada_mes=input("\nIngrese la fecha de entrada (DD/MM): ").split("/") #Se nombran 2 variables a las que se les asigna un input, donde
            se divide el dato del input con un .split en 2 datos diferentes, donde se les asigna a las variables nombradas
91             if fecha_entrada_dia.isdigit() and fecha_entrada_mes.isdigit():#si las variables son un numero
92                 break#sale del bucle para pedir la hora
93             else:
94                 print("Por favor ingrese una fecha valida")
95         except ValueError: #si en alguna parte del bloque de codigo en el try da un ValueError se ejecuta el bloque de codigo del except
96             print("Por favor ingrese la informacion en el formato pedido") #indica el error
97 if len(fecha_entrada_dia)>2 or len(fecha_entrada_dia)<1 or len(fecha_entrada_mes)>2 or len(fecha_entrada_mes)<1: #Creamos 2 len para cada variable para que cuenten sus
dígitos y reinicie el bucle cuando el usuario ingrese más de 2 dígitos o menos de 1 dígito
98     print("Fecha mal ingresada, tiene que ser en formato DD/MM, vuelva a intentarlo\n")
99     continue
100 if len(fecha_entrada_dia)==1:
101     fecha_entrada_dia=f"{0}{fecha_entrada_dia}" #Se cuentan los caracteres de la variable, donde si es igual a 1, se le agregue un 0 antes de imprimir el dígito que
    ingresó el usuario (ej: pasa de 9 a 09 -> 09/11)
102 if len(fecha_entrada_mes)==1:
103     fecha_entrada_mes=f"{0}{fecha_entrada_mes}" #Se cuentan los caracteres de la variable, donde si es igual a 1, se le agregue un 0 antes de imprimir el dígito que
    ingresó el usuario (ej: pasa de 3 a 03 -> 23/04)
104
105 #meses
106 if int(fecha_entrada_mes)<=12 and int(fecha_entrada_mes)>=1: #Se asegura de que el número del mes ingresado por el usuario esté entre el 1 y el 12
107     if fecha_entrada_mes=="01" or fecha_entrada_mes=="03" or fecha_entrada_mes=="05" or fecha_entrada_mes=="07" or fecha_entrada_mes=="08" or fecha_entrada_mes=="10" or
    fecha_entrada_mes=="12": #Mira sí el mes ingresado por el usuario es alguno de los que tienen 31 días
108         if int(fecha_entrada_dia)>31 or int(fecha_entrada_dia)<1:
109             print("Fecha incorrecta, el mes tiene 31 dias, vuelva a intentarlo\n")
110             continue #Hace que el usuario vuelva a ingresar la fecha cuando el día que ingresa es mayor a 31 o menor a 1 (negativo)
111         else:
112             break #Rompe el bucle para continuar cuando el usuario ingresa la fecha correctamente

```



```

101         else:
102             break #Rompe el bucle para continuar cuando el usuario ingresa la fecha correctamente
103     elif fecha_entrada_mes=="02": #Mira sí el mes que ingresó el usuario tiene 29 días (Febrero/Año biciesto)
104         if int(fecha_entrada_dia)>29 or int(fecha_entrada_dia)<1:
105             print("Fecha incorrecta, el mes tiene maximo 29 días, vuelva a intentar\n")
106             continue #Hace que el usuario vuelva a ingresar la fecha cuando el día que ingresa es mayor a 29 o menor a 1 (negativo)
107         else:
108             break #Rompe el bucle cuando el usuario ingresa la fecha correctamente
109     else:
110         print("fecha incorrecta, el mes no existe, vuelva a intentar\n")
111         continue #Hace que el usuario vuelva a ingresar la fecha cuando ingresa un número que no pertenece a ningún mes
112 fecha_entrada=(fecha_entrada_dia,fecha_entrada_mes)

```

1. Verifica que el mes esté entre 1 y 12
  - a. Si el mes es válido, revisa cuántos días tiene ese mes:
    - i. Meses de 31 días: Si el mes es enero, marzo, mayo, julio, agosto, octubre o diciembre, el día debe estar entre 1 y 31.
    - ii. Meses de 30 días: Si el mes es abril, junio, septiembre o noviembre, el día debe estar entre 1 y 30.
    - iii. Febrero: El día debe estar entre 1 y 29.
  - b. Si el día no es válido para ese mes, muestra un mensaje de error y vuelve a pedir la fecha.
2. Si el mes no existe (no está entre 1 y 12), muestra un mensaje de error y vuelve a pedir la fecha.

```

114
115     vehiculos.append((placa, tipo, hora_entrada_, fecha_entrada)) #Agrega los datos que ingresa el usuario (placa, tipo, hora de entrada y fecha de entrada) a la lista
    de vehiculos
116     print(f"\nLos datos ingresados son:\nPlaca: {placa}\nTipo: {tipo}\nHora de entrada: {hora_entrada_[0]}:{hora_entrada_[1]}\nFecha de entrada: {fecha_entrada[0]}/{
    fecha_entrada[1]}")
117

```

1. Guarda el vehículo:
2. Agrega a la lista vehiculos una tupla con la placa, el tipo, la hora y la fecha de entrada del vehículo.
3. Muestra los datos ingresados:
4. Imprime en pantalla todos los datos que el usuario acaba de ingresar para confirmar que se registraron correctamente.





```

135 elif entrada=="S":
136     if len(vehiculos)!=0: #si hay vehiculos en el parqueadero
137         while True:
138             placa_salida=input("\nIngrese la placa de su vehiculo para salir: ").upper() #Le pide al usuario que ingrese la placa del vehículo que desea sacar del parqueadero y
            la coloca en mayúsculas
139             if len(placa_salida)!=6: #longitud de la placa
140                 print("Placa no valida vuelva a intentarlo\n")
141                 continue #Sí la longitud de la placa es diferente a 6, vuelve al inicio del bucle a pedir la placa
142             elif placa_salida[0:3].isalpha()==False or placa_salida[3:5].isdigit()==False: #ver los caracteres de la placa
143                 print("Placa no valida vuelva a intentarlo\n")
144                 continue #Sí la placa no tiene los caracteres correctos, vuelve al inicio del bucle a pedir la placa
145             else:
146                 break #si no hay errores con la placa sale del bucle de pedir la placa y continua con el siguiente
147         contador=0
148         while True:
149             if vehiculos[contador][0]==placa_salida: #verifica que la placa de salida que ingresó el usuario esté en la lista de vehiculos
150                 while True:
151                     while True:#bucle para pedir la hora
152                         try:#Intenta un bloque de codigo
153                             hora_salida,minutos_salida=input("\nIngrese la hora de salida (HH:MM): ").split(":") #Divide la cadena de caracteres (la hora) en datos según el
                                caracter que se le asignó
154                             if hora_salida.isdigit() and minutos_salida.isdigit():#si las variables son un numero
155                                 break#sale del bucle para pedir la hora
156                             else:
157                                 print("Por favor ingrese una hora valida")
158                             except:#si en alguna parte del bloque de codigo en el try da un ValueError se ejecuta el bloque de codigo del except
159                                 print("Por favor ingrese la informacion en el formato pedido")#indica el error
160                             if int(hora_salida)>24 or int(hora_salida)<0 or int(minutos_salida)>59 or int(minutos_salida)<0: #verifica que la hora y los minutos ingresados por el
                                usuario estén dentro de los rangos correctos
161                                 print("Hora mal ingresada, tiene que ser en formato 24h, vuelva a intentarlo\n")
162                                 continue #Sí la hora ingresada por el usuario es mayor a 24 o menor a 0, o los minutos son mayores a 59 o menores a 0, se vuelve a pedir la hora de salida
163                             elif len(hora_salida)>2 or len(hora_salida)<1 or len(minutos_salida)>2 or len(minutos_salida)<1:
164                                 print("Hora no valida, vuelva a intentarlo\n")
165                                 continue #Sí los caracteres de la hora ingresada por el usuario son mayor a 2 o menor a 1, se vuelve a repetir el bucle para pedir la hora de salida de
                                nuevo
166                             else:
167                                 if len(hora_salida)==1: #Cuenta los caracteres de la variable, donde si es igual a 1, se le agregue un 0 antes de imprimir la hora (ej: pasa de 3 a 03 ->
                                03:30)
168                                     hora_salida=f"{0}{hora_salida}"
169                                 if len(minutos_salida)==1: #Cuenta los caracteres de la variable, donde si es igual a 1, se le agregue un 0 antes de imprimir los minutos (ej: pasa de 5
                                a 05 -> 11:05)
170                                     minutos_salida=f"{0}{minutos_salida}"
171                                 hora_salida_=(hora_salida,minutos_salida) #Se crea una variable que almacena la hora de salida y los minutos de salida en una tupla
172                                 break #Rompe el bucle cuando ya la hora de salida y los minutos de salida son correctos

```



```

173 while True:
174     while True:#bucle para pedir la fecha
175         try:#Intenta un bloque de codigo
176             fecha_salida_dia,fecha_salida_mes=input("\nIngrese la fecha de salida (DD/MM): ").split("/") #Divide la cadena de caracteres (la fecha) en datos
según el caracter asignado
177             if fecha_salida_dia.isdigit() and fecha_salida_mes.isdigit():#si las variables son un numero
178                 break#sale del bucle para pedir la hora
179             else:
180                 print("Por favor ingrese una fecha valida")
181         except ValueError:#si en alguna parte del bloque de codigo en el try da un ValueError se ejecuta el bloque de codigo del except
182             print("Por favor ingrese la informacion en el formato pedido")#indica el error
183 if len(fecha_salida_dia)>2 or len(fecha_salida_dia)<1 or len(fecha_salida_mes)>2 or len(fecha_salida_mes)<1:
184     print("Fecha mal ingresada, tiene que ser en formato DD/MM, vuelva a intentarlo\n")
185     continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo cuando la fecha ingresada tiene más de 2 dígitos o
menos de 1 dígito
186 if len(fecha_salida_dia)==1:
187     fecha_salida_dia=f"{0}{fecha_salida_dia}" #Sí el día de la fecha de salida ingresada por el usuario sólo tiene 1 dígito, se le agregará un 0 antes de
imprimir el día (ej: pasa de 9 a 09 -> 09/11)
188 if len(fecha_salida_mes)==1:
189     fecha_salida_mes=f"{0}{fecha_salida_mes}" #Sí el mes de la fecha de salida ingresada por el usuario sólo tiene 1 dígito, se le agregará un 0 antes de
imprimir el mes (ej: pasa de 4 a 04 -> 23/04)
190
191 #meses
192 if int(fecha_salida_mes)<=12 and int(fecha_salida_mes)>=1:
193     if fecha_salida_mes=="01" or fecha_salida_mes=="03" or fecha_salida_mes=="05" or fecha_salida_mes=="07" or fecha_salida_mes=="08" or
fecha_salida_mes=="10" or fecha_salida_mes=="12": #Verifica que el mes ingresado por el usuario sea uno de los que tienen 31 días
194         if int(fecha_salida_dia)>31 or int(fecha_salida_dia)<1:
195             print("Fecha incorrecta, el mes tiene 31 dias, vuelva a intentarlo\n")
196             continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo cuando ingresa un mes de 31 días y el día que
ingresa es mayor a 31 o menor a 1 (negativo)
197         else:
198             break #Rompe el bucle cuando el usuario ingresa la fecha correctamente
199     elif fecha_salida_mes=="04" or fecha_salida_mes=="06" or fecha_salida_mes=="09" or fecha_salida_mes=="11": #Verifica que el mes ingresado por el usuario
sea uno de los que tienen 30 días
200         if int(fecha_salida_dia)>30 or int(fecha_salida_dia)<1:
201             print("Fecha incorrecta, el mes tiene 30 dias, vuelva a intentar\n")
202             continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo cuando ingresa un mes de 30 días y el día que
ingresa es mayor a 30 o menor a 1 (negativo)
203         else:
204             break #Rompe el bucle cuando el usuario ingresa la fecha correctamente
205     elif fecha_salida_mes=="02":
206         if int(fecha_salida_dia)>29 or int(fecha_salida_dia)<1: #Verifica que el mes ingresado por el usuario sea Febrero (posibilidad de año biciesto)
207             print("Fecha incorrecta, el mes tiene maximo 29 dias, vuelva a intentar\n")

```





```

208         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo cuando ingresa el mes 2 (febrero) y el día que
        ingresa es mayor a 29 o menor a 1 (negativo)
209     else:
210         break #Rompe el bucle cuando el usuario ingresa la fecha correctamente
211 else:
212     print("fecha incorrecta, el mes no existe, vuelva a intentar\n")
213     continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo cuando ingresa un mes que no existe
214 fecha_salida=(fecha_salida_dia,fecha_salida_mes) #Se crea una variable que almacena la fecha de salida (día y mes) en una tupla
215
216
217 hora_minutos_entrada=(int(vehiculos[contador][2][0])*60)+int(vehiculos[contador][2][1]) #Convierte la hora de entrada en minutos para sumarla con los minutos de
    entrada
218 hora_minutos_salida=(int(hora_salida)*60)+int(minutos_salida) #Convierte la hora de salida en minutos para sumarla con los minutos de salida y obtener el tiempo
    total de parqueo de los vehículos en minutos
219 #mismo mes
220 if fecha_salida[1]==vehiculos[contador][3][1]:
221     dias_parqueo=int(fecha_salida[0])-int(vehiculos[contador][3][0]) #Sirve para calcular la diferencia de días entre la fecha de entrada y la fecha de salida
    del vehículo, dando así los días de parqueo
222     if dias_parqueo<0:
223         print("Datos de salida incorrectos, ingrese los datos nuevamente\n")
224         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo cuando la fecha de salida es menor a la fecha de
    entrada
225     else:
226         horas_parqueo=hora_minutos_salida-hora_minutos_entrada #Calcula la diferencia de horas entre la hora de entrada y la hora de salida del vehículo
227         if int(vehiculos[contador][3][0])==int(fecha_salida[0]):
228             if horas_parqueo<0:
229                 print("Datos de salida incorrectos, ingrese los datos nuevamente\n")
230                 continue
231             else:
232                 horas_parqueo=round(horas_parqueo/60,2) #Pasa el tiempo de parqueo de minutos a horas, y luego el resultado se redondea a 2 decimales
233                 dias_parqueo*=24 #Convierte los días de parqueo a horas multiplicando por 24
234                 horas_parqueo+=dias_parqueo #Suma las horas de parqueo con los días de parqueo ya pasados a horas
235                 total_pagar=horas_parqueo*precios[vehiculos[contador][1]] #Calcula el total a pagar por el tiempo de parqueo multiplicando las horas totales de
    parqueo por el precio del tipo de vehículo que ingresó el usuario
236                 print(f" Su vehiculo estuvo {horas_parqueo} horas en el parqueadero ".center(20,"-"),"\n",f" Debe pagar: {total_pagar}$ ".center(20,"-"))
                #Imprime el tiempo total de parqueo y el total a pagar por el tiempo de parqueo (los .center son para poner en medio el texto con caracteres de
    relleno, en este caso guiones)
237                 historial.append((vehiculos[contador],(total_pagar,hora_salida,fecha_salida))) #Agrega los datos del vehículo, el total a pagar, la hora de
    salida y la fecha de salida al historial para tener los registros del parqueadero
238                 vehiculos.remove(vehiculos[contador]) #Elimina el vehículo que salió de la lista de los vehículos en el parqueadero
239                 break #Rompe el bucle cuando ya se ha terminado de tomar el registro del vehículo que salió del parqueadero

```



```

240 elif int(vehiculos[contador][3][0])>int(fecha_salida[0]):
241     print("Datos de salida incorrectos, ingrese los datos nuevamente\n")
242     continue
243 else:
244     horas_parqueo=round(horas_parqueo/60,2) #Pasa el tiempo de parqueo de minutos a horas, y luego el resultado se redondea a 2 decimales
245     dias_parqueo*=24 #Convierte los días de parqueo a horas multiplicando por 24
246     horas_parqueo+=dias_parqueo #Suma las horas de parqueo con los días de parqueo ya pasados a horas
247     total_pagar=horas_parqueo*precios[vehiculos[contador][1]] #Calcula el total a pagar por el tiempo de parqueo multiplicando las horas totales de
    parqueo por el precio del tipo de vehículo que ingresó el usuario
248     print(f" Su vehiculo estuvo {horas_parqueo} horas en el parqueadero ".center(20,"-"),"\n",f" Debe pagar: {total_pagar}$ ".center(20,"-")) #Imprime el
    tiempo total de parqueo y el total a pagar por el tiempo de parqueo (los .center son para poner en medio el texto con caracteres de relleno, en este
    caso guiones)
249     historial.append((vehiculos[contador],(total_pagar,hora_salida,fecha_salida))) #Agrega los datos del vehículo, el total a pagar, la hora de salida y
    la fecha de salida al historial para tener los registros del parqueadero
250     vehiculos.remove(vehiculos[contador]) #Elimina el vehículo que salió de la lista de los vehículos en el parqueadero
251     break #Rompe el bucle cuando ya se ha terminado de tomar el registro del vehículo que salió del parqueadero
252
253
254 elif int(fecha_salida[1])>int(vehiculos[contador][3][1]): #Verifica si la fecha de salida es mayor a la fecha de entrada del vehículo
255     contador_dias=vehiculos[contador][3][1] #Obtiene el mes de la fecha de entrada del vehículo
256     dias_parqueo=0
257     while int(contador_dias)<=int(fecha_salida[1]):
258         dias_parqueo+=meses[contador_dias] #Mientras el mes de la fecha de entrada sea menor o igual al mes de la fecha de salida, se suman los días de parqueo
259         if int(contador_dias)<10:
260             contador_dias=f"{0}{int(contador_dias)+1}" #Agrega un 0 antes del número del mes si es menor a 10 para que tenga el formato correcto
261         else:
262             contador_dias=int(contador_dias)+1
263     dias_parqueo-=meses[fecha_salida[1]]-int(fecha_salida[0]) #Resta los días del mes de la fecha de salida menos el día de la fecha de salida ingresada por el
    usuario
264     dias_parqueo-=int(vehiculos[contador][3][0]) #Resta el día de la fecha de entrada del vehículo a los días de parqueo calculados
265     dias_parqueo*=24 #Convierte los días de parqueo a horas multiplicando por 24
266     if dias_parqueo<0:
267         print("Datos de salida incorrectos, ingrese los datos nuevamente\n")
268         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de salida de nuevo si la fecha de salida es menor a la fecha de entrada
269     else:
270         hora_minutos_entrada=(int(vehiculos[contador][2][0])*60)+int(vehiculos[contador][2][1]) #Convierte la hora de entrada en minutos para sumarla con los
    minutos de entrada
271         hora_minutos_salida=(int(hora_salida)*60)+int(minutos_salida) #Convierte la hora de salida en minutos para sumarla con los minutos de salida y obtener el
    tiempo total de parqueo de los vehículos en minutos
272         horas_parqueo=hora_minutos_salida-hora_minutos_entrada #Calcula la diferencia de horas entre la hora de entrada y la hora de salida del vehículo
273         if horas_parqueo<0:

```





```

274         print("Datos de salida incorrectos, ingrese los datos nuevamente\n")
275         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese los datos de salida de nuevo cuando se ingresan de manera incorrecta
276     else:
277         horas_parqueo=round(horas_parqueo/60,2) #Pasa el tiempo de parqueo de minutos a horas, y luego el resultado es redondeado a 2 decimales
278         horas_parqueo+=dias_parqueo #Suma las horas de parqueo con los días de parqueo ya pasados a horas
279         total_pagar=horas_parqueo*precios[vehiculos[contador][1]] #Calcula el total a pagar por el tiempo de parqueo multiplicando las horas totales de
280         parqueo por el precio del tipo de vehículo que ingresó el usuario
281         print(f" Su vehículo estuvo {horas_parqueo} horas en el parqueadero ".center(20,"-"),"\n",f" Debe pagar: {total_pagar}$ ".center(20,"-")) #Imprime el
282         tiempo total de parqueo y el total a pagar por el tiempo de parqueo (los .center son para poner en medio el texto con caracteres de relleno, en este
283         caso guiones)
284         historial.append((vehiculos[contador],(total_pagar,hora_salida,fecha_salida))) #Agrega los datos del vehículo, el total a pagar, la hora de salida y
285         la fecha de salida al historial para tener los registros del parqueadero
286         vehiculos.remove(vehiculos[contador]) #Elimina el vehículo que salió de la lista de los vehículos en el parqueadero
287         break #Rompe el bucle cuando ya se ha terminado de tomar el registro del vehículo que salió del parqueadero
288
289     elif int(fecha_salida[1])<int(vehiculos[contador][3][1]):
290         print("Datos de salida incorrectos, ingrese los datos nuevamente\n")
291         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese los datos de salida de nuevo cuando la fecha de salida es menor a la fecha de entrada
292         del vehículo
293     else:
294         contador+=1
295         if contador>=len(vehiculos):
296             print("El vehiculo no esta registrado en el parqueadero\n")
297             break #Sí no se encuentra la placa del vehículo en la lista de vehículos, se le notificará al usuario y se romperá el bucle
298
299 else:
300     print("No hay vehiculos en el parqueadero\n") #Sí no hay vehículos en el parqueadero, se le notificará al usuario
301

```

## 1. Valida la fecha de salida:

- Pide la fecha en formato DD/MM y separa día y mes.
- Verifica que el día y el mes tengan 1 o 2 dígitos, y si solo tienen uno, les agrega un 0 adelante.
- Revisa que el mes esté entre 1 y 12.
- Según el mes, revisa que el día sea válido (31, 30 o 29 días según el mes).
- Si hay un error, muestra un mensaje y vuelve a pedir la fecha.
- Cuando la fecha es válida, la guarda como una tupla.

## 2. Convierte la hora de entrada y salida a minutos para facilitar el cálculo del tiempo total.



3. Calcula el tiempo de parqueo y el valor a pagar:

a. Si la fecha de entrada y salida es el mismo mes:

i. Calcula la diferencia de días y horas.

ii. Si la fecha de salida es menor que la de entrada, muestra error y vuelve a pedir los datos.

iii. Si todo está bien, convierte el tiempo total a horas y calcula el total a pagar multiplicando por el precio según el tipo de vehículo.

iv. Muestra cuántas horas estuvo el vehículo y cuánto debe pagar.

v. Guarda el registro en el historial y elimina el vehículo de la lista de adentro.

b. Si la fecha de salida es en un mes diferente:

i. Suma los días de los meses intermedios para calcular el total de días.

ii. Convierte los días a horas y suma las horas de entrada y salida.

iii. Calcula el total a pagar y muestra la información.

iv. Guarda el registro en el historial y elimina el vehículo de la lista.

c. Si la fecha de salida es menor que la de entrada (mes anterior):

i. Muestra un mensaje de error y vuelve a pedir los datos.

4. Si la placa no se encuentra en la lista de vehículos:

a. Muestra un mensaje diciendo que el vehículo no está registrado y termina el proceso.

5. Si no hay vehículos en el parqueadero:

a. Muestra un mensaje indicando que no hay vehículos para sacar.

```
296 elif entrada=="X": #Sí el usuario ingresa 'X', se le mostrarán las opciones del programa
297     print("\n\n")
298     print("Opciones del programa".center(20,"-")) #Imprime el título centrado con guiones
299     print("\ningrese 1 para ver las vehiculos adentro del parqueadero\nIngrese 2 para ver el reporte de ingresos por un dia\nIngrese 3 para volver al menu inicial\nIngrese 4
    para salir del programa")
300     ajustes=int(input("Ingrese la accion que desee realizar: ")) #Pide al usuario que ingrese una opción del menú de ajustes
```



```

301 if ajustes==1:
302     if len(vehiculos)!=0: #Cuenta la cantidad de vehículos en el parqueadero
303         contador_vehiculos=0 #Contador que ayuda a recorrer la lista de vehículo
304         print("\nLos vehiculos en el parqueadero actualmente son:")
305         while len(vehiculos)>contador_vehiculos: #Mientras la cantidad de vehículos sea mayor al contador, se imprimirá la información de los vehículos
306             print("".center(100,"=")) #Imprime una línea de separación con guiones para que se vea más ordenado
307             print(f"{{(vehiculos[contador_vehiculos][1]).lower()}} con placa {{vehiculos[contador_vehiculos][0]}}\n    Hora de entrada: {{vehiculos[contador_vehiculos][2][0]}}:
308             {{vehiculos[contador_vehiculos][2][1]}}\n    Fecha de entrada: {{vehiculos[contador_vehiculos][3][0]}}/{{vehiculos[contador_vehiculos][3][1]}}\n") #Imprime los datos
309             de cada vehículo que hay en el parqueadero, donde se accede a los datos de la lista de vehículos con el contador que la recorre
310             print("".center(100,"="))
311             contador_vehiculos+=1 #Incrementa el contador para seguir recorriendo la lista de vehículos
312         print(f"\n\n Hay un total de {{len(vehiculos)}} en el interior del parqueadero actualmente\n") #Imprime la cantidad de vehículos que hay en el parqueadero
313     else:
314         print("\nNo hay ningun vehiculo en el parqueadero") #Sí la cantidad de vehículos es igual a 0, muestra un mensaje indicando que no hay vehículos en el parqueadero

```

Muestra un submenú de opciones:

- 1: Ver los vehículos actualmente adentro del parqueadero.
- 2: Ver el reporte de ingresos de un día específico.
- 3: Volver al menú inicial.
- 4: Salir del programa.

Si el usuario elige la opción 1:

Si hay vehículos en el parqueadero, imprime la información de cada uno (tipo, placa, hora y fecha de entrada).

Si no hay vehículos, muestra un mensaje indicando que el parqueadero está vacío.





```

313 elif ajustes==2:
314     while True:
315         while True:#bucle para pedir la fecha
316             try:#Intenta un bloque de codigo para pedir la fecha
317                 fecha_ajustes_dia,fecha_ajustes_mes=input("\nIngrese la fecha para el reporte de ingresos diario (DD/MM): ").split("/") #Combina la fecha de ajustes en una
                 variable, donde se divide el dato del input con un .split en 2 datos diferentes, donde se les asigna a las variables nombradas
318                 if fecha_ajustes_dia.isdigit() and fecha_ajustes_mes.isdigit():#si las variables son un numero
319                     break#sale del bucle para pedir la hora
320                 else:
321                     print("Por favor ingrese una fecha valida")
322             except ValueError:#si en alguna parte del bloque de codigo en el try da un ValueError se ejecuta el bloque de codigo del except
323                 print("Por favor ingrese la informacion en el formato pedido")#indica el error
324 if len(fecha_ajustes_dia)>2 or len(fecha_ajustes_dia)<1 or len(fecha_ajustes_mes)>2 or len(fecha_ajustes_mes)<1: #Cuenta los dígitos de la fecha de ajustes para
asegurarse de que estén bien ingresados los datos
325     print("Fecha mal ingresada, tiene que ser en formato DD/MM, vuelva a intentarlo\n")
326     continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de ajustes de nuevo con el formato indicado
327 if len(fecha_ajustes_dia)==1:
328     fecha_ajustes_dia=f"{0}{fecha_ajustes_dia}" #Sí los dígitos del día de la fecha de ajustes son igual a 1, se le agregará un 0 antes de imprimir el dígito que
ingrese el usuario (ej: pasa de 8 a 08 -> 08/12)
329 if len(fecha_ajustes_mes)==1:
330     fecha_ajustes_mes=f"{0}{fecha_ajustes_mes}" #Sí los dígitos del mes de la fecha de ajustes son igual a 1, se le agregará un 0 antes de imprimir el dígito que
ingrese el usuario (ej: pasa de 5 a 05 -> 23/05)
331
332 #meses
333 if int(fecha_ajustes_mes)<=12 and int(fecha_ajustes_mes)>=1: #Verifica que el número del mes ingresado por el usuario esté entre el 1 y el 12
334     if fecha_ajustes_mes=="01" or fecha_ajustes_mes=="03" or fecha_ajustes_mes=="05" or fecha_ajustes_mes=="07" or fecha_ajustes_mes=="08" or fecha_ajustes_mes=="10"
or fecha_ajustes_mes=="12": #Verifica que el mes ingresado por el usuario sea uno de los que tienen 31 días
335         if int(fecha_ajustes_dia)>31 or int(fecha_ajustes_dia)<1:
336             print("Fecha incorrecta, el mes tiene 31 días, vuelva a intentarlo\n")
337             continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de nuevo cuando ingresa un mes de 31 días y el día que ingresa es mayor a
31 o menor a 1 (negativo)
338         else:
339             break #Rompe el bucle cuando el usuario ingresa la fecha correctamente
340 elif fecha_ajustes_mes=="04" or fecha_ajustes_mes=="06" or fecha_ajustes_mes=="09" or fecha_ajustes_mes=="11": #Verifica que el mes ingresado por el usuario sea
uno de los que tienen 30 días
341     if int(fecha_ajustes_dia)>30 or int(fecha_ajustes_dia)<1:
342         print("Fecha incorrecta, el mes tiene 30 días, vuelva a intentar\n")
343         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de nuevo cuando ingresa un mes de 30 días y el día que ingresa es mayor a
30 o menor a 1 (negativo)
344     else:
345         break #Rompe el bucle cuando el usuario ingresa la fecha correctamente

```





```

346 elif fecha_ajustes_mes=="02": #Verifica que el mes ingresado por el usuario sea Febrero (posibilidad de año biciesto)
347     if int(fecha_ajustes_dia)>29 or int(fecha_ajustes_dia)<1:
348         print("Fecha incorrecta, el mes tiene maximo 29 dias, vuelva a intentar\n")
349         continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de nuevo cuando ingresa el mes 2 (febrero) y el día que ingresa es mayor
           a 29 o menor a 1 (negativo)
350     else:
351         break #Rompe el bucle cuando el usuario ingresa la fecha correctamente
352 else:
353     print("fecha incorrecta, el mes no existe, vuelva a intentar\n")
354     continue #Vuelve a iniciar el bucle, pidiéndole al usuario que ingrese la fecha de nuevo cuando ingresa un mes que no existe
355 fecha_ajustes=(fecha_ajustes_dia,fecha_ajustes_mes)
356 dia_reporte=[]
357 ajustes_dia_reporte=0
358 while ajustes_dia_reporte<len(historial): #Recorre el historial de vehículos para buscar los reportes del día especificado
359     if historial[ajustes_dia_reporte][1][2]==fecha_ajustes: #Verifica que alguna fecha del historial de reportes coincida con la fecha ingresada por el usuario
360         dia_reporte.append(historial[ajustes_dia_reporte][1][0]) #Agrega el total a pagar del reporte del día especificado a la lista
361         ajustes_dia_reporte+=1 #Incrementa el contador para seguir buscando en el historial de reportes
362     else:
363         ajustes_dia_reporte+=1 #Incrementa el contador para seguir buscando en el historial de reportes
364 if len(dia_reporte)==0:
365     print("\nEse dia no hubieron ingresos en el parqueadero\n") #Sí no hay reportes del día especificado, se le notificará al usuario
366 else:
367     print(f"\nEl total de los ingresos del dia: {fecha_ajustes[0]}/{fecha_ajustes[1]} son:\n {sum(dia_reporte)}$") #Imprime el total de los ingresos del día especificado
           sumando todos los reportes del historial que coincidan con la fecha ingresada por el usuario
368

```

1. Valida la fecha ingresada para el reporte:
  - a. Verifica que el día y el mes tengan 1 o 2 dígitos.
  - b. Si solo tienen un dígito, les agrega un 0 adelante.
  - c. Comprueba que el mes esté entre 1 y 12 y que el día sea válido para ese mes (31, 30 o 29 días según el mes).
  - d. Si hay un error, muestra un mensaje y vuelve a pedir la fecha.
2. Guarda la fecha válida como una tupla fecha\_ajustes.
3. Busca en el historial todos los ingresos que coincidan con esa fecha:
  - a. Recorre el historial de vehículos.
  - b. Si la fecha de salida de un registro coincide con la fecha buscada, agrega el valor pagado a una lista.
4. Muestra el resultado:
  - a. Si no hubo ingresos ese día, muestra un mensaje avisando al usuario.
  - b. Si hubo ingresos, suma todos los valores y muestra el total ganado ese día.



```

368
369 elif ajustes==3:
370     continue #Sí el usuario ingresa 3, se vuelve al menú inicial del programa para que pueda realizar otra acción
371
372 elif ajustes==4:
373     print("Se cierra el parqueadero".center(50,"="),"\n\n", f"Programa finalizado".center(50," "))
374     break #Sí el usuario ingresa 4, se cierra el programa y se imprime un mensaje que indica que el parqueadero se ha cerrado y el programa ha finalizado
375 else:
376     print("Accion no valida, ingrese otra\n")
377     continue #Sí el usuario ingresa una opción que no está en el menú de ajustes, se le notificará y se le pedirá que ingrese una opción válida
378 else:
379     print("\nAccion no valida, ingrese una valida\n")
380     continue
381

```

1. Si el usuario ingresa 3:  
Usa continue para volver al menú principal, permitiendo que el usuario realice otra acción.
2. Si el usuario ingresa 4:  
Imprime un mensaje indicando que el parqueadero se cierra y el programa finaliza con break.
3. Si el usuario ingresa cualquier otra opción:  
Muestra un mensaje de error y vuelve a pedir una opción válida.



# CÓDIGO INVESTIGADO

---

## **.center()**

---

Es un método que centra el texto dentro de una cadena, rellenando con espacios u otros caracteres hasta alcanzar una longitud determinada.

## **.isalpha()**

---

Método que determina si todos los caracteres de una cadena son letras. No permite espacios, números ni símbolos.

## **.split()**

---

Método que divide una cadena en varias partes, usando un separador, y devuelve una lista con los fragmentos resultantes.

## **Try-Except**

---

Estructura que permite manejar errores en la ejecución del programa, evitando que se interrumpa si ocurre una excepción.



# Entradas





=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: e

Ingrese la placa del vehiculo (AAA000): AAA111

Ingrese el tipo de vehiculo (Moto, Carro, Camioneta): Moto

Ingrese la hora de entrada (HH:MM): 2:27

Ingrese la fecha de entrada (DD/MM): 10/07

Los datos ingresados son:

Placa: AAA111

Tipo: MOTO

Hora de entrada: 02:27

Fecha de entrada: 10/07



=====PARQUEADERO=====

El precio por hora para cada vehiculo es:  
{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'  
Para sacar un vehiculo del parqueadero ingrese 'S'  
Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: e

Ingrese la placa del vehiculo (AAA000): AAA222

Ingrese el tipo de vehiculo (Moto, Carro, Camioneta): Carro

Ingrese la hora de entrada (HH:MM): 2:29

Ingrese la fecha de entrada (DD/MM): 10/07

Los datos ingresados son:

Placa: AAA222

Tipo: CARRO

Hora de entrada: 02:29

Fecha de entrada: 10/07



```
=====PARQUEADERO=====

El precio por hora para cada vehiculo es:
{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'
Para sacar un vehiculo del parqueadero ingrese 'S'
Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: e

Ingrese la placa del vehiculo (AAA000): AAA333

Ingrese el tipo de vehiculo (Moto, Carro, Camioneta): moto

Ingrese la hora de entrada (HH:MM): 2:39

Ingrese la fecha de entrada (DD/MM): 10/07

Los datos ingresados son:
Placa: AAA333
Tipo: MOTO
Hora de entrada: 02:39
Fecha de entrada: 10/07
```



=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: e

Ingrese la placa del vehiculo (AAA000): AAA444

Ingrese el tipo de vehiculo (Moto, Carro, Camioneta): camioneta

Ingrese la hora de entrada (HH:MM): 2:41

Ingrese la fecha de entrada (DD/MM): 10/07

Los datos ingresados son:

Placa: AAA444

Tipo: CAMIONETA

Hora de entrada: 02:41

Fecha de entrada: 10/07





=====PARQUEADERO=====

El precio por hora para cada vehiculo es:  
{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'  
Para sacar un vehiculo del parqueadero ingrese 'S'  
Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: e

Ingrese la placa del vehiculo (AAA000): AAA555

Ingrese el tipo de vehiculo (Moto, Carro, Camioneta): camioneta

Ingrese la hora de entrada (HH:MM): 2:42

Ingrese la fecha de entrada (DD/MM): 10/07

Los datos ingresados son:

Placa: AAA555

Tipo: CAMIONETA

Hora de entrada: 02:42

Fecha de entrada: 10/07



# Salidas



=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: s

Ingrese la placa de su vehiculo para salir: AAA111

Ingrese la hora de salida (HH:MM): 18:57

Ingrese la fecha de salida (DD/MM): 10/07

Su vehiculo estuvo 16.5 horas en el parqueadero

Debe pagar: 16500.0\$





=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: s

Ingrese la placa de su vehiculo para salir: AAA222

Ingrese la hora de salida (HH:MM): 14:54

Ingrese la fecha de salida (DD/MM): 10/07

Su vehiculo estuvo 12.42 horas en el parqueadero

Debe pagar: 24840.0\$



=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: s

Ingrese la placa de su vehiculo para salir: AAA333

Ingrese la hora de salida (HH:MM): 6:47

Ingrese la fecha de salida (DD/MM): 10/07

Su vehiculo estuvo 4.13 horas en el parqueadero

Debe pagar: 4130.0\$



=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: s

Ingrese la placa de su vehiculo para salir: s

Placa no valida vuelva a intentarlo

Ingrese la placa de su vehiculo para salir: AAA444

Ingrese la hora de salida (HH:MM): 2:58

Ingrese la fecha de salida (DD/MM): 10/08

Su vehiculo estuvo 744.28 horas en el parqueadero

Debe pagar: 1860700.0\$



=====PARQUEADERO=====

El precio por hora para cada vehiculo es:

{'MOTO': 1000, 'CARRO': 2000, 'CAMIONETA': 2500}

Para entrar un nuevo vehiculo al parqueadero ingrese 'E'

Para sacar un vehiculo del parqueadero ingrese 'S'

Para ver mas opciones del programa ingrese 'X'

Ingrese la accion que va a realizar: s

Ingrese la placa de su vehiculo para salir: AAA555

Ingrese la hora de salida (HH:MM): 5:11

Ingrese la fecha de salida (DD/MM): 10/07

Su vehiculo estuvo 2.48 horas en el parqueadero

Debe pagar: 6200.0\$






# CONCLUSIONES

Este proyecto nos permitió construir un prototipo básico de sistema de parqueadero que opera según los requerimientos iniciales. Pudimos registrar vehículos, calcular cobros y generar reportes, demostrando la viabilidad de la solución para escenarios como la capacitación del personal de Parking Center S.A.S.

A lo largo del desarrollo, aplicamos y fortalecimos conocimientos en programación básica, el uso efectivo de estructuras de datos como listas y tuplas, y la implementación de lógica algorítmica.





**¡Muchas  
gracias  
por su  
atención!**

**Juan David Cedeño Arenas  
Alejandro Ducuara Satizabal  
Isaac Abdiel Contreras Lovich  
Salomé Moreno Sánchez**

**Grupo #4**