Manual de Usuario

Algoritmos y Programación

Medellín, 13 de Julio de 2025

1. Introducción

La Universidad de Antioquia, como institución educativa de gran escala, cuenta con una alta afluencia diaria de estudiantes, docentes, empleados administrativos y visitantes, muchos de los cuales hacen uso del parqueadero institucional.Este sistema está pensado para ser utilizado por el personal responsable del control vehicular dentro de los parqueaderos de la Universidad de Antioquia, facilitando sus labores diarias, minimizando errores y mejorando la eficiencia operativa.

2. Objetivos:

2.1 Objetivo General

El propósito del sistema es automatizar la gestión de parqueaderos, permitiendo registrar la entrada y salida de vehículos, calcular el tiempo de uso, generar cobros automáticos y mantener registros históricos.

2.2 Objetivos específicos

- Diseñar una interfaz de menú interactiva para el personal de administración del parqueadero universitario.
- Implementar el cálculo automático del tiempo de permanencia de cada vehículo y el cobro correspondiente según una tarifa establecida por minuto.
- Facilitar la consulta del estado del parqueadero y la generación de reportes para control interno de la Universidad de Antioquia.

3. Beneficios:

El sistema de gestión de parqueaderos para la Universidad de Antioquia ofrece una serie de beneficios clave para mejorar la eficiencia del servicio. En primer lugar, permite una optimización del tiempo de administración, automatizando tareas que antes requerían intervención manual. Además, contribuye a la reducción de errores humanos en los cálculos de cobro, al emplear funciones precisas que calculan el tiempo de uso del parqueadero. También proporciona un registro histórico del ingreso y salida de vehículos, lo cual facilita el seguimiento, control y análisis de la operación diaria.

4. Especificación de requisitos:

Para garantizar el correcto funcionamiento del sistema, se requiere el cumplimiento de los siguientes aspectos técnicos:

- **Lenguaje de programación:** Python 3.8 o superior.

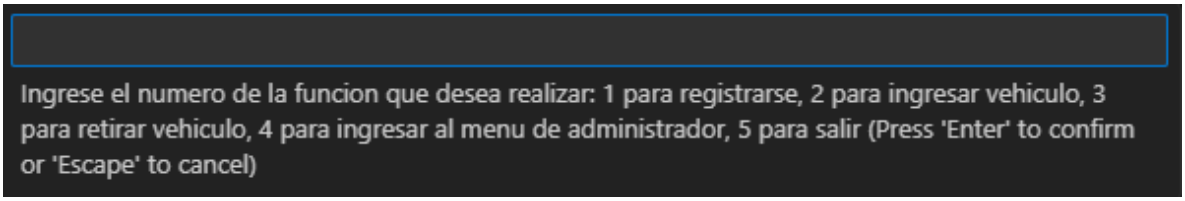
- **Entorno de desarrollo:** Jupyter Notebook, por su enfoque interactivo y facilidad de uso.

- **Librerías utilizadas:** Biblioteca `datetime` (incluida de forma nativa en Python) para el manejo de fechas y horas.

- **Condiciones de ejecución:** El sistema está diseñado para operar de manera local, sin necesidad de conexión a Internet, lo cual resulta ideal para ambientes con conectividad limitada dentro del campus universitario.

5. Manual de usuario

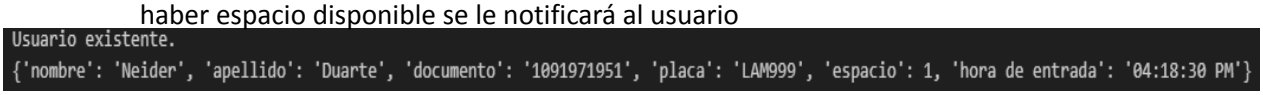
5.1. Inicio del Sistema



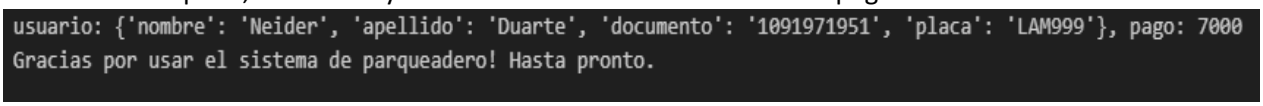
Indica la función que deseas realizar en el sistema.

5.2. Opciones del Menú Principal

- **Registro:** El usuario deberá ingresar sus datos personales(Nombre, documento y placa). Se guardarán en el sistema para su posterior ingreso de vehículo.
- **Ingreso de vehículo:** Deberá ingresar el documento del usuario y placa del vehículo. Luego el sistema marcará su hora de ingreso y espacio ocupado. De no haber espacio disponible se le notificará al usuario



- **Retirar vehículo:** Ingresa el documento y placas. Si el vehículo tiene un espacio ocupado, se retirará y se le mostrará al usuario el monto a pagar.



5.3 Salir del sistema

Finaliza la visita del usuario.

6. Opciones del sistema

6.1.Registro del usuario:

El sistema va a pedir para registrar el usuario su nombre, apellido, documento y el número de placa del vehículo. Al final de esto todos los usuarios registrados se guardan en un diccionario.

```python
def registrar_usuario(nombre, apellido, documento, placa, usuarios_registrados):
    errores = False

    # Nombre
    if len(nombre) < 3:
        print('Error, el nombre debe tener minimo 3 letras')
        errores = True
    if not nombre.isalpha():
        print('Error, solo debe contener letras')
        errores = True

    # Apellido
    if len(apellido) < 3:
        print('Error, el apellido debe tener minimo 3 letras')
        errores = True
    if not apellido.isalpha():
        print('Error, solo debe contener letras')
        errores = True

    # Documento
    documento = str(documento)
    if not documento.isdigit():
        print('Error, el documento debe contener solo numeros')
        errores = True
    if not 3 <= len(documento) <= 15:
        print('Error, el documento debe tener minimo 3 numeros y maximo 15')
        errores = True
```

```
# Placa
if not len(placa) == 6:
    print('Error, la placa debe contener exactamente 6 caracteres')
    errores = True
if not (placa[:3].isalpha() and placa[3:].isdigit()):
    print('Error, la placa debe contener 3 letras y 3 numeros respectivamente')
    errores = True

if errores:
    return None

#Registro de usuario
else:
    usuario = {"nombre": nombre.capitalize(),
               "apellido": apellido.capitalize(),
               "documento": documento,
               "placa": placa.upper()}
    usuarios_registrados.append(usuario)

with open(ruta, "a") as archivo:
    for dic in usuarios_registrados:
        archivo.write(str(dic) + "\n")

return "Usuario registrado exitosamente!", usuario
```

6.2. Registro de entrada de vehículos:

El sistema permite registrar la placa del vehículo junto con la hora exacta de entrada utilizando datetime.now(). Esta información se guarda en un diccionario llamado vehículos.

```
def ingresar_vehiculo(documento, usuarios_registrados):

    # Verificar usuario
    usuario_encontrado = None
    for u in usuarios_registrados:
        if u["documento"] == documento:
            usuario_encontrado = u
            print('Usuario existente.')
            break
    if not usuario_encontrado:
        return "Error: Usuario no registrado."

    #Asignar puesto
    if usuario_encontrado:
        if espacios_disponibles:
            espacio_asignado = espacios_disponibles.pop(0)
            usuario_encontrado["espacio"] = espacio_asignado
            hora_entrada = datetime.now().strftime("%I:%M:%S %p")
            usuario_encontrado["hora de entrada"] = hora_entrada
```

```
        return usuario_encontrado
    else:
        return "No hay espacios disponibles."
```

6.3. Registro de salida de vehículos:

El usuario ingresa la placa del vehículo al salir. El sistema busca la hora de entrada y calcula el tiempo transcurrido.

```
def retirar_vehiculo(documento, placa, usuarios_registrados, espacios_disponibles, tiempos_estancia):

    usuario_encontrado = None
    for u in usuarios_registrados:
        if u["documento"] == documento:
            usuario_encontrado = u
            break
    if not usuario_encontrado:
        return "Error: Usuario no registrado."

    if usuario_encontrado["placa"] != placa:
        return "Error: La placa no coincide con el documento."

    if "espacio" not in usuario_encontrado:
        return "Error: El vehículo no tiene un espacio asignado actualmente."

    else:
        espacio_liberado = usuario_encontrado["espacio"]
        espacios_disponibles.append(espacio_liberado)
        del usuario_encontrado["espacio"]
```

```
formato = "%I:%M:%S %p"
entrada = datetime.strptime(usuario_encontrado["hora de entrada"], formato).time()
fecha_actual = datetime.now().date()
entrada = datetime.combine(fecha_actual, entrada)
salida = datetime.now()
tiempo_transcurrido = salida - entrada
tiempos_estancia.append({'tiempo_transcurrido': tiempo_transcurrido, 'Vehiculo': usuario_encontrado["placa"]})
del usuario_encontrado['hora de entrada']
```

6.4. Cálculo del tiempo y cobro:

El sistema convierte el tiempo total en horas y calcula el cobro total con una tarifa fija por segundos.

```python
valor_hora = 7000
valor_cuartos = 500
total_segundos = tiempo_transcurrido.total_seconds()
horas = int(total_segundos // 3600)
segundos_restantes = total_segundos - (horas * 3600)
cuartos = int(segundos_restantes // 900)
total = (horas * valor_hora) + (cuartos * valor_cuartos)

if total < valor_hora:
    total = valor_hora
```

6.5. Menú Admin

Es donde se encuentran todos los datos registrados del parqueadero, hora de entrada y salida, nombre y apellido de la persona, su numero de documento y placa del vehículo. También se puede ver el número de celdas libres, vehículos retirados y sin retirar, y el pago total del parqueadero.

```python
def menu_admin(usuarios_registrados, vehiculos_registrados, vehiculos_retirados, ganancias):
    vehiculos_sin_retirar = 0
    total_espacios = 50

    for usuario in usuarios_registrados:
        if "espacio" in usuario:
            vehiculos_sin_retirar += 1

    celdas_libres = total_espacios - vehiculos_sin_retirar

    # Calcular promedio de tiempo
    if not tiempos_estancia:
        print("\nNo hay vehículos retirados aún.")
        tiempo_promedio = "No hay datos"
        usuario_max = {"nombre": "N/A", "apellido": "", "placa": ""}
        usuario_min = {"nombre": "N/A", "apellido": "", "placa": ""}
    else:
        suma = sum([t['tiempo_transcurrido'].total_seconds() for t in tiempos_estancia])
        promedio_segundos = suma / len(tiempos_estancia)
        tiempo_promedio = f"{int(promedio_segundos // 3600)}h {int((promedio_segundos % 3600) // 60)}min"
        menor = min(tiempos_estancia, key=lambda t: t['tiempo_transcurrido'])
        mayor = max(tiempos_estancia, key=lambda t: t['tiempo_transcurrido'])
        usuario_min = menor['Vehiculo']
        usuario_max = mayor['Vehiculo']

    print("\n RESUMEN DEL PARQUEADERO \n")
    print(f"Total de vehículos registrados: {vehiculos_registrados}")
    print(f"Total de vehículos retirados: {vehiculos_retirados}")
    print(f"Total de vehículos sin retirar: {vehiculos_sin_retirar}")
    print(f"Total pago de vehículos retirados: ${ganancias}")
```

```python
print(f"Tiempo promedio de estancia por vehículo: {tiempo_promedio}\n")

print("Lista de usuarios registrados:")
for usuario in usuarios_registrados:
    print(f"{usuario['nombre']} {usuario['apellido']} Placa: {usuario['placa']}, Doc: {usuario['documento']}")

print("\nVehículo con mayor tiempo de parqueo:")
print(usuario_max)

print("Vehículo con menor tiempo de parqueo:")
print(usuario_min)

print(f"\nCeldas libres actualmente: {celdas_libres}")
```