**Informe**

a. Descripción General del Software

El proyecto consiste en el desarrollo de un software de consola en Python para gestionar el parqueadero Ágora, ubicado en el sector de la Universidad de Antioquia.  
El objetivo principal es automatizar el registro de usuarios, control de ingreso y salida de vehículos, generación de cobros, facturas virtuales, reportes administrativos, y exportación de datos al finalizar la jornada.

Beneficios:

* Organización eficiente de los espacios de parqueo.
* Reducción de errores manuales en el cobro y registro de usuarios.
* Mejora en la atención al cliente.

## b. Especificación de Requisitos

### Requisitos Funcionales:

* Registrar nuevos usuarios con validaciones estrictas de datos.
* Ingresar vehículos solo de usuarios registrados y asignar un espacio disponible.
* Retirar vehículos, calcular tiempo de permanencia y el cobro correspondiente.
* Acceso restringido para administradores con usuario y contraseña.
* Generación de reportes administrativos detallados.
* Exportar la información en formato CSV al cierre del sistema.

### Requisitos No Funcionales:

* **Seguridad:** Validar acceso al módulo de administración mediante contraseña segura.
* **Usabilidad:** El menú debe ser claro, intuitivo y fácil de navegar.
* **Fiabilidad:** El sistema debe ser capaz de operar durante toda la jornada sin fallos.
* **Compatibilidad:** Funcionar en cualquier sistema operativo que soporte Python 3.10 o superior.

## c. Librerías

Librerías necesarias:

* datetime: Para manejar fechas y horas de entrada/salida de vehículos, permite capturar y calcular diferencias de tiempo.
* csv: Para exportar los datos finales en formato CSV, permite crear y escribir archivos CSV fácilmente.
* re (expresiones regulares): Para validar el formato de placa de los vehículos.
* statistics: Para calcular tiempos promedios de estancia.

## d. Visual

## 

e. Algoritmos

(Pseudocódigo)

Registrar Usuario:

Solicitar nombre, apellido, documento, placa

Validar cada campo

Si todo es válido:

Registrar usuario en lista

Sino:

Mostrar errores

Ingresar Vehículo:

Solicitar documento del usuario

Buscar usuario en lista

Si existe y hay espacio disponible:

Asignar celda

Registrar hora de entrada

Generar recibo virtual

Sino:

Mostrar error

Retirar Vehículo:

Solicitar documento del usuario

Buscar vehículo en registro

Calcular tiempo de permanencia

Calcular valor a pagar

Liberar celda

Mostrar factura

Acceso Administrador:

Solicitar usuario y contraseña

Si son correctos:

Mostrar menú administrador

Sino:

Denegar acceso

## F. Estructuras de datos

### **Lista**

#### **Justificación:**

Las listas son una excelente opción para **almacenar múltiples elementos ordenados**. En este caso, la lista es útil para manejar **usuarios** y **vehículos**, ya que se puede agregar, eliminar y recorrer los elementos fácilmente.

### **Diccionario**

#### **Justificación:**

Un **diccionario** es ideal para almacenar información **clave-valor**, y en el sistema, se necesitará buscar rápidamente datos de un **usuario o vehículo** basado en una **clave única** (por ejemplo, el **documento del usuario** o la **placa del vehículo**).

### **Tupla**

#### **Justificación:**

Las **tuplas** son **inmutables**, lo que significa que una vez que se crean, no se pueden modificar. Esto es útil para almacenar datos que **no deben cambiar**. En este caso, se usarían para almacenar información **fija** que no necesite modificación, como el **costo por hora** o la **estructura de los espacios de parqueo**.

g. Presentación de Resultados

--- REPORTE ADMINISTRATIVO ---

Total de vehículos registrados: x

Total de vehículos retirados: x

Total de vehículos actualmente parqueados: x

Total recaudado: $x

Tiempo promedio de estancia: x hora x minutos

Vehículo con estancia máxima: ABC123 (3 horas)

Vehículo con estancia mínima: XYZ789 (30 minutos)

Celdas ocupadas: x / 50