Plan de desarrollo versionado

Versión 1.0

Objetivo: Entregar un sistema completo y funcional para la gestión de un parqueadero.

Componentes Principales:

1. Diseño de interfaz básico y registro de usuario

Objetivo: Crear la estructura básica del sistema con un menú funcional.

- Importar todas las librerías necesarias para la ejecución del código.
- Configuración de zona horaria con pytz para una mayor precisión.
- Función “fecha_hora” para obtener fecha y hora local.
- Función “imprimir_menu” con diseño centrado y formato constante.
- Sistema de logging básico con registrar_log().

Dificultades encontradas:

- Solución del desfase horario con pytz.timezone(“America/Bogota”), ya que el datetime da la hora con cinco horas de diferencia.
- Formateo correcto de fechas y horas con zfill(2)

2. Registros de usuarios y Validaciones

Objetivo: Implementar el registro de los usuarios con sus respectivas validaciones

- Función `registro_usuario` para almacenar los datos en el diccionario.
- Función `validar_datos` con verificaciones para:
  1. Documento (longitud, solo números).
  2. Nombre/Apellido (longitud, solo letras).
  3. Placa (formato AAA999).
  4. Fecha de nacimiento (formato YYYY-MM-DD, valores válidos.

Dificultades encontradas:

- La validación de fecha de nacimiento fue lenta debido a la gran cantidad de valores para evaluar.
- Manejo de múltiples mensajes de error concatenados.
- Problemas con duplicación de claves en diccionario.

3. Exportación a CSV

Objetivo: Implementar persistencia de datos mediante archivos CSV

- Función `exportar_datos_csv()` que genera:
  1. Archivo usuarios.csv con datos personales
  2. Archivo vehiculos.csv con registros de parqueo
- Formateo correcto de fechas/horas en CSV
- Manejo de encoding UTF-8 para que se muestren y procesen correctamente todos los datos en diferentes entornos.

Dificultades encontradas:

- Aprendizaje del módulo CSV de Python
- Formateo correcto de datetime a strings, ya que algunos datos no pueden ser guardados directamente en CSV(este solo guarda texto).
- Estructuración de datos anidados para exportación, en caso de variables como vehículos retirados que contienen en si otros diccionarios con fechas.
- Problemas con la serialización de objetos datetime, ya que CSV no guarda todos los datos directamente.

4. Gestión de vehículos

Objetivo: Implementar ingreso y retiro de vehículos

- Diccionario `vehiculos_parqueados` y `vehiculos_retirados` para seguimiento.
- Función `ingresar_vehiculo` con:
  1. Verificación de capacidad máxima (64 vehículos).
  2. Comprobación de usuario registrado.
  3. Generación de recibo con datos del usuario y hora de ingreso.

- Función `retirar_vehiculo` con:
  1. Cálculo de tiempo estacionado.
  2. Sistema de cobro por horas (7000 COP) y cuartos de hora (1500 COP).
  3. Descuento por cumpleaños (10%).
  4. Generación de factura detallada con datos de usuario, y detalles de estancia en el parqueadero (tiempo de permanencia, descuentos en caso de ser favorecido y monto total a pagar).

Dificultades encontradas:

- Cálculo preciso del tiempo de permanencia en el parqueadero y redondeo de los cuartos de hora.
- Validación de fecha de cumpleaños para descuento.

5. Modulo para administrador

Objetivo: Crear panel de control para administradores

- Sistema de autenticación con usuarios/contraseñas predefinidas
- 10 opciones de administración:
  1. Conteo de vehículos registrados tanto los parqueados como los registrados.
  2. Vehículos retirados (guardados en el historial).
  3. Vehículos sin retirar (vehículos que están actualmente).
  4. Total, de pagos recibidos (teniendo en cuenta los descuentos).
  5. Tiempo promedio de estancia.
  6. Listado completo de usuarios con todos los detalles.
  7. Vehículos con tiempos extremos (Max/min).
  8. Eliminación de usuarios con confirmación y depuración de datos asociados al mismo.
  9. Exportación a CSV(listado de usuarios con datos y listado de vehículos con registro de parqueo).
  10. Salir del módulo.

Dificultades encontradas:

- Cálculos estadísticos complejos usando timedelta.
- Manejo de casos bordes (como la división por cero en el caso estadístico, fecha de nacimiento invalida, datos como placa o documento inexistentes,

entre otros) que de no ser corregidos pueden generar errores en la ejecución del sistema.
- Eliminación correcta del usuario con su vehículo retirado.

Mejoras Finales y Depuración (sobre la versión 1.0)

Objetivo: Pulir detalles y corregir errores

- Mejoras implementadas:
  1. Animaciones de "Guardando datos" con time.sleep().
  2. Reubicación de todos los imports en la parte superior para tener una mayor practicidad y evitar errores.
  3. Funciones como imprimir_escena(), ingresar_vehiculo(), retirar_vehiculo() y modulo_administrador() también fueron reubicadas para evitar su redefinición y mejorar el código.
  4. Se busco una consistencia en todos los cálculos, especialmente en la parte del calculo de cuartos de hora, ya que inicialmente se tenían formas diferentes para calcular esta parte.
  5. Validación adicional de datos en todos los módulos.
  6. Corrección de bugs en cálculos de tiempo y cobros.
  7. Optimizar el uso de la función limpiar_consola() para mejorar la experiencia del usuario, ya que como se planteo inicialmente no cumplía la funcionalidad que se esperaba.
  8. Mejoras en la presentación visual con centrado y espaciado.
  9. Optimización del programa principal.

Lecciones aprendidas

- Manejo de tiempo: pytz fue esencial para la configuración horaria.
- Validaciones: Las fechas requieren verificaciones exhaustivas.
- Estructura de datos: Los diccionarios anidados son poderosos ya que simplifican mucho el manejo de datos, pero requieren cuidado.
- Persistencia: CSV es útil, pero tiene limitaciones con tipos complejos como(datetime).
- Interfaz: Pequeños detalles visuales mejoran mucho la experiencia de usuario (como un espaciado constante el cual mejora la legibilidad).
- Manejo de errores: La validación anticipada evita muchos problemas posteriores.

Estado actual del proyecto

El proyecto se encuentra funcional, con las siguientes características completas:

- Registro de usuario con validaciones sólidas.
- Control de zona horaria para Colombia correctamente configurado.
- Cobros de permanencia y descuentos correctamente configurados para su ejecución.
- Menús interactivos para usuarios y administrador.
- Exportación e importación de datos en formato CSV.