Versificación del sistema

Introducción sobre la versificación del código Para dar indicaciones sobre el plan versionado cabe resaltar cierta información sobre el punto 3 del trabajo. Este trabajo se centra en dar indicaciones sobre el desarrollo del código y como su avance se va subiendo al GitHub, nuestro grupo no siguió del todo estas indicaciones pues el real izamiento del código se realizó de manera consecutiva, que se quiere decir con esto, que el código se realizó en un lapso de tiempo rápido y no se centró en subir sus diferentes versiones al GitHub si no que se centró en hacerlo de escalonada si no que se centró en hacerlo de manera rápida. Por consiguiente, a esta explicación se dará una descripción sobre los diferentes módulos en su fase final no en sus versiones de prueba.

Descripción del módulo del administrador

Esta parte del trabajo abarca el desarrollo sobre la implementación del módulo del administrador del sistema del parqueadero. Esta etapa del trabajo se basa en un código el cual se centra en permitir el ingreso exclusivo de usuarios los cuales ejercen el cargo de administradores por medio de una autenticación con credenciales guardadas en el archivo admin.json. Tras la autenticación el administrador podrá acceder al sistema pudiendo tener a la mano reportes que incluyen el total de usuarios registrados, así como cuantos automóviles ingresan y salen del parqueadero, además de cuanto se recauda enfocado en el parqueadero, así como un tiempo de permanencia promedio, calculando de esta manera el máximo y el mínimo de minutos. La lógica de implementación se centra en emplear archivos planos, además poder contralar datos como las fechas por medio de datetime, logrando así poder integrar un análisis funcional del parqueadero.

```
import json
import csv
import os
from datetime import datetime
from utilidades import RUTA_USUARIOS, RUTA_INGRESOS, RUTA_RETIROS

RUTA_ADMIN = os.path.join("src", "datos", "admin.json")
```

```
def menu_administrador():
    print("\n--- ACCESO ADMINISTRADOR ---")
    usuario = input("Usuario: ").strip()
    clave = input("Contraseña: ").strip()
```

```
    if not autenticar(usuario, clave):
        print("❌Acceso denegado.")
        return
```

```
    print("✅Acceso concedido.")
    mostrar_reportes()
```

```
def autenticar(usuario, clave):
    try:
        with open(RUTA_ADMIN, "r") as file:
            data = json.load(file)
            return data.get(usuario) == clave
    except FileNotFoundError:
        return False
```

```
def mostrar_reportes():
    print("\n--- REPORTES ADMIN ---")
```

```
    usuarios = sum(1 for _ in open(RUTA_USUARIOS)) if os.path.exists(RUTA_USUARIOS) else 0
    ingresados = sum(1 for _ in open(RUTA_INGRESOS)) if os.path.exists(RUTA_INGRESOS) else 0
    retirados = sum(1 for _ in open(RUTA_RETIROS)) if os.path.exists(RUTA_RETIROS) else 0
```

```
    pagos = []
    tiempos = []
```

```
    try:
        with open(RUTA_RETIROS, mode="r") as archivo:
            lector = csv.reader(archivo)
            for fila in lector:
                pagos.append(int(fila[6]))
                inicio = datetime.strptime(fila[2], "%Y-%m-%d %H:%M:%S")
                fin = datetime.strptime(fila[3], "%Y-%m-%d %H:%M:%S")
                tiempos.append((fin - inicio).total_seconds() / 60)
    except:
        pass
```

```
    print(f"👤 Usuarios registrados: {usuarios}")
    print(f"🚗 Vehículos en parqueadero: {ingresados}")
    print(f"🚙 Vehículos retirados: {retirados}")
    print(f"💰 Total recaudado: ${sum(pagos):,.0f}")
    if tiempos:
        print(f"⏱ Tiempo promedio de estadía: {sum(tiempos)/len(tiempos):.1f} minutos")
        print(f"📈 Máximo: {max(tiempos):.1f} min, Mínimo: {min(tiempos):.1f} min")
```

Descripción del módulo del menú principal

El código presente representa el módulo principal del proyecto de parqueadero "Smart Parking-Universidad de Antioquia". La finalidad del código radica en ser un conector sobre la interfaz con la capacidad de acceder a todas las funciones que el programa tiene. Dentro de la información que tiene el módulo se caracteriza la función que cumple al importar submenús para un manejo sobre los usuarios, como lo son el (menú administrador), (menu_usuarios), (menú vehículos) y la exportación de datos (exportar datos) en archivos csv. La introducción del código se muestra el logo que representa el proyecto del parqueadero, además de un menú grafico el cual indica a los nuevos usuarios una guía con cinco opciones las cuales son registrar usuario, ingresar vehículo, retirar vehículo, modo administrador, y por último salir, cada una de estas opciones al ser abiertas representaran un módulo en específico, dependiendo de la opción que se elija esta redirige un módulo nuevo.

El código plasma un menú de entrada amigable para los usuarios que facilita su uso, además de que estructura la información de manera sencilla esto con la finalidad de facilitar el uso del software para los operarios y administradores del parqueadero. La base del trabajo resaltando desde el registro de usuarios nuevos que ingresan al parqueadero, hasta un proceso de concepto de cobranza, registrando por un intervalo de tiempo la actividad que ha tenido el parqueadero en su labor. En conclusión, este menú tiene la funcionalidad de actuar como un núcleo de navegación sobre el sistema para nuevos usuarios, conformando todos los procesos que se presentan en el parqueadero logrando así tener un mejor manejo y una mejor facilidad hacia la automatización y gestionamiento de las operaciones que se presentan en el parqueadero.

```
from usuario import menu_usuarios
from vehiculo import menu_vehiculos
from administrador import menu_administrador
from utilidades import exportar_datos

def menu_principal():
    def print_logo():
        print("""
┌──────────────────────────┐
│  ▄▄▄▄ ▄  ▄ ▄▄ ▄▄▄▄ ▄    │
│  ▄    ▄▄ ▄▄ ▄ ▄ ▄ ▄ ▄    │
│  ▄▄▄▄ ▄ ▄ ▄ ▄▄▄▄ ▄▄▄▄ ▄  │
│     ▄ ▄  ▄ ▄ ▄ ▄ ▄  ▄   │
│  ▄▄▄▄ ▄  ▄ ▄ ▄ ▄ ▄  ▄   │
│                          │
│ SMART PARKING - Universidad │
│        de Antioquia      │
└──────────────────────────┘
""")
```

```
    while True:
        print("\n")
        print_logo()
        print("┌──────────────────────────────┐")
        print("│         MENÚ PRINCIPAL       │")
        print("├──────────────────────────────┤")
        print("│ 1. Registrar usuario         │")
        print("│ 2. Ingresar vehículo         │")
        print("│ 3. Retirar vehículo          │")
        print("│ 4. Modo administrador        │")
        print("│ 5. Salir                     │")
        print("└──────────────────────────────┘")
        opcion = input("Ingrese una opción: ")
```

```
        if opcion == "1":
            menu_usuarios()
        elif opcion == "2":
            menu_vehiculos("ingreso")
        elif opcion == "3":
            menu_vehiculos("retiro")
        elif opcion == "4":
            menu_administrador()
        elif opcion == "5":
            print("\n┌──────────────────────────────┐")
            print("│      >> EXPORTAR DATOS <<    │")
            print("└──────────────────────────────┘")
            confirmar = input("¿Generar reporte CSV? (S/N): ").lower()
            if confirmar == "s":
                exportar_datos()
                print("✓ Archivo 'reporte_smartparking_DD-MM-YYYY.csv' creado |\n¡Hasta
pronto!")
            else:
                print("¡Hasta pronto!")
            break
        else:
            print("Opción no válida.")
```

```
if __name__ == "__main__":
    menu_principal()
```

Descripción registro de usuarios

Como ya se ha hecho con antelación con los anteriores módulos se dará paso a describir esta función del software de manera muy general con la finalidad de explicar cada parte que compone el código en el manual de usuario. La finalidad de este módulo radica en el gestionamiento de registro de nuevos usuarios que entran al parqueadero. El módulo abarca bastante información que se centra en funciones de validación de datos (nombre, apellido, documento del usuario, y la placa del automóvil), esto se conecta con un archivo csv "usuarios.csv" donde se gurda toda la información ya recopilada por los usuarios ubicada en la carpeta src/datos.

El proceso de registro de usuarios da inicio cuando se efectúa la función menu_usuarios(), la cual cumple con la función de llamar a registro_usuario(). Por medio de esta carpeta se abarca toda la información. En casos de que surjan errores en las diferentes validaciones de datos como nombres, números, placas con errores de caracteres o documento no numéricos, el sistema cumple con la función de hacer una devolución sobre los errores que se han encontrado en la validación de información del usuario para que este pueda validar nuevamente sus datos.

Este módulo es fundamental dentro de los estándares que se plantean en el sistema, ya que permite efectuar un control sobre los usuarios que pueden acceder al sistema del parqueadero para prestar sus diferentes servicios. Mostrándose así este módulo como una base de seguridad en cuanto a la

integridad y buen manejo de los datos, permitiendo evitar errores en eventos posteriores como ingreso, concepto de cobro o la generación de estadisticas y de reportes para el administrador.

```
import csv
import re
import os

RUTA_USUARIOS = os.path.join("src", "datos", "usuarios.csv")
```

```
def validar_nombre(nombre):
    return nombre.isalpha() and len(nombre) >= 3
```

```
def validar_documento(documento):
    return documento.isdigit() and 3 <= len(documento) <= 15
```

```
def validar_placa(placa):
    return bool(re.match(r"^[A-Z]{3}[0-9]{3}$", placa.upper()))
```

```
def registrar_usuario():
    print("\n--- REGISTRO DE USUARIO ---")
```

```
    nombre = input("Ingrese el nombre: ").strip()
    apellido = input("Ingrese el apellido: ").strip()
    documento = input("Ingrese el número de documento: ").strip()
    placa = input("Ingrese la placa del vehículo (ej. ABC123): ").strip().upper()
```

```
    errores = []
```

```
    if not validar_nombre(nombre):
        errores.append(" El nombre debe tener al menos 3 letras y no contener números.")
```

```
    if not validar_nombre(apellido):
        errores.append(" El apellido debe tener al menos 3 letras y no contener números.")
```

```
    if not validar_documento(documento):
        errores.append(" El documento debe contener entre 3 y 15 dígitos numéricos.")
```

```
    if not validar_placa(placa):
        errores.append(" La placa debe tener el formato correcto: 3 letras seguidas de 3
números.")
```

```
    if errores:
        print("\n".join(errores))
        return
```

```
    with open(RUTA_USUARIOS, mode="a", newline="", encoding="utf-8") as archivo:
        escritor = csv.writer(archivo)
        escritor.writerow([documento, nombre, apellido, placa])
        print(f" Usuario registrado exitosamente.")
```

```
def menu_usuarios():
    registrar_usuario()
```

Módulo operaciones centrales del parqueadero

Este módulo es el motor del sistema ya que abarca el gestionamiento del proyecto del parqueadero, su finalidad se centra en ocuparse de las operaciones básicas operativas que permiten el buen funcionamiento y buena resolución al momento de operar. Gestiona de forma óptima las validaciones y el registro de usuarios a través de la función buscar_usuario(), que se asesora del archivo usuario.csv con la medites de validar y verificar si un conductor está autorizado de emplear el servicio

del parqueadero. Se centra en el manejo y control sobre la ocupación del parqueadero por medio de obtener_celda(), que lee el archivo ingresos.csv con el propósito de identificar espacios libres sobre las 50 celdas disponibles, empleando diversos conjuntos que radican en la optimización de un buen modelo garantizando la búsqueda de que no se repitan espacios a una celda asignada previamente a una plaza asignada. El código también se centra en el manejo de las tarifas a través de calcular_tiempo_y_costo (), el cual se encarga de decidir el costo total, teniendo consideración sobre el tiempo de uso del parqueadero a través de horas completas y fracciones de cuarto de hora, fijando un precio de 7000 pesos colombianos, utilizando la librería datetime para resolver cálculos sobre el tiempo de estacionamiento sobre el parqueadero. El software del parqueadero por medio de archivos CSV(usarios.csv, ingresos.csv, retiros.csv), funcionan como una base de datos haciendo registros sobre la información de carácter relevante. Además de incluir la función exportar_datos() capacitado en la expiación de sobre los reportes enfocados en el parqueadero. En conclusión, este módulo del sistema del parqueadero se interconecta con los múltiples componentes del sistema, funcionando como una interfaz que integra usuarios de ingreso, retiros de vehículos, y el sistema del administrador, generando así un sistema integro y estructurado sobre la gestión y el buen manejo sobre el parqueadero universitario.

```
import os
import csv
from datetime import datetime


RUTA_USUARIOS = os.path.join("src", "datos", "usuarios.csv")
RUTA_INGRESOS = os.path.join("src", "datos", "ingresos.csv")
RUTA_RETIROS = os.path.join("src", "datos", "retiros.csv")
MAX_CELDAS = 50
```

```
def buscar_usuario(documento):
    try:
        with open(RUTA_USUARIOS, mode="r") as archivo:
            lector = csv.reader(archivo)
         def exportar_datos():
    print("✅ Datos exportados automáticamente.")   for fila in lector:
                if fila[0] == documento:
                    return True
    except FileNotFoundError:
        pass
    return False
```

```
def obtener_celda_disponible():
    ocupadas = set()
    try:
        with open(RUTA_INGRESOS, mode="r") as archivo:
            lector = csv.reader(archivo)
            for fila in lector:
                ocupadas.add(int(fila[1]))
    except FileNotFoundError:
        pass
```

```
    for celda in range(1, MAX_CELDAS + 1):
        if celda not in ocupadas:
            return celda
    return None
```

```
def liberar_celda(celda):
    pass
```

```python
def calcular_tiempo_y_costo(inicio, fin):
    total_min = int((fin - inicio).total_seconds() // 60)
    horas = total_min // 60
    cuartos = (total_min % 60) // 15
    total = (horas * 7000) + (cuartos * 1500)
    if total < 7000:
        total = 7000
    return horas, cuartos, total
```

Módulo gestión de ingreso y retiros de vehículos

Este módulo se centra en enfatizar el gestionamiento sobre las operaciones básicas del parqueadero las cuales son funciones simples pero fundamentales relacionadas con el ingreso y retiros de automóviles. Por medio de las funciones ingresos_vahiculos y retiro_vehiculo, se centra en la validación de los usuarios para permitir el registro e ingreso del vehículo, de esta manera si la validación del usuario es cierta se procedera a asignar automáticamente una celda libre para el vehículo. Siendo muy similar al módulo anterior, pero ejemplificando la información, buscando redundar más en el proceso de gestionamiento de retiros, cálculo de tiempo sobre el parqueadero, así como de su costo correspondiente. Este módulo se centra en la interacción sobre funciones suplementarias, que son de gran importancia en su resolución, como lo son la verificación de usuarios, manejo de archivos CSV así como de las tarifas y su cálculo correspondiente, permitiendo de esta manejara asegurar una buena administración en cuanto al manejo sobre los movimiento fundamentales del parqueadero se refiere como lo es la automatización vehicular que circula en el parqueadero.

```python
import csv
import os
from datetime import datetime
from utilidades import calcular_tiempo_y_costo, buscar_usuario, obtener_celda_disponible,
liberar_celda, RUTA_INGRESOS, RUTA_RETIROS, MAX_CELDAS

def menu_vehiculos(accion):
    if accion == "ingreso":
        ingreso_vehiculo()
    elif accion == "retiro":
        retiro_vehiculo()
```

```python
def ingreso_vehiculo():
    print("\n--- INGRESO DE VEHÍCULO ---")
    documento = input("Ingrese el documento del usuario: ").strip()
```

```python
    if not buscar_usuario(documento):
        print(" El usuario no está registrado.")
        return
```

```python
    celda = obtener_celda_disponible()
    if celda is None:
        print(" No hay espacios disponibles.")
        return
```

```python
    hora_ingreso = datetime.now()
    datos = [documento, celda, hora_ingreso.strftime("%Y-%m-%d %H:%M:%S")]

    with open(RUTA_INGRESOS, mode="a", newline="") as archivo:
        escritor = csv.writer(archivo)
        escritor.writerow(datos)
```

```python
    print(f" Vehículo ingresado en la celda {celda} a las {hora_ingreso.strftime('%H:%M')}.")
```

```python
def retiro_vehiculo():
```

```
    print("\n--- RETIRO DE VEHÍCULO ---")
    documento = input("Ingrese el documento del usuario: ").strip()

    registros = []
    encontrado = None

    with open(RUTA_INGRESOS, mode="r") as archivo:
        lector = csv.reader(archivo)
        for fila in lector:
            if fila[0] == documento:
                encontrado = fila
            else:
                registros.append(fila)

    if not encontrado:
        print(" No se encontró un vehículo registrado con ese documento.")
        return

    celda = encontrado[1]
    hora_ingreso = datetime.strptime(encontrado[2], "%Y-%m-%d %H:%M:%S")
    hora_salida = datetime.now()

    horas, cuartos, total = calcular_tiempo_y_costo(hora_ingreso, hora_salida)

    with open(RUTA_RETIROS, mode="a", newline="") as archivo:
        escritor = csv.writer(archivo)
        escritor.writerow([documento, celda, hora_ingreso.strftime("%Y-%m-%d %H:%M:%S"),
                           hora_salida.strftime("%Y-%m-%d %H:%M:%S"), horas, cuartos, total])

    with open(RUTA_INGRESOS, mode="w", newline="") as archivo:
        escritor = csv.writer(archivo)
        escritor.writerows(registros)

    liberar_celda(celda)
    print(f" Vehículo retirado. Total a pagar: ${total:,.0f} COP")
```