Descripción general.

La aplicación de escritorio fue desarrollada con Python, porque permite proporcionar una vista amigable al usuario y obtener información relacionada a los recursos del sistema mediante instrucciones y procedimientos en Visual Studio Code. Python con la ayuda de Post proporciona la comunicación correcta entre la interfaz y el código funcional de Java Script, mostrando detalles como el porcentaje de uso de CPU, el espacio de disco ocupado, el espacio en disco disponible, y el espacio de disco total en tiempo real, esta comunicación en tiempo real se realiza mediante eventos, con las instrucciones Event "e" desde ctrl + s para enviar un mensaje de actualización y Alert desde la interfaz para que funcione como un Mensaje de Información o Error. La interfaz fue realizada con HTML, porque permite crear diseños muy llamativos y fáciles de implementar.

Requisitos mínimos:

Para el desarrollo de este programa e necesito con las siguiente características:

Procesador Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz

RAM instalada 16.0 GB (15.8 GB utilizable)

Tipo de sistema Sistema operativo Windows 11.

Diccionario de Métodos:

- Comentario.__init__(self, pelicula, autor, mensaje): El método constructor de la clase Comentario. Inicializa los atributos pelicula, autor y mensaje de la instancia.
- Comentario.toDict(self): Devuelve un diccionario que contiene los atributos de la instancia de Comentario.
- Pelicula.__init__(self, nombre, genero, clasificacion, anio, duracion, link): El método constructor de la clase Pelicula. Inicializa los atributos nombre, genero, clasificacion, anio, duracion y link de la instancia.
- Pelicula.toDict(self): Devuelve un diccionario que contiene los atributos de la instancia de Pelicula.
- Usuario.__init__(self, nombre, apellido, nombre_usuario, contrasenia, tipo): El método constructor de la clase Usuario. Inicializa los atributos nombre, apellido, nombre_usuario, contrasenia y tipo de la instancia.
- Usuario.toDict(self): Devuelve un diccionario que contiene los atributos de la instancia de Usuario.
- CargarPeliculas(datos): Toma un diccionario datos que contiene un texto con información de películas, y devuelve una lista de diccionarios que representan cada película.
- GetPelicula(datos, peliculas): Toma un diccionario datos que contiene el nombre de una película, y una lista de diccionarios peliculas que representan las películas cargadas. Devuelve un diccionario que contiene la información de la película correspondiente al nombre proporcionado y un código de estado HTTP (200 si se encuentra la película, 400 en caso contrario).
- EditarPelicula(datos, peliculas): Toma un diccionario datos que contiene información de una película a editar, y una lista de diccionarios peliculas que representan las películas cargadas. Devuelve un diccionario que contiene la lista actualizada de películas y un código de estado HTTP (200 si se edita la película, 400 en caso contrario).
- EliminarPelicula(datos, peliculas): Toma un diccionario datos que contiene el nombre de una película, y una lista de diccionarios peliculas que representan las películas cargadas. Devuelve un diccionario que contiene la lista actualizada de películas y un código de estado HTTP (200 si se elimina la película, 400 en caso contrario).
- GetComentarios(datos, comentarios): Toma un diccionario datos que contiene el nombre de una película, y una lista de diccionarios comentarios que representan los comentarios cargados. Devuelve un diccionario que contiene la lista de comentarios correspondientes a la película proporcionada y un código de estado HTTP (200 si se encuentran comentarios, 400 en caso contrario).

- EliminarUsuario(datos, usuarios): Toma un diccionario datos que contiene el nombre de un usuario, y una lista de diccionarios usuarios que representan los usuarios registrados. Devuelve un diccionario que contiene la lista actualizada de usuarios y un código de estado HTTP (200 si se elimina el usuario, 400 en caso contrario).
- RegistrarUsuario(datos, usuarios): Toma un diccionario datos que contiene información de un nuevo usuario a registrar, y una lista de diccionarios usuarios que representan los usuarios registrados. Devuelve un diccionario que indica si se ha registrado el usuario correctamente o no, y un código de estado HTTP (200 si se registra el usuario, 400 en caso contrario).
- RecuperarContrasenia(datos, usuarios): Toma un diccionario datos que contiene el nombre de un usuario, y una lista de diccionarios usuarios que representan los usuarios registrados. Devuelve un diccionario que contiene la contraseña del usuario correspondiente al nombre proporcionado y un código de estado HTTP (200 si se encuentra la contraseña, 400 en caso contrario).
- IniciarSesion(datos, usuarios): Toma un diccionario datos que contiene información de un usuario que quiere iniciar sesión, y una lista de diccionarios usuarios que representan los usuarios registrados. Devuelve un diccionario que contiene información del usuario que ha iniciado sesión, si existe, y un código de estado HTTP (200 si se inicia sesión, 400 en caso contrario).
- GetUsuarioEndatos(datos, usuarios): Toma un diccionario datos que contiene el nombre de un usuario, y una lista de diccionarios usuarios que representan los usuarios registrados. Devuelve un diccionario que contiene la información del usuario correspondiente al nombre proporcionado y un código de estado HTTP (200 si se encuentra el usuario, 400 en caso contrario).

Diccionario de Clases:

```python
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad


class Estudiante:
    def __init__(self, nombre, edad, carrera):
        self.nombre = nombre
        self.edad = edad
        self.carrera = carrera
```

```
class Profesor:
    def __init__(self, nombre, edad, asignatura):
        self.nombre = nombre
        self.edad = edad
        self.asignatura = asignatura

# Definimos un diccionario que contiene clases como valores
dic_clases = {
    "persona": Persona,
    "estudiante": Estudiante,
    "profesor": Profesor
}
```

Instancias:

```
persona1 = dic_clases["persona"]("Juan", 30)
estudiante1 = dic_clases["estudiante"]("María", 20, "Informática")
profesor1 = dic_clases["profesor"]("Pedro", 40, "Matemáticas")
```

Atributos:

```
print(persona1.nombre, persona1.edad)
print(estudiante1.nombre, estudiante1.edad, estudiante1.carrera)
print(profesor1.nombre, profesor1.edad, profesor1.asignatura)
```
```
```