

2022

ADMINISTRACION DE SISTEMAS OPERATIVOS POR VOZ.



Autor: Galindo García, Luis María

Tutor: Niño Camazón, Jesús M.

2/15/2022

Contenido

| | |
|--|----|
| 1. Introducción | 3 |
| 2. Objetivo | 4 |
| 3. Estado del arte/mercado | 5 |
| Cortana (Microsoft) | 5 |
| Siri (IOS) | 6 |
| Alexa (Amazon) | 7 |
| 4. Diseño de la solución | 8 |
| Que voy a utilizar, y cuál es su definición: | 9 |
| • Powershell | 9 |
| • Python | 9 |
| • DLLs | 9 |
| • Notificaciones | 9 |
| 5. Desarrollo | 10 |
| • Crear un bucle de escucha sin errores | 10 |
| • Definir las palabras a reconocer del reconocimiento de voz | 10 |
| • Entrenar al reconocimiento con mi voz | 11 |
| • Definir DLL de funcionamiento del ratón | 13 |
| • Crear Python para reconocimiento de objetos | 14 |
| • Establecer notificaciones emergentes | 15 |
| • Explicación código | 16 |

| | | |
|----|---|----|
| 6. | Aplicación práctica (depende del proyecto concreto) | 19 |
| 7. | Conclusiones..... | 23 |
| 8. | Líneas abiertas/futuras ampliaciones | 24 |
| 9. | Bibliografía/Webgrafía | 25 |
| | Documentación | 25 |
| | Módulos y notificaciones | 26 |
| | Código..... | 26 |

1. Introducción

Debido a la innovación que me parece los motores de voz, decidí hacer mi trabajo sobre que menos, un administrador de sistemas, pero por voz, esta herramienta se puede desarrollar en cualquier ámbito de nuestra vida cotidiana, automatización de casas, garajes, respiraderos, etc.

2. Objetivo

El objetivo de mi proyecto es una administración de un sistema operativo Windows 10, mediante la voz.

Esta administración del sistema está “dedicada” o “entornada” a los dos años de ASIR, con esto quiero definir que, al ser un tema tan amplio, en su mayoría será la administración de cosas vistas del grado, o elementos a consultar que se utilicen día a día en la vida de un administrador de sistemas.

La voz permitirá ahorrar tiempo e incluso conocimientos, pero solo siendo utilizada productivamente, ya que hay cosas que se deberán escribir.

Cosas como:

- Panel de control.
- Firewall.
- XAMPP.
- Usuarios.
- Carpetas.
- Unidades de almacenamiento.
- Adaptadores red.

Así como programas utilizados como:

- Cisco.
- Google.
- VMware.

3. Estado del arte/mercado

En la actualidad, existen en su mayoría dos motores de voz:

Cortana (Microsoft)

Cortana es el asistente de productividad personal de Microsoft que te ayuda a ahorrar tiempo y centrarte en lo más importante.

Cortana puede establecer recordatorios, reconocer voz natural sin la necesidad de ingresar el teclado y responder preguntas utilizando información del motor de búsqueda de Bing.

Microsoft está en proceso de dismantelar Cortana. En abril de 2021, Microsoft eliminó las aplicaciones de Cortana para iOS y Android,⁵ y en Windows 11 ya no está preinstalada.

Para empezar, selecciona el icono de Cortana en la barra de tareas. Estas son algunas de las cosas que Cortana puede hacer por ti:

1. Administra tu calendario y mantén tu agenda al día
2. Únete a una reunión en Microsoft Teams o averigua con quién es tu próxima reunión
3. Crear y administrar listas
4. Configura recordatorios y alarmas
5. Buscar hechos, definiciones e información
6. Abre aplicaciones en tu ordenador

Siri (IOS)

Inteligencia artificial con funciones de asistente personal a veces con su propia personalidad para iOS, macOS, tvOS y watchOS.

Esta aplicación utiliza procesamiento del lenguaje natural para responder preguntas, hacer recomendaciones y realizar acciones mediante la delegación de solicitudes hacia un conjunto de servicios web que ha ido aumentando con el tiempo.

Esta aplicación para iOS es el primer producto lanzado al público de SRI venture group, un grupo de desarrollo de software enfocado en aplicaciones de inteligencia virtual (no confundir con inteligencia artificial).

Siri fue adquirida por Apple Inc. el 28 de abril de 2010.

Puede realizar diversas tareas, dependiendo de lo que quieras. Busca la información sobre lo que le has pedido en las aplicaciones disponibles. Si no puede encontrar la información en éstas, buscará la instrucción en Internet.

Alexa (Amazon)

Asistente virtual desarrollado por Amazon, utilizado por primera vez en el altavoz inteligente Amazon Echo.

Los usuarios pueden extender las habilidades de Alexa instalando "skills" (funcionalidades adicionales desarrolladas por terceros parecidas a las aplicaciones) desde la app Alexa. También pueden crear rutinas para automatizar sus dispositivos inteligentes en función de un comando de voz, hora o ubicación.

Los dispositivos compatibles con Alexa permiten a los usuarios activar el sistema utilizando una palabra de activación (como Alexa, Echo o Amazon); otros dispositivos (como la app de Amazon Alexa y la app de Amazon Music para Android y iOS) requieren que el usuario pulse un botón para activar el modo de escucha de Alexa.

4. Diseño de la solución

El diseño se basa en crear mediante PowerShell un bucle en escucha que utilice el micrófono y dentro de este haya palabras asociadas a un script.

A su vez dentro de este bucle de escucha, habrá más scripts independientes, o no, entre ellos, asociados a palabras en concreto, que habrán sido predefinidas anteriormente.

Las palabras predefinidas no son legibles, o, mejor dicho, la capacidad de escucha del sistema operativo es poco fiable que dé en el blanco, por lo cual deberé entrenar el reconocimiento de voz de Windows.

Ya que si no hiciera esto habría pocas probabilidades de que el motor de voz sepa que digo, o que el código funcione bien.

Para el reconocimiento de objetos utilizare Python, el cual ayudara a powershell a localizar los objetivos, DLLs de Windows, las cuales me ayudaran a cargar acciones como “click derecho”, y notificaciones las cuales harán que la navegación sea más intuitiva.

Que voy a utilizar, y cuál es su definición:

- **Powershell**

Es una solución de automatización de tareas multiplataforma formada por un shell de línea de comandos, un lenguaje de scripting y un marco de administración de configuración. PowerShell funciona en Windows 10, Linux y macOS.

- **Python**

Es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.

- **DLLs**

Un archivo DLL es una biblioteca que contiene código y datos que pueden usar más de un programa al mismo tiempo. Por ejemplo, en Windows sistemas operativos, el archivo DLL Comdlg32 realiza funciones comunes relacionadas con el cuadro de diálogo.

- **Notificaciones**

Para que sea más visualmente atractivo e intuitivo ya que así se sabe que reconoce o hace el ordenador.

5. Desarrollo

- **Crear un bucle de escucha sin errores**

El código será una estructura de un bucle de escucha en el cual se recogerá en tiempo real todas las órdenes y cuando las entienda y las compare, que ejecute el código asociado.

Tuve que pensar bien la estructura al hacerlo ya que se producían errores en los que se congelaba el código o había que decir la misma palabra dos o tres veces hasta que se iniciara, esto ocurría, aunque reconociese la palabra la primera vez.

```
#Cargar los espacios de nombres System.Speech que contienen los tipos que admiten el reconocimiento de voz
[void][System.Reflection.Assembly]::LoadWithPartialName("System.Speech")

#Crear un objeto de tipo motor de síntesis de voz que permite a PowerShell escuchar voz para una gramática con restricciones
$speechRecogEngRestr = [System.Speech.Recognition.SpeechRecognitionEngine]::new()
```

```
##-----
do
{
    $orden = $speechRecogEngRestr.Recognize()
    $orden.text
}
```

- **Definir las palabras a reconocer del reconocimiento de voz**

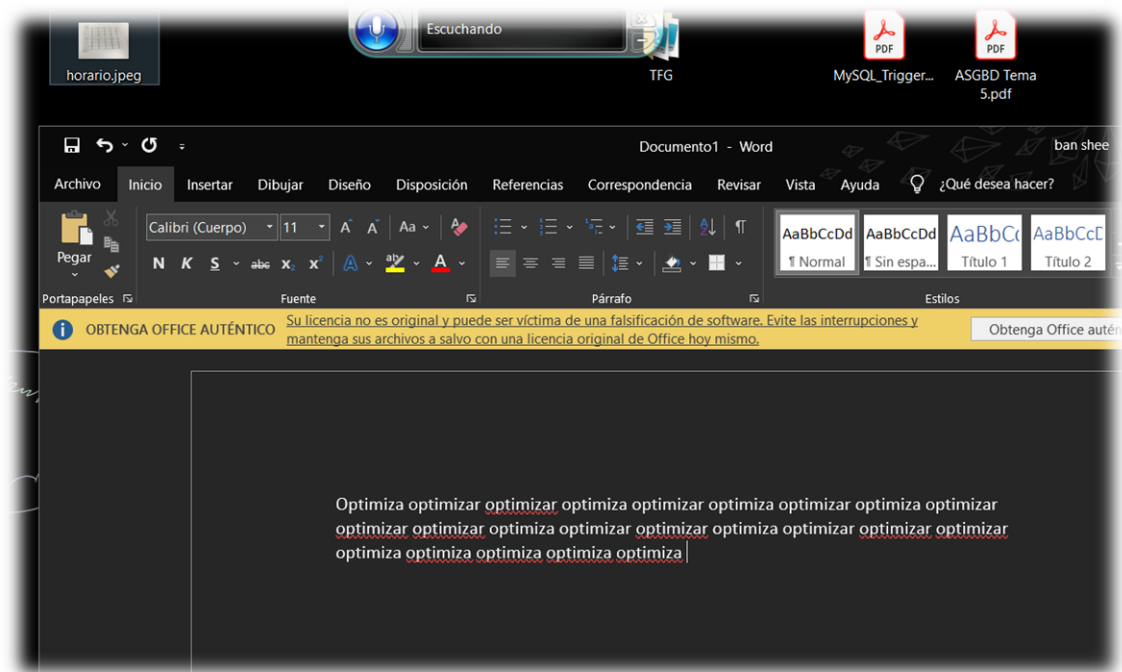
Una recopilación y definición de todas las palabras que iba a utilizar en las ordenes que quería que se ejecutase, siendo algunas de ellas compatibles con otras, como las ordenes minimizar, cerrar y expandir, las cuales se pueden utilizar en las ventanas del sistema, que no de aplicación.

```
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("notepad"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("cerrar notepad"))
#-----
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("champ"))
#-----
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("crome"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("cerrar crome"))
#-----
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("desfragmentador"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("ce"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("de"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("analizar"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("optimizar"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("cerrar desfragmentador"))
#-----
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("apaga"))
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("reinicia"))
#-----
$speechRecogEngRestr.LoadGrammar([System.Speech.Recognition.GrammarBuilder]::new("cisco"))
```

- **Entrenar al reconocimiento con mi voz**

Breve explicación: El reconocimiento de voz es la capacidad de una máquina o programa para identificar palabras y frases en lenguaje hablado y convertirlas a un formato legible por máquina.

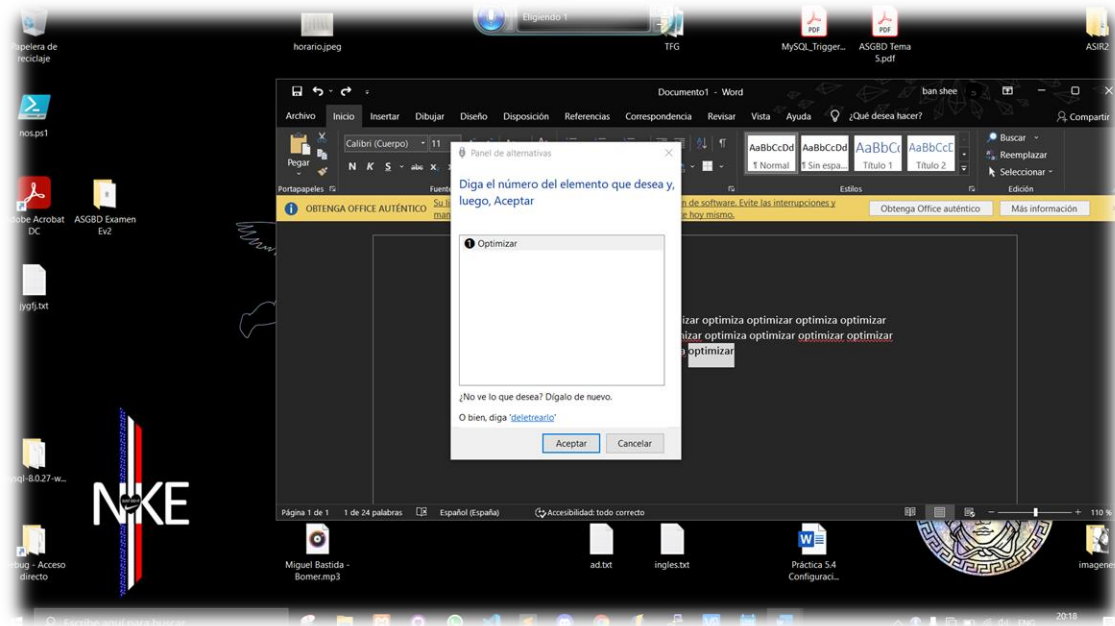
En este caso utilizo el reconocimiento y el motor de síntesis de voz en Windows.



En esta primera imagen se puede observar la palabra optimiza repetidas veces escrita, esto es una forma que pensé sobre como enseñar al reconocimiento de voz de Windows de una forma sencilla y “rápida”.

Esta forma se basa en repetir la palabra objetivo numerosas veces falle, o no.

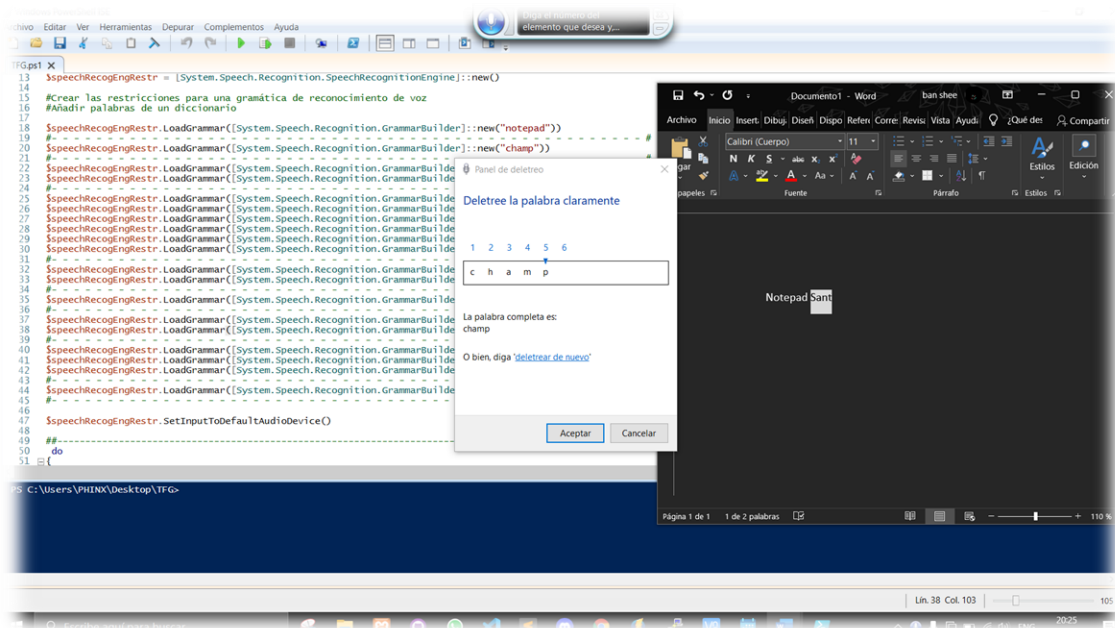
Las primeras 50 veces tenían una tasa de fallo muy grande, incluso reconociendo palabras en ningún momento dictadas, en unos 5-10 min solo dictándole esa palabra y a diferentes distancias del pc la tasa de error de la escucha era bastante insignificante.



La “gracia” de esto es que, si yo digo una palabra y reconoce otra, inmediatamente hay que corregirle, porque si sigues diciendo esa palabra y la sigue reconociendo así, el reconocimiento dictaminara que eso que escucha es esa palabra y no, cuando yo decía desfragmentar, si no escribía esa palabra, antes de volverla a decir yo decía: “Corregir”

Y automáticamente me sacaba un listado de posibles palabras que podían ser, si está en la lista con decir el numero de la opción y decir aceptar, corrige esa palabra, pero no significa que nunca más se vaya a equivocar, todo depende del entrenamiento que lleve.

ADMINISTRACION DE SISTEMAS OPERATIVOS POR VOZ



En el caso de que aun diciendo la palabra para corregirla no la entendiera ni se acercase, tenia que decir “deletrear”, y deletrear letra a letra la palabra, cosa que también hay que entrenarle las letras porque también se equivoca, y en corregir una palabra así podías llegar a tirarte tu tiempo.

- **Definir DLL de funcionamiento del ratón**

```
$MouseEventSig = @'  
[DllImport("user32.dll", CharSet=CharSet.Auto, CallingConvention=CallingConvention.StdCall)]  
public static extern void mouse_event(long dwFlags, long dx, long dy, long cButtons, long dwExtraInfo);  
'@
```

Este es un llamamiento a una DLL del ratón.

```
$startSleep = Seconds 1  
[System.Windows.Forms.Cursor]::Position = New-Object System.Drawing.Point((([Int]$posicion[0].replace("left=", "")+15), ([Int]$posicion[1].replace("top=", "")+15))  
$MouseEvent::mouse_event(0x00000002, 0, 0, 0, 0)  
$MouseEvent::mouse_event(0x00000004, 0, 0, 0, 0)
```

Y aquí se utiliza para poder clicar en los objetivos propuestos.

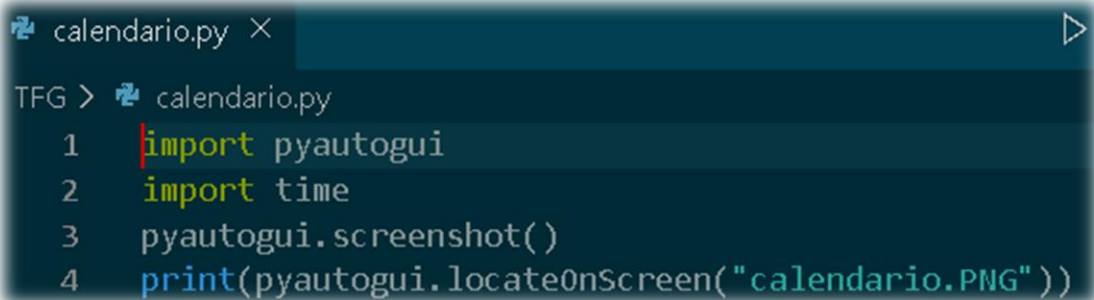
- **Crear Python para reconocimiento de objetos**

Me encontré con casos en donde solo el reconocimiento de una imagen me podría conseguir el objetivo, por lo cual junte Python con PowerShell.

Cree un Python por cada imagen, en donde localiza y determina la posición del objetivo.



| | | |
|---------------------|------------------|-------------|
| c.py | 10/02/2022 17:43 | Python File |
| calendario.py | 17/02/2022 16:47 | Python File |
| cierra.py | 17/02/2022 17:05 | Python File |
| crome.py | 10/02/2022 17:15 | Python File |
| d.py | 17/02/2022 16:40 | Python File |
| desfrag-analiza.py | 10/02/2022 17:54 | Python File |
| desfrag-optimiza.py | 10/02/2022 17:54 | Python File |
| maximiza.py | 15/02/2022 19:45 | Python File |
| minimiza.py | 29/04/2022 12:49 | Python File |
| regla.py | 22/04/2022 12:41 | Python File |
| scriptp1.py | 29/03/2022 16:37 | Python File |



```
TFG > calendario.py
1 import pyautogui
2 import time
3 pyautogui.screenshot()
4 print(pyautogui.locateOnScreen("calendario.PNG"))
```

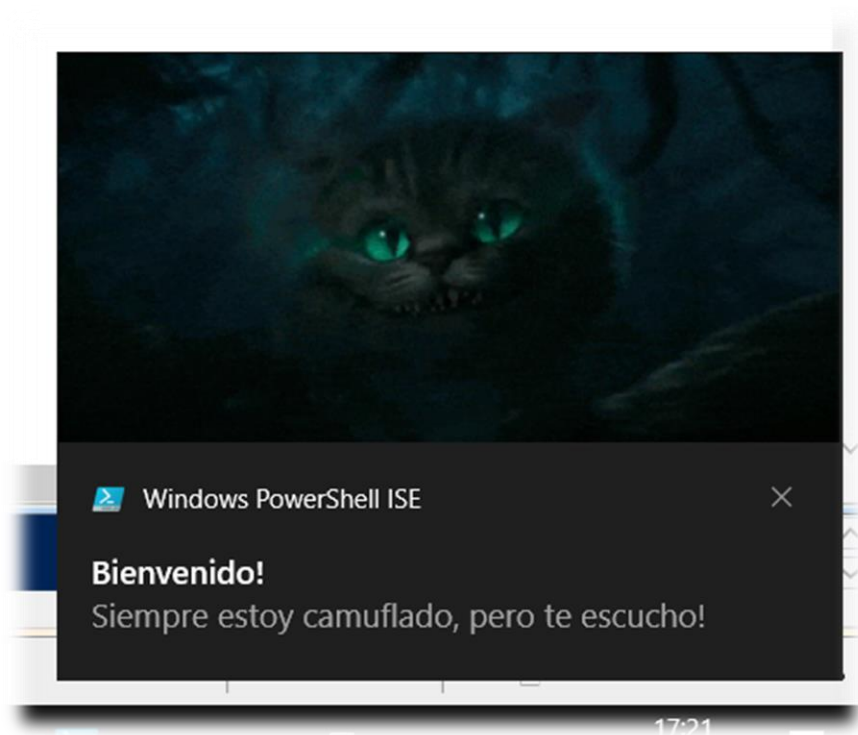
Estos códigos van relacionados con una captura de pantalla para que este donde este el objetivo, recopile sus coordenadas.

- **Establecer notificaciones emergentes**

A medida que fui creando el código, para poderlo hacer sin fallos, necesitaba crear algún tipo de señal para verificar que realmente me entendía, o que hacía justo lo que yo quería y no otra cosa.

De aquí salieron las notificaciones, las cuales hay una personalizada para cada orden, en un primer principio las hice ensamblando las del propio Windows, pero al verlas muy pobres decidí buscar el cómo hacerlas.

Encontré un módulo de PowerShell llamado burnttoast, dando origen a cosas como esta.



- **Explicación código**

(no estarán todos los apartados ya que fluctúa mucho debido a actualizaciones, mejoras, bugs o nuevas lógicas a seguir).

- **Notepad**, el cual se inicia y se para mediante código PowerShell.
- Para ejecutar **XAMPP**, accede a su ruta absoluta e incide en su .exe, para arrancar este programa, elegí una combinación en tiempo real de teclas, en donde al saltar la ventana la ruta de teclas que he establecido arrancara directamente XAMPP y otra para MySQL, siendo combinables.
- **Chrome** incide en su .exe de su ruta absoluta, así como su cierre.
- **Desfragmentador** incide en su ruta absoluta, las opciones que ofrece este programa, las definí mediante Python, para poder seleccionar disco, así como sus acciones.

El código Python utiliza un proceso de detección de imagen, detectando un punto de la pantalla según la imagen que le haya establecido, luego mediante una DLL cargada, PowerShell hace clic sobre la posición que Python ha relacionado.

El cierre del desfragmentador actúa sobre la ventana, esta combinado con órdenes como cerrar minimizar y maximizar.
- El **apagado y reinicio** son comandos que actúan sobre el nombre del equipo.

- **Cisco** incide sobre una ruta absoluta, con cierre combinado de ventanas, la posibilidad de selección de router, etc.

Es el mismo procedimiento que con el desfragmentador, Python localiza una imagen y PowerShell actúa sobre ella.

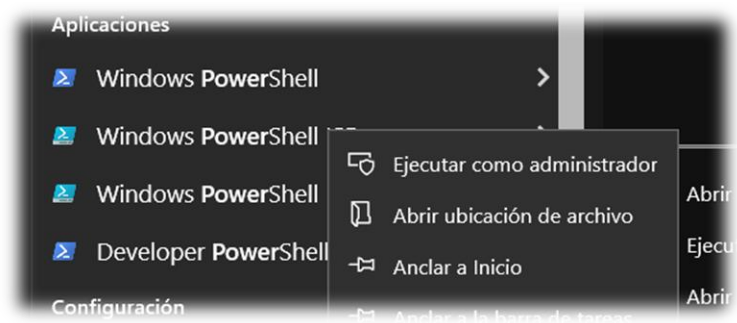
- La apertura del **calendario** actúa sobre Python y PowerShell, pero el cierre sobre el proceso, es una situación complicada porque no es un programa, es una aplicación.
- Una solución amplia y eficiente fueron las ordenes **minimiza cierra** y **maximiza**, las tres funcionan con Python, y actúan sobre la barra, cuadrado y x de las ventanas, ventanas del sistema, no incide sobre ventana como Google chrome que son ajenas.
- El **disco duro** lo calcula mediante cmdlets de PowerShell y te dictaminan la cantidad de almacenamiento que tiene en los discos, haciendo una llamada al sistema.
- **Explorador** te ejecuta el explorador de archivos, actuando sobre el nombre del programa siendo combinable con otras ordenes de ventanas.
- **Panel de control** inicia con el nombre del programa, pero accede al panel de configuración de Windows 10, no al panel de Windows 7.
- **Cortafuegos** inicia con una llamada al panel de control y dentro de este al firewall, con cierre de ventanas combinable. También puedes ejecutar una orden de nueva regla.

- El **registro** actúa con una llamada directa al nombre del programa, con cierre de ventanas combinable.
- **Adaptadores** recopilara información de adaptadores y la lanzara en forma de notificación para evitar residuos de ficheros, y un acceso más rápido a la información.
- **Árbol** es una mini herramienta de información de archivos, la cual, dándole una ruta, te creara una estructura en árbol de la información, carpeta madre, subdirectorios etc.
- **Recopilar** crea un directorio, el cual sobrescribe si este ya está creado y en el incluye diferentes ficheros agrupados según el tipo de información a consultar.
- Incluyo también una orden llamada **Andel** en honor al centro, por la buena enseñanza y por el valor de los profesores, ya que lo aprendido y las formas ha sido excelente.

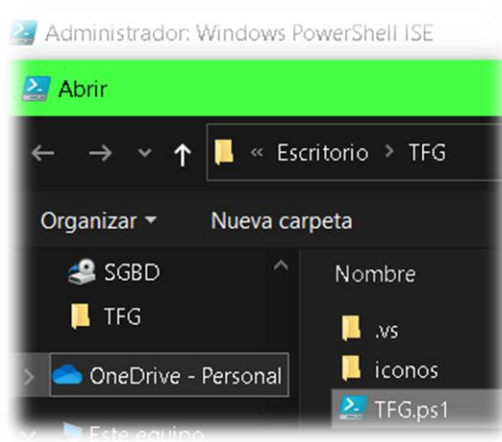
6. Aplicación práctica

El ejemplo práctico que realizare, será una consulta de el almacenamiento que tengo, ya que mirare la unidad D: accediendo al explorador, pero al ver la cantidad de elementos, realizo una consulta en árbol para ver la información más rápidamente.

Para iniciar el código debemos arrancar **PowerShell en administrador**.

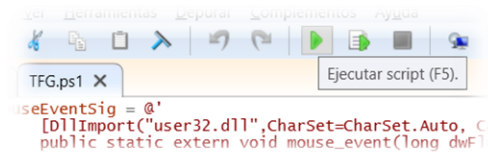


Una vez dentro cargamos el script .ps1.

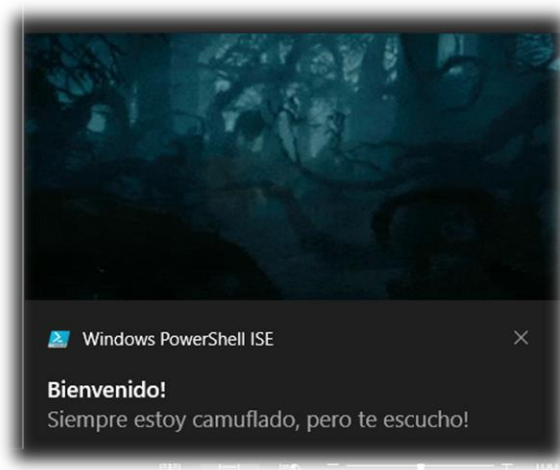


ADMINISTRACION DE SISTEMAS OPERATIVOS POR VOZ

Una vez dentro solo debemos presionar **F5** o el botón verde del “**play**”.



Automáticamente empezara a ejecutarse el bucle en escucha y saltara esta notificación.



Al decir ‘**disco duro**’, saltara esta notificación calculando las unidades.



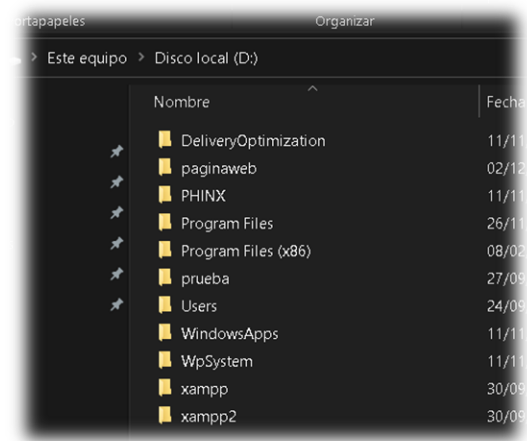
La unidad D: tiene este almacenamiento.



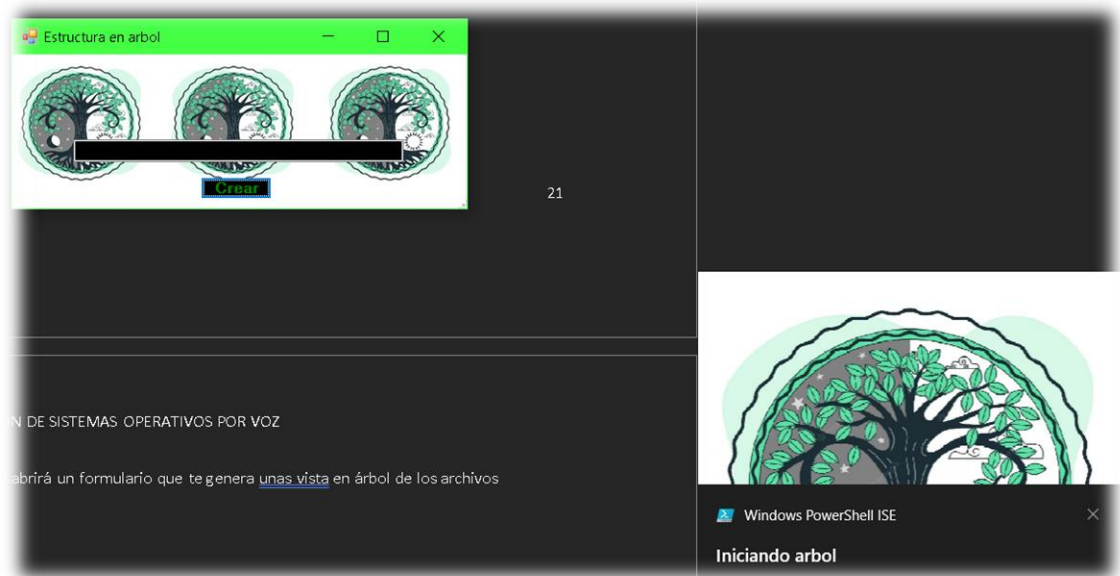
Al decir ‘**explorador**’, accedemos a este.



Al ver que tengo muchas carpetas, decido ejecutar un árbol.



Al decir **‘árbol’** se abrirá un formulario que te genera una vista en árbol de los archivos.



Copiamos y pegamos la ruta y le damos a crear.



Finalmente decimos **‘cerrar’** y cerrara la ventana.



7. Conclusiones

Después de este tiempo desarrollando mi proyecto, esto de utilizar la voz al ser tan tedioso y complejo ahora entiendo porque los asistentes por voz son tan limitados, pero a su vez un buen código elaborado te permitiría trabajos más veloces e incluso lanzar códigos propios automatizados con solo una palabra, este código lo desarrolle entorno a un medioambiente de clase, pero poniendo un ejemplo, si se aplicase a una casa, serian ordenes como persianas, luz, temperatura etc.

Convirtiéndose en un código de voz de administración del hogar, en mi opinión algo con mucho potencial.

8. Líneas abiertas/futuras ampliaciones

Ante todo, debo decir, que en mi opinión es muchísimo más desarrollable este proyecto, pero tiene varios inconvenientes:

- Rutas absolutas, las cuales si el programa cambia de lugar queda obsoleta la orden, a no ser que se cambie la ruta en el código (más **tedioso**).
- **Entrenamiento** costoso del motor de voz, ya que se necesita una cantidad de tiempo importante para que tu ordenador reconozca lo que se hable.
- Una **lógica** con buenos cimientos ya que la ejecución de scripts no controlados puede llevar a problemas graves, así como dar rodeos, o malas ejecuciones de lo deseado.

Las **ampliaciones** de este proyecto son incontables, ya que, combinando órdenes y tipos de lenguaje como PowerShell y Python, por ejemplo, en el caso de cisco, podría llegar a hacerse todo un apartado para utilizarlo completamente por voz, pero esto conlleva mucho tiempo y esfuerzo, he preferido centrarme en más cosas menos desarrolladas para poder enseñar su potencial.

9. Bibliografía/Webgrafía

Documentación:

- Cortana

https://es.wikipedia.org/wiki/Microsoft_Cortana

- Alexa

https://es.wikipedia.org/wiki/Amazon_Alexa

- Siri

<https://es.wikipedia.org/wiki/Siri>

- Motor de voz de Windows

<https://support.microsoft.com/es-es/windows/comandos-de-reconocimiento-de-voz-de-windows-9d25ef36-994d-f367-a81a>

[a326160128c7#:~:text=Windows%20El%20reconocimiento%20de%20voz,usar%20con%20reconocimiento%20de%20voz](https://support.microsoft.com/es-es/windows/comandos-de-reconocimiento-de-voz-de-windows-9d25ef36-994d-f367-a81a?sim=Windows%20El%20reconocimiento%20de%20voz,usar%20con%20reconocimiento%20de%20voz)

<https://docs.microsoft.com/es-es/powershell/scripting/overview?view=powershell-7.2>

- Python

<https://www.becas-santander.com/es/blog/python-que-es.html>

- Powershell

<https://docs.microsoft.com/es-es/powershell/scripting/overview?view=powershell-7.2>

Módulos y notificaciones:

- Error PowerShell - burnt toast:

<https://github.com/MicrosoftDocs/OfficeDocs-SkypeForBusiness/issues/3094>

<https://www.easy365manager.com/install-module-a-parameter-cannot-be-found-that-matches-parameter-name-allowprerelease/>

- Burntoast

<https://github.com/Windos/BurntToast>

- Iconos

<https://icon-icons.com/>

Código:

- Formulario:

<https://www.jesusninoc.com/01/30/crear-un-formulario-en-powershell/>

El resto del código son desarrollos propios de lo estudiado a lo largo del curso.