Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura
Cátedra UNESCO Tecnologías de apoyo para la Inclusión Educativa
GIIATA
GIE-RED
ECUADOR SALESIANOS DON BOSCO

REVISTA

JUVENTUD Y CIENCIA SOLIDARIA:

En el camino de la investigación

Desarrollo de un prototipo de asistente virtual enfocado a personas con discapacidad

Diego Patricio Rivera Guachichulca

Diego Patricio Rivera Guachichulca, nací el 4 de diciembre del 2004, tengo 15 años, soy estudiante de la Unidad Educativa Técnico Salesiano, actualmente pertenezco al grupo del 2E2. Soy un apasionado por el desarrollo web, me gustan los videojuegos y siempre me ha interesado el mundo de la programación; mi género musical favorito es el power metal.

Resumen

El siguiente artículo se basa en el desarrollo de un asistente virtual enfocado en una tecnología de inclusión. El objetivo de este proyecto es facilitar las tareas cotidianas a las personas con discapacidad. Para el futuro se espera que pueda ser utilizado por personas con discapacidad visual para que usen sus dispositivos con mayor facilidad. Este proyecto fue programado usando C# como base y también Microsoft Visual Studio (2019) para elaborar la interfaz gráfica. Actualmente el programa recibe con éxito órdenes como: «Abrir Word», «Abrir Google», pero más adelante se espera ampliar para explorar archivos. Se proyecta la creación de un software que garantice el correcto y completo uso de dispositivos para personas con discapacidades y también un aumento de productividad, esto a manera de contribución con las tecnologías inclusivas.

Palabras clave: C#, comandos, Visual Studio, programación, clase, librería, atajos de teclado

Explicación del tema

Antecedentes

Uno de los objetivos de este proyecto es el ayudar a personas con discapacidad a manejar sus dispositivos

con mayor facilidad, mediante la implementación de tecnologías inclusivas, en este caso, con el desarrollo de un asistente virtual.

Con el avance de la tecnología, el ser humano se ha centrado en la búsqueda de la eficiencia en su trabajo. Existen múltiples asistentes virtuales que simplifican las tareas cotidianas, entre estos se pueden diferenciar: Alfred, Cortana y el asistente de Google [1], los mismos han servido de inspiración para el presente trabajo, pues tienen un gran impacto en la sociedad cuando presentan soluciones a problemas que se afrontan en la vida cotidiana. Parte de la base de los asistentes son los atajos de teclado que tiene el editor de texto Vim. Se desea implementar todos estos recursos en este prototipo, en búsqueda de mayor eficiencia para facilitar su uso a la gran parte de las personas.

Para iniciar este proyecto se buscó un programa que permitiera usar libremente C#, además que se pueda diseñar interfaces gráficas; la respuesta a esto fue Microsoft Visual Studio (2019).

Herramientas

Visual Studio

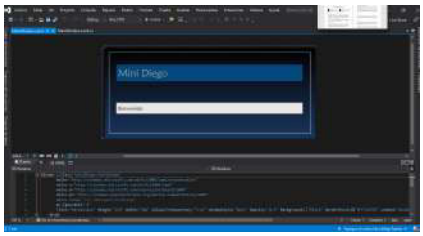

Figura 1. Ambiente de trabajo de Visual Studio
Fuente: Autor

Visual Studio es un entorno de desarrollo que facilita la conexión de las interfaces gráficas con el código y sus variantes.

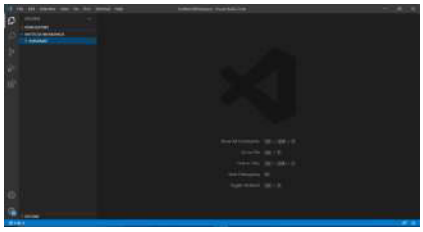

Figura 2. Ambiente de trabajo de Visual Studio Code
Fuente: Autor

Blend

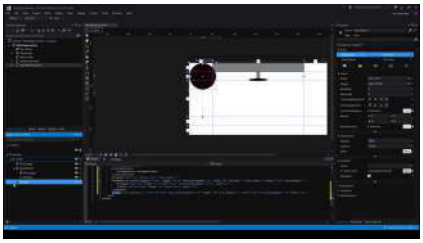

Figura 3. Ambiente de trabajo del software Blend
Fuente: Autor.

Es una de las herramientas más utilizadas en la actualidad ya sea para programación orientada a objetos (POO), desarrollo web y creación de aplicaciones. Como fue desarrollado por Microsoft, hace que haya gran cantidad de información en Internet para intentar solucionar algún problema que se puede presentar.

Desarrollo del código

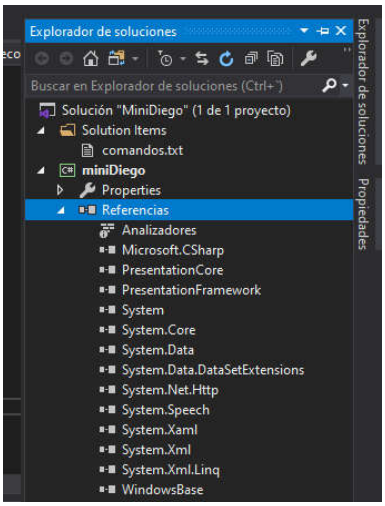

Figura 4. Ventana de referencias en el explorador de soluciones
Fuente: Autor

Además de la instalación del Visual Studio se requieren las extensiones necesarias para el desarrollo del proyecto descargadas directamente desde el mismo Installer de Visual Studio: Desarrollo de escritorio.net, Desarrollo para el escritorio con C++, Desarrollo móvil con C++.

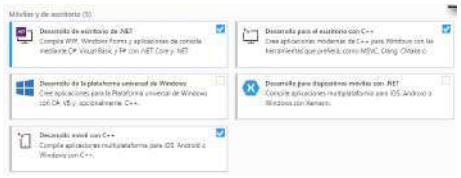

Figura 5. Extensiones usadas para el desarrollo del proyecto
Fuente: Autor

Se buscó una referencia (librería) que permitiera realizar el reconocimiento de voz; la referencia seleccionada fue System.speech que viene con las extensiones previamente instaladas. La manera de agregar la referencia al proyecto es añadir como nueva referencia desde el explorador de soluciones.

De esta referencia obtenemos varias librerías, pero las que se usaron fueron System.Speech.Synthesis y System.Speech.Recognition. Para añadir las bibliotecas se las tiene que poner con la instrucción «using» en la parte de referencias en el código [2].

Parte de referencias en el código

```
using System.Speech.Synthesis;//Salida de voz
using System.Speech.Recognition;//Reconocedor de voz
```

Figura 6. Implementación System Speech al código
Fuente: Autor

Las librerías necesarias se implementaron en el código.

Lo más recomendable es realizar la interfaz para después enlazar con lo que sería el código. Al principio no es necesario enfocarse en realizar una buena interfaz (la interfaz actual del proyecto no siempre fue la misma, simplemente tenía una etiqueta para el reconocimiento del audio).

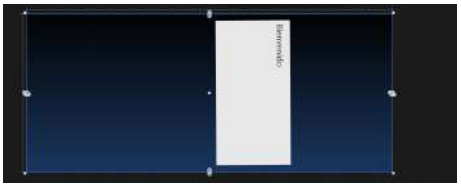

Figura 7. Primer intento del desarrollo de la Interfaz
Fuente: Autor

Posteriormente se siguió desarrollando hasta tener una funcionalidad y un diseño adecuados.

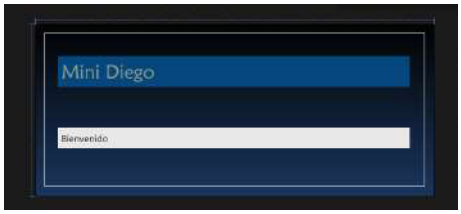

Figura 8. Interfaz Final desarrollada para el proyecto
Fuente: Autor

Viene la parte más importante, el código; en el inicio del código, como ya se había mencionado, va la parte de las referencias y librerías que se están usando y de ser necesario se les documenta una pequeña descripción con «//».

```
//Inicio del apartado de librerias y referencias
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Speech.Synthesis;//Salida de voz
using System.Speech.Recognition;//Reconocedor de voz
//Fin del apartado de librerias y refrencias
```

Figura 9. Todas las librerías y referencias utilizadas
Fuente: Autor

Después de haber definido todas las referencias y librerías que se van a usar, se procedió a crear dos objetos y dos variables.

En el siguiente método se pone que se está iniciando el sistema y para asegurarse que funciona el asistente nos dirá «Iniciando», esto se realiza con el código: ***miniDiego.SpeakAsync ("Iniciando");***

```
public MainWindow()
{
    InitializeComponent();//Iniciando sistema
    miniDiego.SpeakAsync("Iniciando");//Habla
}
```

Figura 10. Iniciación del código y del asistente
Fuente: Autor

A continuación, vienen los métodos principales.

Primero está el método en el que se realizan las instrucciones dependiendo del comando que el asistente haya entendido.

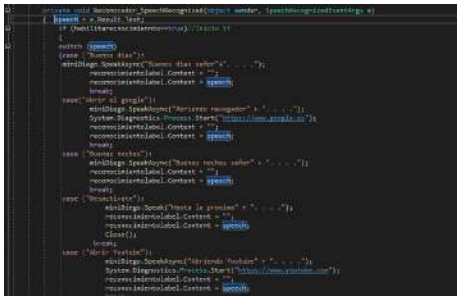

Figura 11. Estructura switch con las instrucciones
Fuente: Autor.

En este se encuentran todas las acciones que va a realizar el asistente.

Se utilizó un switch que sirve para generar varios casos, es decir, que dependiendo del caso se realizará una acción.

Figura 12. Ejemplo del uso de la estructura switch
Fuente: Autor.

Una vez definidas las acciones se pone lo que realizará, es decir, si nos dice buenos días o buenas noches [3].

Figura 13. Ejemplo de diferentes acciones que se pueden utilizar
Fuente: Autor

La instrucción SpeakAsync es para que el asistente realice la acción de hablar [4].

Para realizar acciones como abrir el Google se realiza la instrucción: System.Diagnostics.Process.Start, a continuación, un ejemplo:

Figura 14. Para cerrar el asistente se utiliza el comando Close
Fuente: [3]

Figura 15. Código para desactivar el programa.
Fuente: Autor

Para saber si el reconocimiento está activo se creó una variable del tipo booleano.

Si el booleano es false el reconocimiento de voz se deshabilita.

Pero si es true significa que está activo y se realizará el proceso con normalidad.

Figura 16. Acciones en caso de que el booleano sea true
Fuente: Autora

Después se debe iniciar todos los elementos necesarios para el reconocimiento de voz y funcionamiento correcto del asistente [2].

Luego se escriben las instrucciones que va a recibir el asistente usando: *reconocedor.LoadGrammarAsync(new Grammar(newGrammarBuilder(new Choices(""))))*, esto debe ir en el mismo método en el que se iniciaron los elementos para el reconocimiento de voz y funcionamiento del asistente.

Figura 17. Objetos y generador de gramáticas
Fuente: Autor

Y, por último, se le enlaza en la ventana principal para que esté presente en todo el código [2].

Figura 18. Enlace de los pasos desarrollados a la ventana principal
Fuente: Autor

Funcionamiento

La siguiente foto sirve para evidenciar el funcionamiento del asistente; se puede ver su interfaz en acción junto con el navegador abierto que fue la orden que se le dio. Se ejecutó el programa y se le ordenó que abriera Google, como se puede apreciar en la fotografía encontramos Google abierto y el programa ejecutado.

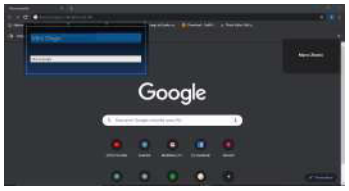

Figura 19. Funcionamiento del Asistente en Google
Fuente: Autor

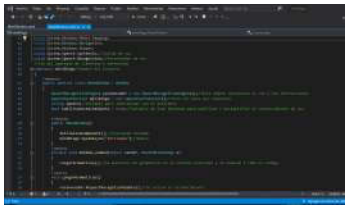

Figura 20. Interfaz del Asistente funcional
Fuente: Autor

Conclusiones

El resultado que se obtuvo del proyecto es bastante positivo ya que en casi todas las pruebas este reconoce exactamente lo que se dice. Dependiendo de la instrucción recibida siempre realiza la acción deseada, además de que después de varias pruebas se encontró la manera de que no sea introducida otra instrucción antes de que se acabe una anterior que se estaba ejecutando. Como se puede apreciar en un futuro puede ser un asistente con tecnología inclusiva para personas con discapacidades, no videntes o que no puedan usar un teclado o un mouse.

En el proceso hubo varios intentos fallidos, por ejemplo, en varias ocasiones el programa no compilaba por estructuras mal hechas, pero además de esos fallos hubo con el recibimiento de instrucciones y la eficiencia del asistente.

A continuación, se enlistan algunos de los fallos que existieron durante el desarrollo del programa:

Label no definido

Para que se pueda ver la salida de texto se usó un *label*; este fue el primer error que se dio porque no se hallaba manera de crear el objeto *label* en la parte de código. La solución fue enlazar el *label* de la parte de diseño al código creando un objeto.

Generador de gramáticas no encontrado

La idea original era que las instrucciones estén en un archivo de texto enlazado con la parte de código; se enlazan de manera correcta, pero simplemente captaba una instrucción y las otras no eran reconocidas, la solución fue poner las instrucciones dentro de la parte de código y cambiar la estructura del generador de gramáticas.

No se ejecutaban los programas

En las primeras pruebas para probar el asistente no se ejecutaban los programas de Office, estando definidos como word.exe, la solución fue investigar el nombre del archivo y cambiar word.exe por winword.exe.

Agradecimientos

Agradezco a todas las personas que me apoyaron en el desarrollo de este proyecto; al ingeniero Freddy Ávila, quien me inculcó el gusto por la programación, así como también al ingeniero Wilson Cedillo, quien me enseñó los primeros pasos para programar.

A mis tutores, los señores Pablo Celi, David Montesdeoca, Kevin Pinta y Marco Zhunio, quienes me brindaron su tiempo y dedicación para darme las pautas necesarias para culminar este proyecto satisfactoriamente.

Al ingeniero Néstor Rivera quien se encargó del vínculo entre la Unidad Educativa Técnico Salesiano y la Universidad Politécnica Salesiana

Referencias

[1] Microsoft, «SpeechSynthesizer Class (System.Speech.Synthesis),» 2020. [En línea]. Disponible en https://bit.ly/362qutV. [Último acceso: 12 noviembre 2020].

[2] Microsoft, «SpeechRecognitionEngineClass (System.Speech.Recognition),» 2020. [En línea]. Disponible en https://bit.ly/399PzoV. [Último acceso: 12 noviembre 2020].

[3] L. Joyanes, «Programación en C, C++, Java y UML 2da Edición (McGrawhill),» 2009.

[4] Google, «Asistente virtual de Google,» 2020. [En línea]. Disponible en https://bit.ly/3l3f0dV. [Último acceso: 12 noviembre 2020].