UNIVERSIDAD DE SALAMANCA

Memoria

ESCUELA POLITÉCNICA SUPERIOR DE ZAMORA

APLICACIÓN DE ACCESIBILIDAD MEDIANTE SEGUIMIENTO OCULAR

Autor: **Jorge Enrique González Gonzalo**

Tutor: **Jesús Ángel Román Gallego**

Departamento de Informática y Automática

Fecha de Adjudicación: 17/11/2023

Fecha de presentación: Febrero 2024

ÍNDICE

Lista de abreviaturas

1. **TFG**

   Trabajo fin de grado

2. **XAML**

   eXtensible Application Markup Language

3. **WPF**

   Windows Presentation Foundation

4. **MVVM**

   Model View ViewModel

5. **RAD**

   Rapid Application Development

6. **CNN**

   Convolutional neutral network

7. **RNN**

   Recurrent neural network

8. **API**

   Application Programming Interface

9. **YOLO**

   You Only Look Once

Lista de ilustraciones

1. Introducción

A lo largo de esta memoria se explicará el proceso de desarrollo del Trabajo Final de Grado, "Aplicación de accesibilidad mediante seguimiento ocular", el objetivo de este proyecto es crear una aplicación en Windows para facilitar la comunicación de personas con movilidad reducida, está aplicación está complementada con un seguidor ocular que funciona gracias a una cámara, ya sea la interna del dispositivo o una externa.

1.1. Motivación

Durante mi periodo de prácticas se comentaros diversos temas y dispositivos de interés, entre esos dispositivos se encontraba el *Tobii Eye Tracker 5*, este dispositivo es capaz de hacer un seguimiento de la cabeza y los ojos simultáneamente, pero debido a su precio pensé en hacer una versión, que, aunque fuera menos precisa, fuera alcanzable al mayor número de personas posible, eliminando el coste del dispositivo y haciendo que funcione únicamente con la cámara del dispositivo.

Además, como motivación más personal, se ha buscado un tema en el que se tuviera poca experiencia, ya que se ha buscado ampliar conocimiento y hacer un proyecto que poco tuviera que ver con lo aprendido a lo largo de la carrera. Si bien es cierto que los conocimientos de la carrera han sido una parte clave para el desarrollo del trabajo fin de grado, se ha buscado usar tecnologías y lenguajes que no se habían usado con anterioridad.

1.2. Contenido de la memoria

Con el fin de realizar la memoria acorde a la normativa de la Escuela Politécnica Superior de Zamora se han seguido las normas de estilo, las cuales se encuentran en la página oficial de la universidad.

1.2.1. Documentación técnica

Dentro de este apartado se ven los diferentes anexos que complementan la memoria, acompañados de una breve explicación de los contenidos que contiene cada uno de ellos.

- **Anexo 1 – Plan de Proyecto Software**
  Se define el plan temporal del proyecto y además se aporta una breve descripción de cada apartado.
- **Anexo 2 – Especificación de requisitos**
  Se define el comportamiento del sistema, se tratan los objetivos del sistema, así como los requisitos funcionales, no funcionales y de la información.
- **Anexo 3 – Especificación de diseño**
  Se definen y explican las interfaces del **TFG**, la interfaz de la configuración y la interfaz de la aplicación.
- **Anexo 4 – Manual técnico de programador**
  Se define el contenido de la carpeta del TFG, explicando cada directorio de forma individual, así como el contenido de estos y su finalidad.
- **Anexo 5 – Manual de usuario**
  Se define los requisitos para el funcionamiento de la aplicación y el paso a paso de la instalación de todas las funcionalidades necesarias para un correcto funcionamiento.

2. Objetivos del proyecto

En este apartado se detallan los objetivos del proyecto, aunque como se comentó anteriormente el objetivo principal de este proyecto es proporcionar una aplicación para facilitar la comunicación de personas con movilidad reducida, esta aplicación consta de botones reproducibles en audio, el usuario puede escribir lo que desee y posteriormente reproducir ese texto en audio. La forma de uso de esta aplicación y además la integración de la accesibilidad viene de un seguidor ocular, que detecta mediante la cámara del ordenador la posición de los ojos con el fin de mover el cursor del ratón, además si se mantienen los ojos cerrados un determinado tiempo se realizará un clic, más adelante en la memoria se explica como modificar este parámetro y diferentes recomendaciones.

2.1. Objetivos específicos

Para este proyecto se han fijado una serie de objetivos específicos, los cuales son:

- El desarrollo de un seguidor ocular que no necesite de periféricos especiales y pueda sustituir en gran parte a un ratón de ordenado.
- Gestionar botones reproducibles, esto se refiere a poder añadir y eliminar botones que pueden ser reproducidos en audio una vez fijado el texto que desee el usuario.
- El desarrollo de una aplicación en Windows que permita junto al seguidor ocular gestionar los botones mencionados anteriormente, además de hacer los procesos necesarios para poder convertir el texto en audio.
- La adaptabilidad del seguidor ocular a los rangos de movimiento del usuario.
- La accesibilidad de la aplicación enfocada al uso del seguidor ocular.
- El correcto almacenamiento de la información.

2.2. Objetivos técnicos

Como se ha comentado de manera breve en la motivación del trabajo, se ha buscado usar lenguajes y tecnologías que no se habían usado con anterioridad.

- Para el almacenamiento de información se han usado dos tipos de archivos, archivos .json y archivos .csv, los cuales se explicarán más adelante.
- Para el desarrollo del seguidor ocular se ha usado Python acompañado por diferentes librerías.
- En cuanto a la aplicación de Windows, para la parte de la interfaz de usuario y la parte visual, se ha usado **XAML**, pero para la parte funcional se usa C#.
- La aplicación de Windows además de usar los lenguajes mencionados anteriormente se ha creado mediante un marco de interfaces de usuario que crea aplicaciones de cliente de escritorio, llamado **WPF** y organizado junto a un patrón de diseño llamado **MVVM** con el fin de separar la interfaz de usuario de la parte lógica.

3. Conceptos teóricos

Durante la realización de este proyecto se han tenido en cuenta todos los conocimientos que se han adquirido a lo largo del grado, ya que han permitido comprender los diferentes lenguajes, aplicar correctamente el patrón de diseño, incluso implementar la interfaz de usuario, así como almacenar la información y poder usarla o modificarla de una forma eficiente.

Para un desarrollo correcto del proyecto se ha usado una metodología ágil llamada *Metodología RAD*.

3.1. Metodologías ágiles

El enfoque ágil para el desarrollo de software busca implementar sistemas de software en funcionamiento de manera continua mediante iteraciones rápidas. Aunque la expresión "metodología ágil" puede ser engañosa, ya que no prescribe instrucciones específicas para el desarrollo, se refiere más a una mentalidad colaborativa y a la definición de valores que orientan las decisiones en el proceso. Las metodologías ágiles buscan proporcionar rápidamente pequeñas piezas de software funcional para mejorar la satisfacción del cliente, empleando enfoques flexibles y trabajo en equipo para lograr mejoras continuas. Este enfoque implica la colaboración regular de pequeños equipos autoorganizados de desarrolladores y representantes empresariales durante todo el ciclo de vida del desarrollo, favoreciendo una documentación sencilla y aceptando cambios en lugar de resistirse a ellos. [1]

Los principales puntos de estas metodologías son los siguientes:

- **Iterativo e Incremental:**
  Se basa en ciclos de desarrollo cortos y repetitivos, entregando incrementos funcionales del software en cada iteración para adaptarse a cambios y mejoras continuas.
- **Colaboración y Comunicación:**
  Prioriza la interacción constante entre los miembros del equipo de desarrollo fomentando la comunicación efectiva para comprender y satisfacer las necesidades del cliente.
- **Flexibilidad y Adaptabilidad:**
  Se adapta fácilmente a cambios en los requisitos del proyecto, permitiendo ajustes durante todo el proceso de desarrollo en lugar de seguir rigurosamente un plan inicial.
- **Entrega Rápida y Valor al Cliente:**
  Busca proporcionar rápidamente funcionalidades utilizables, priorizando la satisfacción del cliente al ofrecer valor tangible en cada entrega, en lugar de esperar hasta el final del proyecto.

3.1.1. Metodología RAD

La metodología del desarrollo rápido de aplicaciones es un enfoque ágil que se centra en la creación rápida de prototipos sobre la planificación costosa. Este modelo permite priorizar el trabajo antes que las palabras, siguiendo una serie de fases donde se obvia una planificación estricta.

Pese a que este modelo ha evolucionado a lo largo de los años, los puntos principales se pueden definir como los siguientes:

- Definir los requisitos
  Ya que no se requieren requisitos detallados, se fijan requisitos amplios, los cuales permiten definir requisitos más específicos a lo largo del ciclo de desarrollo.
- Prototipo
  En este apartado se realiza el desarrollo, donde se crean prototipos con diferentes características y se define que está bien o mal.
- Recibir comentarios
  Esta parte es una retroalimentación de opiniones, donde se toman en cuenta comentarios para continuar con el prototipo y hacer que cumpla con los requisitos, cuando estos comentarios son estrictamente positivos se pasa al siguiente paso, sino se repiten los anteriores.
- Finalizar el software
  En esta etapa, se finalizan características, funciones, estética e interfaz del software y además se puede optimizar o rediseñar su implementación con el fin de mejorar la estabilidad, capacidad de mantenimiento y la usabilidad.

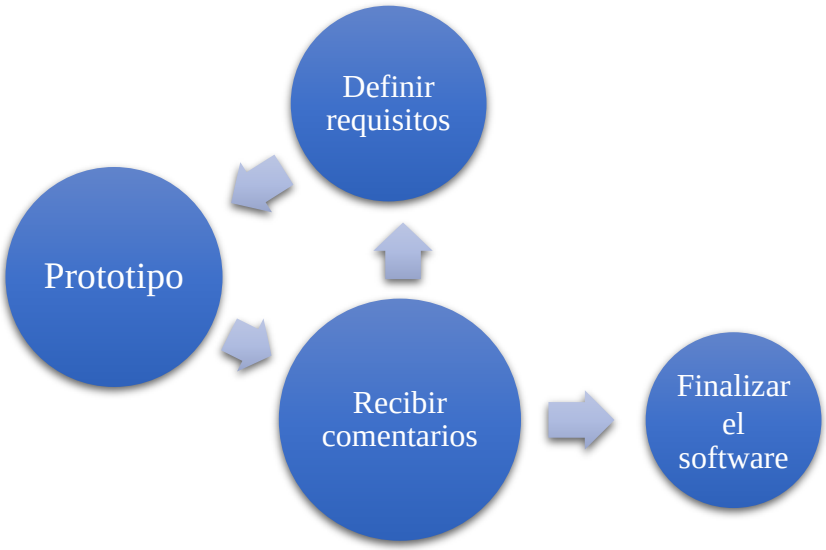

Ilustración 1: Puntos metodología RAD

Como todas las metodologías tiene ventajas y desventajas.

- **Ventajas**
  - **Velocidad:**

    Es más probable que los proyectos terminen a tiempo y para satisfacción del cliente en el momento de la entrega.
  - **Costo:**

    Con el desarrollo rápido de aplicaciones, los desarrolladores crean los sistemas exactos que requiere el cliente, y nada más.
  - **Satisfacción del desarrollador:**

    Como en RAD el cliente está presente en cada paso del desarrollo y el desarrollador presenta su trabajo con frecuencia, al entregar el trabajo final el desarrollador puede estar confiado en que recibirá buenos comentarios.
- **Desventajas**
  - **Escala:**

    Las prácticas de RAD se complican cuando se expanden más allá de un solo equipo o requieren comunicación entre equipos.
  - **Compromiso:**

El ciclo frecuente de prototipos RAD requiere que los desarrolladores y clientes se comprometan a reuniones frecuentes que, al principio, pueden parecer consumir ciclos innecesarios para ambas partes.

- **Enfoque de interfaz:**
  La metodología RAD motiva a los desarrolladores a encontrar la solución perfecta para el cliente. El cliente juzga la calidad de la solución en su interacción y, a menudo, todo con lo que interactúa es una fachada. [2]

3.2. Visión artificial

La visión artificial es un campo de la inteligencia artificial (IA) que permite a los ordenadores y sistemas extraer información significativa a partir de imágenes digitales, videos y otras entradas visuales, y tomar medidas o realizar recomendaciones en función de esa información. Si la IA permite a los ordenadores pensar, la visión artificial les permite ver, observar y comprender.

La visión artificial funciona de manera muy similar a la visión humana, excepto que los humanos tienen una ventaja inicial. La vista humana tiene la ventaja de toda una vida de contexto para entrenar cómo distinguir los objetos, a qué distancia están, si se están moviendo y si hay algo mal en una imagen.

La visión artificial entrena a las máquinas para realizar estas funciones, pero tiene que hacerlo en mucho menos tiempo con cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual. Como un sistema entrenado para inspeccionar productos o ver un activo de producción puede analizar miles de productos o procesos por minuto, detectando defectos o problemas imperceptibles, puede superar rápidamente las capacidades humanas.

La visión artificial se utiliza en sectores que van desde la energía y los servicios públicos hasta la fabricación y la automoción, y el mercado sigue creciendo.

La visión artificial depende de la recopilación y análisis de grandes cantidades de datos para reconocer imágenes. Se utilizan dos tecnologías principales: machine learning, que permite a la computadora aprender por sí misma a partir de datos, y redes neuronales convolucionales (**CNN**), que descomponen las imágenes en píxeles etiquetados para hacer predicciones precisas. Las CNN funcionan mediante convoluciones y ajustan sus predicciones iterativamente para mejorar su precisión, lo que les permite reconocer imágenes de manera similar a los humanos. Además, las redes neuronales recurrentes (**RNN**) se utilizan para comprender cómo las imágenes se relacionan entre sí en secuencias de video. [3]

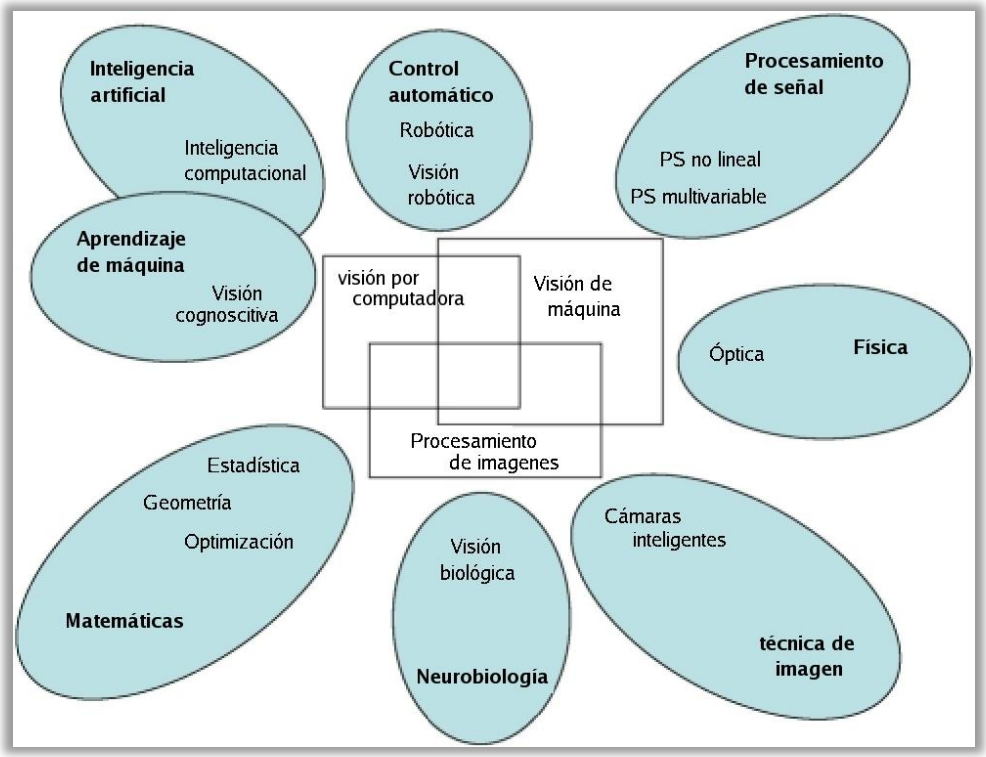

Ilustración 2: Visión inteligente y áreas de conocimiento [4]

3.3. Librerías de Python

Como se ha comentado en el apartado de objetivo, para este trabajo se han usado diferentes librerías de Python, estas se refieren a un conjunto de funcionalidades que permiten al usuario llevar a cabo nuevas tareas que antes no eran posibles, además permiten codificar el lenguaje con el objeto de crear una interfaz independiente. [5]

3.3.1. Dlib

Dlib es un moderno conjunto de herramientas de C++ que contiene algoritmos y herramientas de aprendizaje automático para crear software complejo en C++ para resolver problemas del mundo real. Se utiliza tanto en la industria como en el mundo académico en una amplia gama de dominios que incluyen robótica, dispositivos integrados, teléfonos móviles y grandes entornos informáticos de alto rendimiento. La licencia de código abierto de Dlib permite utilizarlo en cualquier aplicación, de forma gratuita. [6]

Dlib fue creada y mantenida por David King, se creó con deep learning y permite el reconocimiento de caras en una imagen. La librería requiere cargar todos los modelos necesarios, estos incluyen un detector para encontrar la cara, un predictor de forma para situar los puntos de referencia de la cara y un modelo de reconocimiento facial. Esta librería es una de las más importantes en lo que se refiere a reconocimiento facial, ya que se han desarrollado algoritmos para la detección de objetos o reconocimiento de rasgos biométricos.

La red neuronal convolucional está compuesta por 29 capas y fue entrenada con tres millones de caras, lo cual, garantiza una precisión del 99,38% al momento de detectar e identificar un rostro. [7]

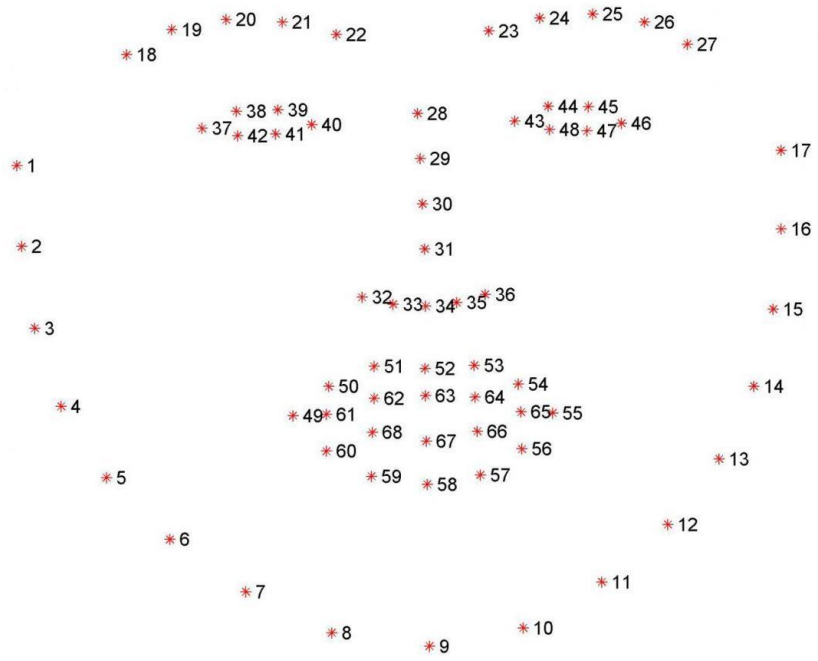

Ilustración 3: Facial_landmarks_68markup

Lo que se aprecia en la Ilustración 3 es una imagen de referencia al predictor de puntos faciales.

3.3.2. OpenCV

OpenCV es una biblioteca de computación visual que se puede usar para desarrollar aplicaciones de visión artificial en Python. Esto significa que podemos aprovechar las herramientas de procesamiento de imágenes que proporciona OpenCV para desarrollar aplicaciones que puedan analizar imágenes, detectar objetos, realizar el seguimiento de objetos, identificar patrones, etc.

Otra ventaja de usar OpenCV en Python es que es una biblioteca de código abierto. Esto significa que podemos acceder al código fuente de OpenCV y modificarlo según sea necesario para adaptarlo a nuestras necesidades. Esto nos permite crear aplicaciones personalizadas que aprovechen las funcionalidades de OpenCV sin tener que reinventar la rueda.

Las herramientas básicas para el procesamiento de imágenes permiten cargar y mostrar una imagen, cambiar su tamaño y recortar partes de ella. Estas operaciones se pueden realizar utilizando la librería de procesamiento de imágenes OpenCV. [8]

Ilustración 4: OpenCV [9]

4. Técnicas y herramientas

Este apartado está enfocado a definir y explicar técnicas como es MVVM o herramientas como Visual Studio Code, además de una pequeña explicación de lo que es y que aporta cada una, también se justifica su uso en este trabajo.

4.1. Técnicas

Como ya se ha comentado en la presentación de este apartado, se han usado diferentes técnicas en este proyecto, ya sea por una futura escalabilidad o por comodidad.

4.1.1. MVVM

La experiencia del desarrollador de .NET normalmente supone crear una interfaz de usuario en XAML y, luego, agregar código subyacente que funcione en la interfaz de usuario. A medida que las aplicaciones se modifican y aumentan de tamaño y ámbito, pueden surgir problemas de mantenimiento complejos. Estos problemas incluyen el acoplamiento estricto entre los controles de interfaz de usuario y la lógica de negocios, lo que aumenta el costo de realizar modificaciones de la interfaz de usuario y la dificultad de realizar pruebas unitarias de este código.

El patrón MVVM ayuda a separar limpiamente la lógica de presentación y negocios de una aplicación de su interfaz de usuario. Mantener una separación limpia entre la lógica de la aplicación y la interfaz de usuario ayuda a abordar numerosos problemas de desarrollo y facilita la prueba, el mantenimiento y la evolución de una aplicación. También puede mejorar considerablemente las oportunidades de reutilización del código y permite a los desarrolladores y a los diseñadores de la interfaz de usuario colaborar más fácilmente al desarrollar sus respectivas partes de una aplicación. [10]

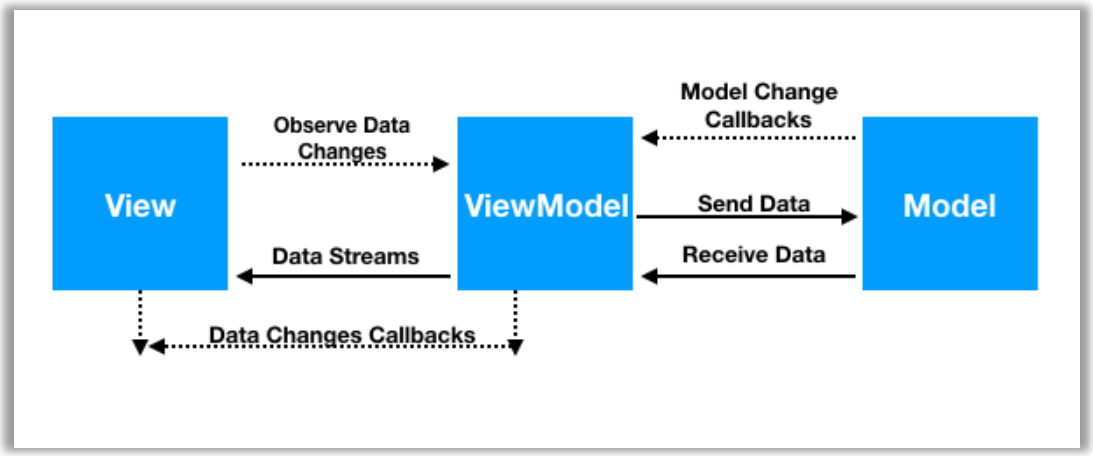

Ilustración 5: MVVM

Como se ha comentado anteriormente y, además, se puede observar en la Ilustración 5, este modelo ha permitido que el proyecto sea más escalable en un futuro gracias a la separación de ámbitos, gracias a tener estas separaciones se pueden realizar cambios ya sea en la interfaz de usuario, como en la lógica de programación sin afectarse mutuamente al aplicar estos cambios. El MVVM se compone de las siguientes partes:

- **Model**

  El modelo representa los datos y la lógica de negocio de la aplicación. El modelo se encarga de obtener los datos que se utilizan en la aplicación.
- **View**

  La vista es la representación visual de la información que obtenemos del ViewModel y tiene una doble función, por un lado, se encarga de lanzar acciones al ViewModel y por otro lado se encarga de escuchar los cambios de modelo que recibe del ViewModel, un cambio en alguna de las propiedades del ViewModel, si la estamos escuchando desde la View, nos enteraremos y podremos realizar algún cambio en la View.
- **ViewModel**

  El ViewModel es el mediador entre la View y el Model. Se encarga de conectar estos dos componentes. Por un lado, nutre a la vista con la información que le llega de la capa inferior, del Modelo. Y por otro lado se encarga de recibir acciones de la View, y estas acciones pueden quedarse en el ViewModel o pueden pasar a la capa del Modelo.

4.1.2. WPF

WPF es un marco de interfaz de usuario que crea aplicaciones de cliente de escritorio. La plataforma de desarrollo de WPF admite un amplio conjunto de características de desarrollo de aplicaciones, incluido un modelo de aplicación, recursos, controles, gráficos, diseños, enlace de datos, documentos y seguridad. Es un subconjunto de .NET Framework. WPF usa el lenguaje XAML para proporcionar un modelo declarativo para la programación de aplicaciones. [11]

4.2. Herramientas

En este apartado se habla de las herramientas que se ha usado para desarrollar el proyecto.

4.2.1. Visual Studio Code

Visual Studio Code es un editor de código fuente liviano pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C++, C#, Java, Python, PHP, Go, .NET). [12]

Visual Studio Code es un editor de código fuente construido sobre el framework Electron. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado. [13]

Ilustración 6: Visual Studio Code

4.2.2. Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Fortran, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros. [14]

Ilustración 7: Microsoft Visual Studio

5. Aspectos relevantes del desarrollo del proyecto

Este apartado está enfocado en mencionar algunos aspectos relevantes a la hora de realizar el proyecto y se justifica el porqué de algunas decisiones.

5.1. Aspectos relevantes del diseño

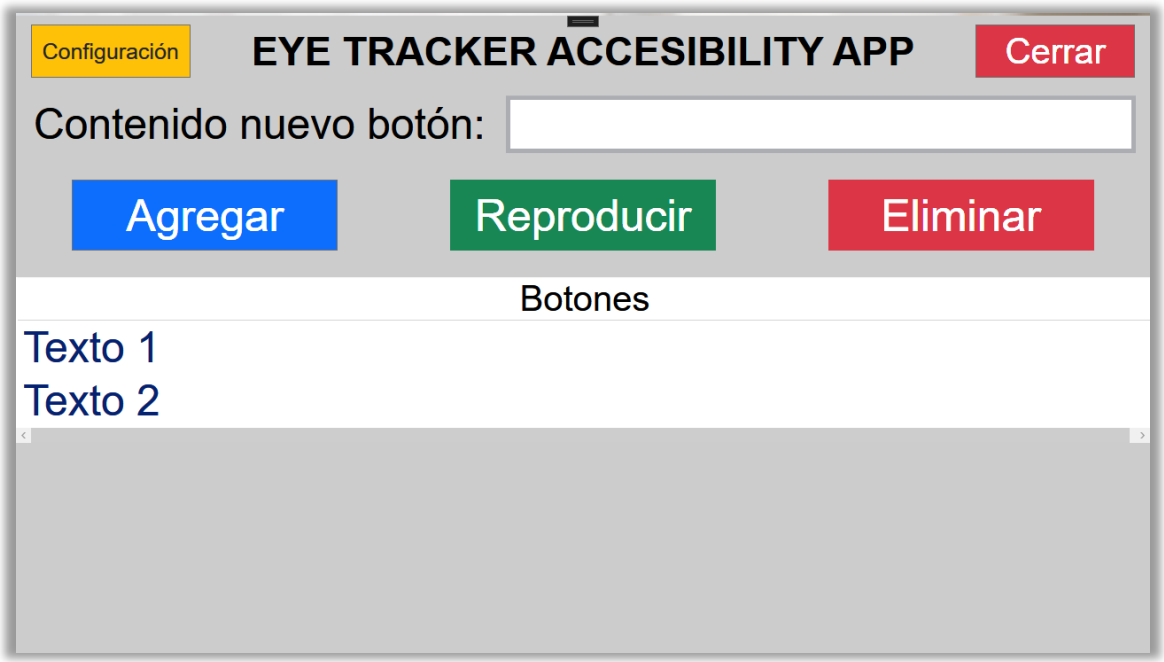

Ilustración 8: Interfaz de usuario

Como se puede observar en la Ilustración 8 se ha tenido en cuenta la posible dificultad de apuntado con el cursor, por lo que se ha diseñado con unos botones de funcionalidades grandes, con el fin de hacer un diseño sencillo e intuitivo, además se ha buscado contar con una sola vista con el fin de minimizar el movimiento.

También se tuvo en cuenta poner un botón para modificar los textos, pero se consideró que el número de acciones que se requerían para modificar un texto serian superiores a eliminar un texto erróneo y crear uno nuevo, por lo que se optó por no implementarlo.

5.2. Aspectos relevantes de la implementación

En este punto se comentan algunos puntos relevantes de la implementación para tener en cuenta. Como son:

- **Punto de referencia del seguidor ocular:**

  El seguidor ocular calcula como punto de referencia el punto medio de ambos ojos, para ello se calcula el punto medio de cada ojo (A, B), y posteriormente el punto medio de los puntos anteriores (C).

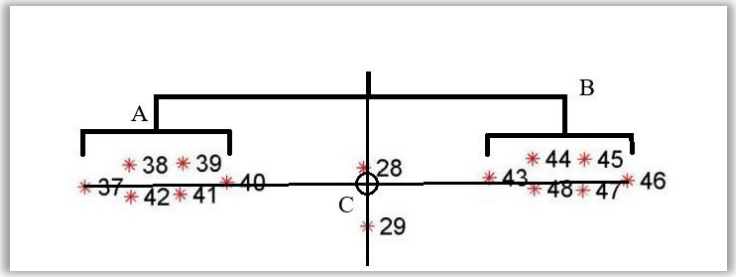

Ilustración 9: Punto de referencia seguidor ocular

  La representación en la Ilustración 9 es únicamente una aproximación para entender donde se sitúa el punto de referencia.

- **Relación de aspecto en el pestañeo:**

  La relación de aspecto del pestañeo hace referencia a la proporción de apertura de los parpados, de esta forma se puede detectar cuando se han cerrado lo suficiente los ojos para poder considerarlo un cierre de ojos.

- **Tiempo de pestañeo:**

  El tiempo de pestañeo se refiere en la aplicación al tiempo que deben permanecer los ojos cerrados para que la aplicación lo entienda como un clic voluntario y no como un pestañeo casual, esto se debe a que durante las pruebas se tuvo que implementar para poder controlar mejor los clics y además se puede personalizar para que cada usuario lo configure como más se acomode a sus gustos.

6. Líneas futuras

Como líneas futuras resaltaré algunas que en mi opinión podrían ser interesantes:

- **YOLO**

  Dentro de la carpeta del proyecto existe una carpeta con un modelo de reconocimiento ocular entrenado con YOLOv5, se podría implementar o mejorar.
- **Juegos educativos con Gamificación**

  Se podría implementar algún juego usando gamificación para reforzar la educación de niños y adolescentes.
- **Mejora de la interfaz**

  Se podría modificar la aplicación para mejorar efectos de los botones, estilo de los textos, etc.

Referencias

[1] RedHat, «¿Qué es la metodología ágil?,» [En línea]. Available: https://www.redhat.com/es/topics/devops/what-is-agile-methodology. [Último acceso: 06 02 2024].

[2] Tecnologías información, «Desarrollo rápido de aaplicaciones (RAD),» [En línea]. Available: https://www.tecnologias-informacion.com/metodologia-rad.html. [Último acceso: 06 Febrero 2024].

[3] IBM, «¿Qué es la visión artificial?,» [En línea]. Available: https://www.ibm.com/es-es/topics/computer-vision. [Último acceso: 06 Febrero 2024].

[4] Wikipedia, «Esquema de las relaciones entre la visión por ordenador y otras áreas afines,» [En línea]. Available: https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial#/media/Archivo:CVoverview2-ES.jpg. [Último acceso: 06 Febrero 2024].

[5] IMMUNE, «Librerías de Python, ¿qué son y cuáles son las mejores?,» [En línea]. Available: https://immune.institute/blog/librerias-python-que-son/. [Último acceso: 06 Febrero 2024].

[6] Dlib, «Dlib C++ Library,» [En línea]. Available: http://dlib.net/. [Último acceso: 06 Febrero 2024].

[7] Dlib, «Automatic Learning Rate Scheduling That Really Works,» [En línea]. Available: https://blog.dlib.net/. [Último acceso: 06 Febrero 2024].

[8] Imagina Formación, «Procesamiento de omágenes con OpenCV en Python,» [En línea]. Available: https://imaginaformacion.com/tutoriales/opencv-en-python. [Último acceso: 06 Febrero 2024].

[9] L. Llamas, «Instalar OpenCV en Linux Mint o Ubuntu,» [En línea]. Available: https://www.luisllamas.es/instalar-opencv-en-linux-mint-o-ubuntu/. [Último acceso: 06 Febrero 2024].

[10] m. g. e. y I. , «Modelo-Vista-Modelo de vista (MVVM),» [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm. [Último acceso: 06 Febrero 2024].

[11] a. y p. . [En línea]. Available: https://learn.microsoft.com/es-es/dotnet/desktop/wpf/getting-started/?view=netframeworkdesktop-4.8. [Último acceso: 06 Febrero 2024].

[12] Visual Studio Code, «Getting Started,» [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: 06 Febrero 2024].

[13] Wikipedia, «Visual Studio Code,» [En línea]. Available: https://es.wikipedia.org/wiki/Visual_Studio_Code. [Último acceso: 06 Febrero 2024].

[14] Wikipedia, «Microsoft Visual Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio. [Último acceso: 06 Febrero 2024].