Aplicación móvil para el control y monitorización de la evolución de la Degeneración Macular Asociada a la Edad (DMAE)

Trabajo Fin de Grado
Curso 2022-2023

Daniel Castro López, Íñigo Rolando García, Sonia Marcos de Benito, Víctor de la Fuente Sabalete

Joaquín Recas Piorno, María Guijarro Mata-García

Grado en Ingeniería de Software y Computadores
Facultad de Informática
Universidad Complutense de Madrid

Aplicación móvil para el control y monitorización de la evolución de la Degeneración Macular Asociada a la Edad (DMAE)

Trabajo de Fin de Grado en Ingeniería de Software y Computadores

Daniel Castro López, Íñigo Rolando García, Sonia Marcos de Benito, Víctor de la Fuente Sabalete

Joaquín Recas Piorno, María Guijarro Mata-García

Convocatoria: Junio 2023

Grado en Ingeniería de Software y Computadores
Facultad de Informática
Universidad Complutense de Madrid

DÍA de MES de AÑO

Resumen

Nuestro trabajo de fin de grado consiste en desarrollar una aplicación Android que mediante la cámara del teléfono permite en tiempo real, grabar y reconocer gestos realizados con la mano por el usuario. La aplicación muestra en pantalla la imagen recogida por la cámara y permite modificar la manera en la que dicha imagen es presentada (modificando el valor del zoom, contraste o brillo).

En concreto, esta aplicación puede resultar útil para personas que padecen DMAE (Degeneración Macular Asociada a la Edad), una enfermedad que produce una degeneración de la mácula. Las personas que padecen DMAE experiencian como su visión se vuelve menos nítida y el aumento de un campo borroso en el centro de su visión. De tal manera, les resultará complicado interactuar de forma habitual con un dispositivo móvil.

Este proyecto sirve como prueba de concepto de una aplicación de realidad aumentada que permita al usuario interactuar con el sistema mediante gestos, sin necesidad de usar otros métodos de entrada como puedan ser botones y/o comandos de voz.

Palabras clave

Android, MediaPipe, Python, Java, cámara, gesto, reconocimiento, IA

Abstract

Our final degree project consists of developing an Android application that uses the phone's camera to record and recognize hand gestures made by the user in real time. The application displays on screen the image collected by the camera and allows users to modify the way in which the image is presented (modifying the zoom, contrast or brightness).

In particular, this application can be useful for people suffering from AMD (Age-Related Macular Degeneration), a disease that causes degeneration of the macula. People suffering from AMD experience how their vision becomes less sharp and the increase of a blurred field in the center of their vision. As such, they will find it difficult to interact on a regular basis with a mobile device.

This project serves as a proof of concept for an augmented reality application that allows the user to interact with the system through gestures, without the need to use other input methods such as buttons and/or voice commands.

Keywords

Android, MediaPipe, Python, Java, camera, gesture, recognition, AI

Índice de contenidos

1 Introducción 1
1.1. Motivación 1
1.2. Objetivos 1
1.3. Plan de trabajo 3
2. Estado de la cuestión 10
A Smart Watch-based Gesture Recognition System for Assisting People with Visual Impairments 11
Learning Platform for Visually Impaired Children through Artificial Intelligence and Computer Vision 11
Static Hand Gesture Recognition using an Android Device 12
Vision-based Gesture Tracking for Teleoperating Mobile Manipulators 13
Smart Control of Home Appliances Using Hand Gesture Recognition in an IoT-Enabled System 14
MediaPipe Hands 15
OpenPose 16
OpenCV 16
CameraX 16
Chacuopy 17
TensorFlow 17
3. Desarrollo de la aplicación 17
3.1. Gestión de riesgos 17
3.2. Plan de pruebas 22
3.3. Iteraciones del proyecto 25
3.3.1. Fase de investigación 25
3.3.2. Iteración 1 29
3.3.3. Iteración 2 37
3.3.4. Iteración 3 39
4. Resultados 45
4.1. Manual de usuario 45
4.2. Aplicación 50
4.3. Análisis ético 54
4.3.1. Licencias de software 54
4.3.2. Uso de la aplicación: restricciones y soluciones implementadas 55
5. Conclusiones y trabajo futuro 57
Apéndice A - Pruebas primera iteración 84
Apéndice B - Pruebas segunda iteración 88
Apéndice C - Pruebas tercera iteración 91

Índice de figuras

Imagen 1.3.1: Cronograma de la iteración inicial 6
Imagen 1.3.2: Primera parte del cronograma de la iteración 1 7
Imagen 1.3.3: Segunda parte del cronograma de la iteración 1 7
Imagen 1.3.4: Cronograma de la iteración 2 8
Imagen 1.3.5: Cronograma de la iteración 3 9
Imagen 1.3.6: Gráfico de porcentajes de tiempo invertidos en cada tipo de tarea 10
Imagen 3.1.1: Matriz de probabilidades y impacto de los riesgos 18
Imagen 3.3.2.1. Lista de correspondencia entre la versión Android y el de API 29
Imagen 3.3.2.2: Puntos de la mano que usa el objeto de la librería MediaPipe 32
Imagen 3.3.2.3: Resultado visual de la librería MediaPipe con una mano real 32
Imagen 3.3.3.1: Pantalla inicial de la aplicación con el botón de información y pantalla de manual de usuario 38
Imagen 3.3.4.1: Gesto restablecer 41
Imagen 3.3.4.2: Diálogo para pedir los permisos de la cámara en la aplicación Android 42
Imágen 3.3.4.3: Ventana para modificar los ajustes del sistema 43
Imagen 3.3.4.4: Diálogo modal con la explicación de acceso a los permisos 45
Imagen 3.3.4.5: mensaje de no poder editar el brillo en la aplicación 45
Imagen 4.1.1: Gesto Brillo 46
Imagen 4.1.2: Gesto Contraste 47
Imagen 4.1.3: Gesto Zoom 47
Imagen 4.1.4: Gesto Aumentar 48
Imagen 4.1.5: Gesto Disminuir 49
Imagen 4.2.1: Matriz de confusión 52
Imagen 4.2.2 Informe de clasificación 52
Figura 4.2.3 : Porcentaje de acierto de los distintos gestos 54
Figure 1.3.1: Timeline of the initial iteration 63
Figure 1.3.2: Iteration 1 timeline 64
Figure 1.3.3: Timeline for iteration 2 65
Figure 1.3.4: Timeline of iteration 3 66
Figure 1.3.5: Graph of percentages of time spent on each type of task 66

Índice de tablas

Tabla 3.1.1: Identificación de los riesgos 19
Tabla 3.1.2: Descripción de las soluciones de los riesgos 22
Tabla 3.2.1: Plan de pruebas 25
Tabla 4.2.1: Objetivos funcionales 50
Tabla 4.2.2: Objetivos de accesibilidad 51
Tabla 4.2.3: Objetivos éticos 51

1 Introducción

1.1. Motivación

Existe una amplia variedad de programas y aplicaciones que son capaces de identificar gestos, sin embargo, la mayoría de estas herramientas tienen como objetivo principal otros fines distintos al de ayudar a solucionar problemas para personas con discapacidad visual o mejorar su calidad de vida.

Concretamente, este proyecto va enfocado a mejorar el día a día de personas que padecen DMAE (Degeneración Macular Asociada a la Edad), una enfermedad que produce una degeneración de la mácula. En consecuencia, las personas que padecen DMAE experiencian como su visión se vuelve menos nítida y el aumento de un campo borroso en el centro de su visión, por lo que modificar la forma en la que ven el mundo puede otorgarles una gran ayuda a la hora de desenvolverse en sus tareas cotidianas.

Además, consideramos que resultaría sumamente beneficioso y oportuno el uso de gestos como medio de interacción con las gafas, ya que actualmente, las personas con discapacidad visual suelen depender de dispositivos externos para registrar las acciones que desean realizar.

Además, esta iniciativa representa un proyecto altamente interesante, que puede servir como una excelente oportunidad para demostrar el enorme potencial que poseen las tecnologías basadas en inteligencia artificial para resolver necesidades muy específicas y mejorar la calidad de vida de personas con discapacidades.

1.2. Objetivos

Para definir los principales objetivos de este proyecto, distinguiremos entre tres tipos diferenciados: objetivos funcionales (que permitirán que la aplicación reconozca gestos en el menor tiempo posible para realizar la acción asociada al mismo), objetivos de accesibilidad (que permitirán a un usuario de la aplicación interactuar con la misma sin ningún problema) y objetivos éticos (que permitirán al usuario tomar decisiones sobre el uso de la cámara dentro de la aplicación).

En cuanto a los objetivos funcionales encontramos:

- Sistema empotrado sin necesidad de conexión a internet: la aplicación podrá ser usada como una aplicación de las propias gafas de realidad aumentada y el usuario podrá usarla sin necesidad de estar conectado a internet, ya que se trata de una aplicación enfocada a mejorar la accesibilidad de las gafas para personas con problemas de visión, la conectividad no debe restringir su uso.
- El sistema podrá ser usado como aplicación Android en un smartphone y como aplicación Android en las gafas de realidad aumentada. En consecuencia, primero se desarrollará la aplicación Android para conseguir la detección de los gestos y más tarde se trasladará la implementación a las gafas.
- Detección de gestos: la aplicación detectará una serie de gestos que estarán asociados a una característica de la imagen (brillo, zoom y contraste), una vez detectado se podrá aumentar o disminuir dicha característica mediante la realización de otro gesto.
- Tiempo de respuesta: dado que es una aplicación que detecta gestos y realiza acciones en base a esto, el tiempo de respuesta debe ser lo suficientemente bajo como para suponer una mejora sobre la realización de estas acciones de forma manual mediante el uso de un mando. El objetivo de este proyecto es poder detectar un gesto con un tiempo de respuesta máximo de tres segundos.

Con relación a los objetivos de accesibilidad:

- Detección de gestos: asociado al objetivo funcional para detectar gestos y realizar una acción en base al mismo, se indicará de manera clara qué gesto ha sido detectado para que el usuario sepa qué característica de la imágen va a ser modificada. Además, en caso de no poder reconocer el gesto se informará al usuario.
- Manual de usuario: la aplicación contará con un manual de usuario que podrá ser consultado en caso de existir alguna duda con la funcionalidad de la app. Este manual será lo más sencillo posible para que una persona con el mínimo conocimiento en Tecnologías de la información y comunicación (TIC) pueda entender el funcionamiento de la aplicación.
- Narrador: debido a que la aplicación está destinada a personas con problemas de visión, se incorporará un narrador a todas las acciones mencionadas

anteriormente (detección de gestos, errores…) para que una persona ciega pueda usar la aplicación sin ningún problema.

- Personalización del texto: el usuario podrá cambiar tanto la tipografía como el tamaño de letra para poder usar la aplicación sin problemas. Estos ajustes se guardarán para que la próxima vez que abra la aplicación estén disponibles.

Por último, respecto a los objetivos éticos encontramos:

- En relación a la cámara: como la aplicación hace uso de la cámara se permitirá que el usuario pueda revocar el uso de la misma en cualquier momento. Además, no se permitirán hacer capturas de pantalla dentro de la propia aplicación y no se guardará ningún registro de las imágenes usadas para la detección de los gestos.
- En relación a la licencia del software: debido a que se trata de una aplicación de cierta complejidad y con mucha posibilidad de mejora, todo el software creado tendrá una licencia de software libre. De esta manera se podrá tomar como base esta aplicación para poder realizar múltiples aplicaciones con el objetivo de ayudar a personas con problemas de visión.

1.3. Plan de trabajo

Para el desarrollo del proyecto se ha utilizado una metodología iterativa e incremental, el proyecto se divide en varias iteraciones o ciclos cortos de trabajo, obteniendo un progreso gradual en el desarrollo. Cada iteración se divide a su vez en varios procesos: planificación, diseño, desarrollo, pruebas y entrega. Estas iteraciones se repiten hasta finalizar el proyecto.

Las ventajas del uso de esta metodología son principalmente la facilidad para adaptar el desarrollo ante requisitos cambiantes a medida que se obtiene más información, además al contar con entregas continuas permite detectar y corregir errores en fases tempranas del desarrollo lo que se traduce en una mejora de la calidad del producto final.

En cuanto a la gestión del flujo de trabajo, se ha utilizado Kanban[1], una técnica de gestión visual con la que se pretende incrementar la eficiencia y calidad del desarrollo software. Para ello se utiliza un tablero Kanban en el que se representa el flujo de trabajo y el estado de las tareas. Este tablero se divide en varias columnas: "Pendiente", "En desarrollo", "Pruebas" y "Finalizado".

Pendiente: La tarea se encuentra definida pero aún no ha comenzado a desarrollarse, por lo tanto, ningún miembro del equipo tiene esa tarea asignada.

En desarrollo: La tarea ha sido asignada a uno o varios miembros del equipo de desarrollo, la tarea se mantendrá en esta etapa hasta que haya sido completada.

Pruebas: Una vez la tarea ha sido desarrollada debe ser probada antes de darse por finalizada, las pruebas deben ser realizadas siempre por una persona distinta a la que ha realizado el desarrollo de la tarea, tras la realización de las pruebas la tarea puede darse por finalizada o, en caso de que no hay superado las pruebas, se deberá enviar de vuelta al estado “En desarrollo” para proceder a solucionar los fallos encontrados. Para las tareas de investigación la fase de pruebas no es necesaria y se da por finalizada una vez se haya obtenido y documentado la información necesaria.

Finalizado: Al finalizar exitosamente las pruebas en una tarea el estado de esta pasa a finalizado.

Para representar el estado de las diferentes tareas hemos utilizado la herramienta de Asana[2], que permite la creación de tableros Kanban virtuales. Los miembros del equipo de desarrollo pueden desplazar las tareas entre las diferentes columnas a medida que se avanza en las diferentes etapas.

Para el control de versiones la herramienta utilizada ha sido GitHub[3], se creó una rama principal para subir las tareas completadas y adicionalmente, si alguna tarea requería de ciertas pruebas, se habilitaba una nueva rama para esta.

En el caso específico de nuestro proyecto, se ha dividido en 3 iteraciones principales y una fase inicial de investigación previa al desarrollo, para ninguna de ellas se ha determinado una fecha de finalización o entrega a excepción de la última iteración que

debía estar completada en los plazos indicados por la Universidad Complutense de Madrid para la entrega del proyecto.

Durante la fase inicial o fase de investigación, el equipo de desarrollo realizó un estudio de las investigaciones o proyectos existentes sobre la temática principal del proyecto, además de analizar las ventajas e inconvenientes de diferentes tecnologías que podrían ser utilizadas en el desarrollo del mismo. El objetivo de esta fase inicial era definir de una manera más precisa el alcance del proyecto y buscar la tecnología más adecuada para cumplir con los objetivos propuestos.

Fase inicial:

En primera instancia, el equipo de desarrollo se dedicó al estudio de proyectos y aplicaciones existentes similares a nuestro proyecto, para ello se analizaron diferentes estudios relacionados con el uso de inteligencia artificial para ayudar a personas con discapacidad visual. Este estudio ayudó al equipo de desarrollo a conocer las investigaciones pendientes y poder así definir de una forma más precisa los requisitos de la aplicación final.

Una vez conocidas las aplicaciones existentes y teniendo una vista panorámica de las investigaciones pendientes, el siguiente paso sería investigar las tecnologías que podrían ser utilizadas para el desarrollo del proyecto.

Por último, teniendo un listado de tecnologías existentes y sabiendo los pros y contras de cada una se realizaron pequeñas pruebas de funcionalidad en cada una de las tecnologías para decidir cuál de ellas era la óptima para las características del proyecto.

Se puede ver un cronograma de la fase inicial en la imagen 1.3.1.

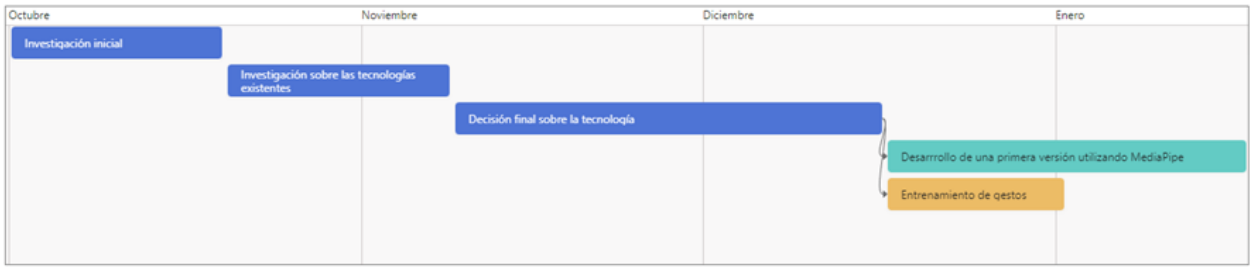

Imagen 1.3.1: Cronograma de la iteración inicial

Iteración 1:

La primera iteración del proyecto tenía como objetivo la finalización de las funcionalidades básicas de la aplicación, entre las que se encontraban la detección de los gestos de zoom, brillo, contraste, aumentar y disminuir, y la funcionalidad de estos, modificando los parámetros necesarios de la cámara.

Esta primera iteración era sin duda la más ambiciosa del proyecto, ya que no solo requería implementar la detección de los gestos descritos anteriormente y la modificación de la configuración de la cámara, si no que, al ser nuestra primera toma de contacto con el proyecto, fue necesario realizar una investigación paralela al desarrollo que provocó que esta iteración fuera también la más extendida en el tiempo. Pero permitió asentar una base funcional sólida sobre la que desarrollar las siguientes iteraciones.

Para dar por finalizada la iteración se realizaron las pruebas requeridas y se corrigieron errores que surgieron de las mismas.

Se puede ver un cronograma de la iteración 1 en las imágenes 1.3.2 y 1.3.3

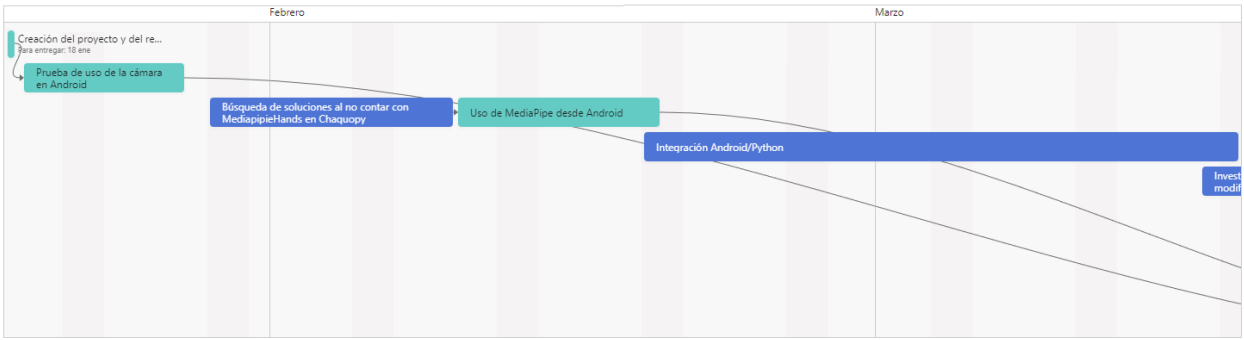

Imagen 1.3.2: Primera parte del cronograma de la iteración 1

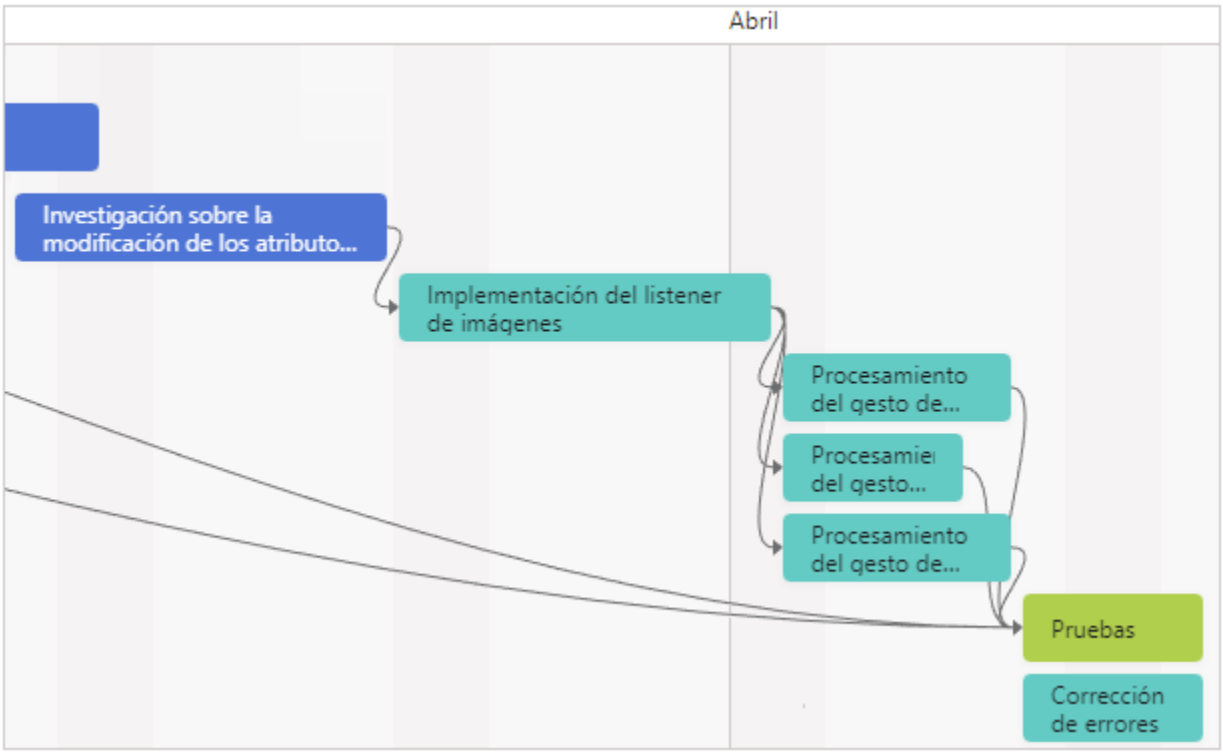

Imagen 1.3.3: Segunda parte del cronograma de la iteración 1

Iteración 2:

El objetivo para esta iteración era el desarrollo de dos nuevas funcionalidades, el acceso a un manual de usuario desde la propia aplicación y la implementación del

narrador de voz. Ambas funcionalidades se implementaron de forma paralela y se añadió una funcionalidad para evitar el bloqueo de la pantalla mientras la aplicación está en uso. Una vez finalizadas se añadió una nueva funcionalidad que guardaba la configuración actual al cerrar la aplicación, de esta manera se cargarían los parámetros guardados al reiniciar la aplicación.

Por último, se realizaron todas las pruebas necesarias y se solucionaron los errores derivados de las mismas.

Se puede ver un cronograma de la iteración 2 en la imagen 1.3.3

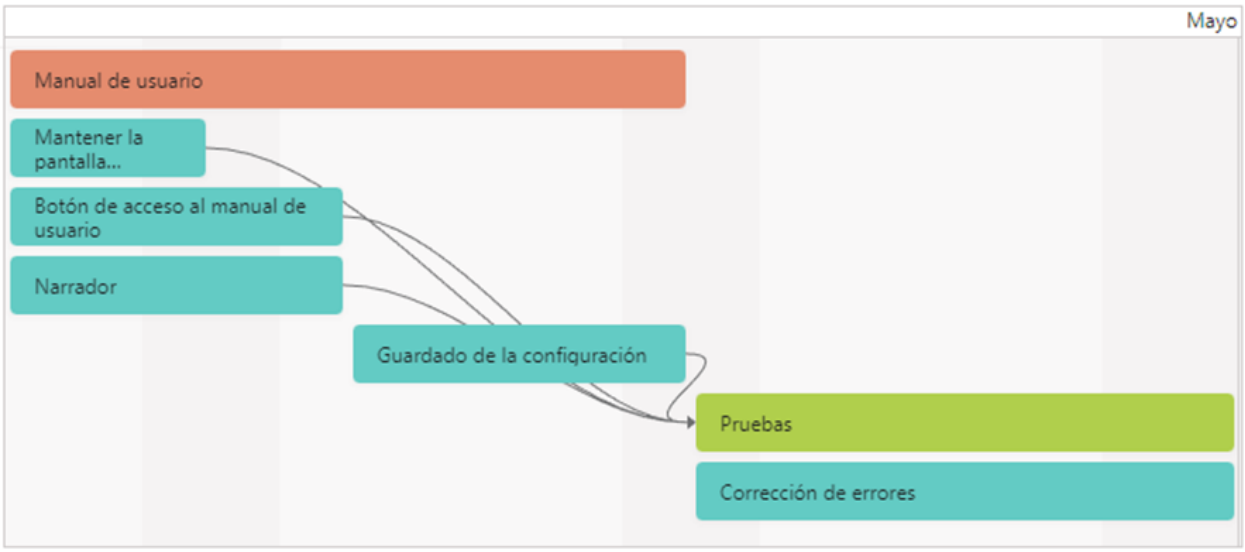

Imagen 1.3.4: Cronograma de la iteración 2

Iteración 3:

El objetivo de la tercera y última iteración consistió en implementar una serie de advertencias y limitaciones con el objetivo de garantizar la máxima integridad ética de la aplicación. Además, se incluyó un nuevo gesto que permitía reiniciar la configuración a los valores definidos por defecto, añadir este nuevo gesto no supuso un gran esfuerzo ya que la aplicación había sido pensada para ser escalable en cuanto a la detección de nuevos gestos.

Una vez finalizado el desarrollo se realizaron las pruebas asociadas y se corrigieron los errores encontrados

Se puede ver un cronograma de la iteración 3 en la imagen 1.3.4.

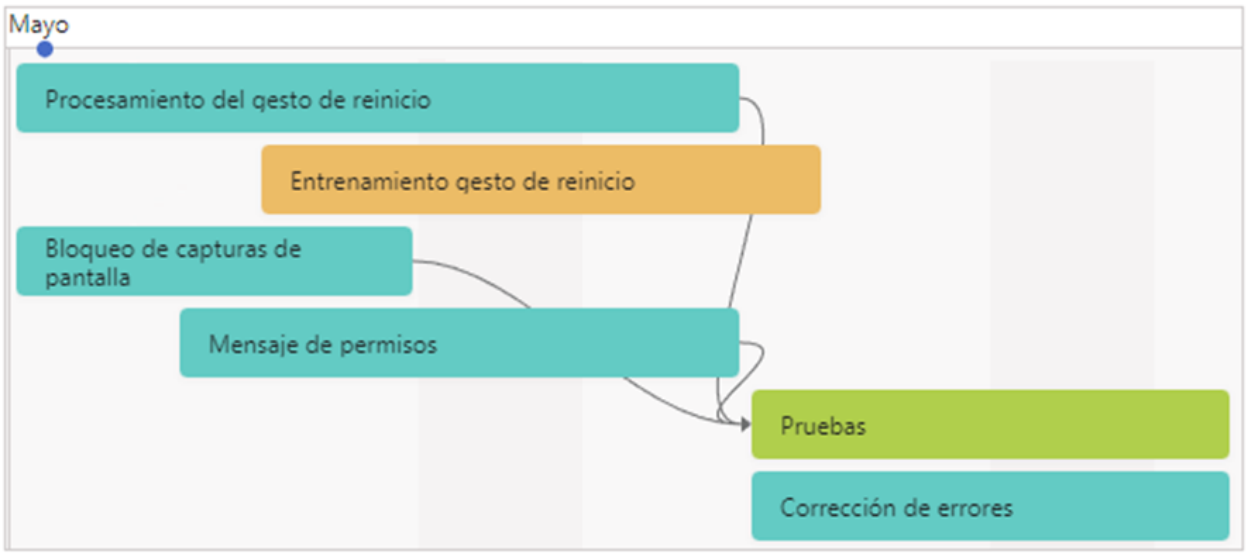

Imagen 1.3.5: Cronograma de la iteración 3

Por último, se ha creado un diagrama circular que representa el porcentaje de tiempo que se ha dedicado a cada tipo de tarea, esto se refleja en la imágen 1.3.5.

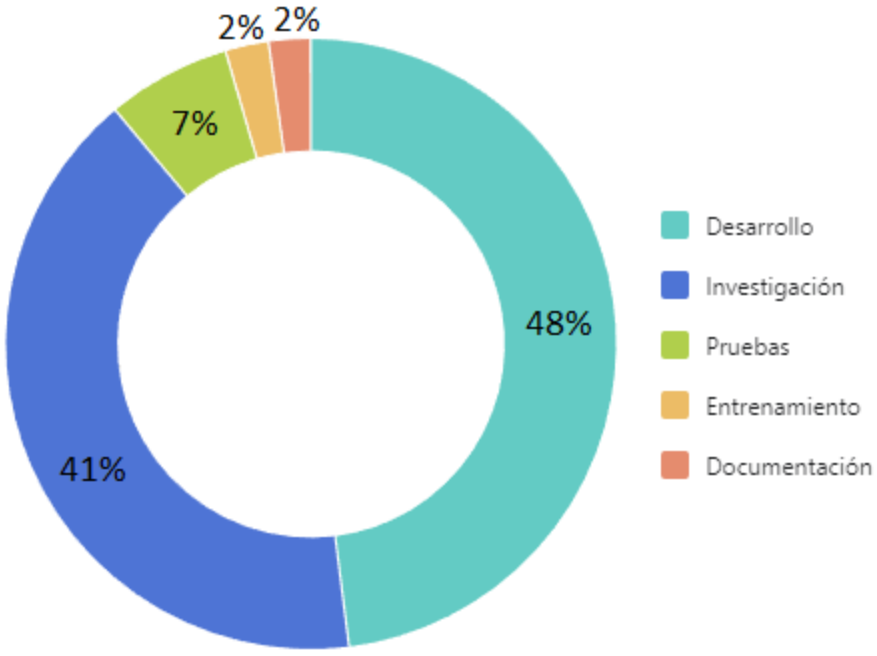

Imagen 1.3.6: Gráfico de porcentajes de tiempo invertidos en cada tipo de tarea

2. Estado de la cuestión

Con el fin de poder estudiar otros trabajos relacionados con la materia, se realizó una investigación con dos objetivos principales, por un lado, encontrar soluciones similares para personas con problemas de visión y, por otro, proyectos que implementaran tecnologías que pudieran ser de utilidad para la detección de gestos.

Esta primera aproximación sería de utilidad a la hora de concretar los objetivos del proyecto, diferenciándose de cualquier otro existente, además de conocer las librerías más importantes en la detección de gestos y facilitar la elección de las más apropiadas para el proyecto.

En primer lugar, se explican los proyectos que se encontraron en relación al objetivo principal de este proyecto, ayudar a personas con problemas de visión:

A Smart Watch-based Gesture Recognition System for Assisting People with Visual Impairments

Este proyecto[4] nace de los inconvenientes que surgen cuando una persona con problemas de visión debe utilizar las funcionalidades de un smartphone. Actualmente, la solución alcanzada para las personas que no pueden usar la interacción táctil de estos smartphones es interactuar a través de comandos de voz, sin embargo, tal como se explica en este proyecto esta solución puede no ser la más adecuada para todos los entornos (por ejemplo, entornos con mucho ruido).

Es por esto que en este proyecto se plantea la posibilidad de usar un smartwatch junto con un smartphone (mediante conexión Bluetooth), de tal manera que el smartwatch se encargue de detectar la acción que quiere llevar a cabo el usuario (mediante la realización de un gesto específico) y el smartphone de efectuarla.

Para realizar la detección de gestos se utiliza la información obtenida del acelerómetro del smartwatch, que es posteriormente enviada al smartphone y este. Usando la imágen capturada mediante distintas técnicas de clasificación de Support Vector Machine (SVM), será capaz de reconocer objetos relevantes como pueden ser señales de suelo mojado o distintos logos.

El fin del proyecto es similar al objetivo principal de nuestro trabajo, ya que, mediante el reconocimiento de gestos, es capaz de realizar acciones que ayudan a personas con problemas de visión. Sin embargo, en este proyecto la detección de gestos supone el uso de un dispositivo hardware adicional, lo que requiere una conexión entre los dos dispositivos que puede llegar a causar problemas al tratar de realizar un escalado de la aplicación, además de requerir una lógica mucho más compleja.

Learning Platform for Visually Impaired Children through Artificial Intelligence and Computer Vision

Este trabajo[5] consiste en una aplicación enfocada a niños con problemas de visión, su objetivo principal es ayudarles a comprender los objetos que hay a su alrededor. Para

ello, es capaz de identificar objetos y en función de los gestos del usuario un narrador lee el nombre del objeto seleccionado.

Este proyecto hace uso de una Red Convolucional para lograr la detección tanto de los gestos del usuario como de los objetos que le rodean, obteniendo un resultado sumamente positivo.

Es importante destacar que se realiza una diferenciación entre entornos al aire libre y cerrados, esto se debe a que los entornos cerrados pueden tratarse de un entorno mucho más complicado para la detección de gestos y objetos, ya que pueden ser lugares de baja luminosidad. Cuando la aplicación se utiliza en exteriores, el usuario podrá indicar el objeto apuntándole, mientras que, en un entorno interior, deberá agarrarlo.

Este proyecto guarda relación con nuestro proyecto ya que se utiliza para facilitar el día a día de personas con problemas de visión, el proyecto se enfoca concretamente en la educación de niños con problemas de visión. Es interesante ya que ayuda a entender otro punto de vista a tener en cuenta para nuestro proyecto, la dificultad de detectar gestos mediante una cámara en entornos de baja luminosidad y la posibilidad de diferenciar los gestos en función de si el usuario se encuentra en espacio interior o exterior.

Los trabajos listados a continuación tienen objetivos técnicos similares al de este proyecto, pero no se centran en sustentar ayuda a personas con problemas de visión:

Static Hand Gesture Recognition using an Android Device

El primer trabajo[6] que se encontró relacionado con la temática principal del proyecto fue un estudio relacionado con la detección de gestos estáticos en Android. En este proyecto se realiza una predicción de gestos en base una imagen mediante tres fases: preprocesamiento, extracción de características y clasificación.

En la primera etapa se realiza un preprocesamiento de la imagen original para que al realizar el entrenamiento y predicción se minimicen aquellas características que no sean necesarias (como puede ser la iluminación o el ángulo desde el que se ha tomado la

foto), esto se consigue transformando la imagen a una escala de grises, realizando una reorientación de la imagen en base a la posición de la mano, cortando todos los elementos innecesarios para quedarse solo con la imagen de la mano y por último normalizando los datos de la imagen.

La segunda etapa está basada en la técnica Principal Component Analysis (PCA) que en base a una imagen obtiene sólo las características principales y necesarias para el entrenamiento y la clasificación. Esta técnica hace que las imágenes pesen mucho menos optimizando así la aplicación Android.

Por último, se realiza la clasificación de la imagen que se había procesado en las etapas anteriores. Para esta clasificación se usa un modelo de clasificación conocido como Support Vector Machines (SVM) combinado con el modelo de árboles de decisión. Usando estos dos métodos obtiene un porcentaje de precisión muy elevado y consigue detectar los 5 gestos con los que se ha entrenado el sistema usando las imágenes obtenidas con un dispositivo Android.

Este proyecto es muy interesante ya que explica de forma bastante resumida como se ha realizado el proceso de clasificación de los gestos mediante una imágen, sin embargo, nuestro proyecto no tiene cómo objetivo sólo el reconocimiento de gestos, sino el uso de este reconocimiento como vía para realizar ciertas acciones que cambien las características de la cámara en tiempo real.

Vision-based Gesture Tracking for Teleoperating Mobile Manipulators

En este trabajo[7] se realiza un estudio de la viabilidad de la librería MediaPipe para el reconocimiento de gestos. Esta aplicación se centra en un objetivo muy distinto al de nuestro proyecto (ya que trata de poder asociar acciones de un robot a gestos para reducir el coste de teleoperación) pero aun así sirve como ejemplo para ver el uso de esta librería.

Para asociar los landmarks que usa MediaPipe a un gesto concreto se utiliza una red neuronal básica con tres capas. Cabe destacar que esta aplicación tiene un rendimiento de 15 fotogramas por segundo en la simulación que se realiza, aunque estiman un mayor rendimiento fuera de la misma, esto en nuestro proyecto debe tenerse muy en cuenta y mejorarse todo lo posible. En referencia a este rendimiento también hay que

tener en cuenta que se está realizando en un ordenador y no en un móvil por lo que dependiendo de la implementación de esta librería para distintos lenguajes se podrán tener problemas futuros en la optimización de la aplicación.

Aun así, parece que esta librería da un resultado muy bueno en la detección de las manos y que es un método fiable para la detección de gestos por lo que para el desarrollo de nuestro proyecto se establece esta librería como un eje fundamental.

Smart Control of Home Appliances Using Hand Gesture Recognition in an IoT-Enabled System

Este proyecto[8], al igual que el que se ha explicado anteriormente, también usa la librería MediaPipe para la detección de gestos en una aplicación Android. Al igual que la anterior el objetivo de este proyecto es conectar una aplicación Android con un sistema encargado de realizar cambios en un sistema, en este caso, cambiar el estado de unos leds: si está encendido o no y el color asociado, sin embargo, difiere con el nuestro. Aun así, muestra la conexión entre la aplicación Android donde se detectan los gestos mediante el uso de MediaPipe y un hardware encargado de realizar las acciones requeridas mediante wifi.

Aunque la idea principal de nuestra aplicación sea un sistema empotrado sin ningún tipo de requerimiento de conexión, este trabajo aproxima una posible implementación que quizás podría tenerse en cuenta en caso de que no se puedan cumplir los objetivos principales.

El sistema está formado por 4 módulos:

Los dos primeros módulos se encuentran en la aplicación Java. El primero de estos módulos es el encargado de capturar las imágenes de la cámara y enviar cada fotograma capturado al segundo módulo, este segundo módulo recibe y analiza las imágenes utilizando la librería Mediapipe con el objetivo de detectar una mano, en cuyo caso se envía la matriz generada con los landmarks de la mano al tercer módulo, situado en la aplicación Python, en el cual haciendo uso de CNN y la librería TensorFlow se asocia la matriz de los landmarks con el identificador de uno de los gestos existentes

o en caso de no superar el porcentaje de acierto mínimo establecido, se asocia al identificador de gesto no detectado, este identificador se envía al cuarto y último módulo, situado nuevamente en la aplicación Java, donde se realiza la acción asociada al gesto.

Para la implementación de estos módulos y en base a lo estudiado en los proyectos explicados anteriormente se han decidido usar las siguientes librerías:

MediaPipe Hands

MediaPipe hands[9] es una biblioteca de código abierto desarrollada por Google que utiliza técnicas de aprendizaje automático para generar a partir de una imagen de una mano 21 puntos de referencia (landmarks).

MediaPipe se destaca de otros enfoques existentes actualmente ya que permite un rendimiento a tiempo real en dispositivos móviles e incluso se adapta a varias manos de manera simultánea.

MediaPipe utiliza una canalización de Machine Learning que se compone por varios modelos que trabajan conjuntamente:

- Modelo de detección de manos: Este modelo se utiliza para detectar la presencia de manos en una imagen o video. Para ello se utiliza una red neuronal convolucional (CNN) con la que es posible predecir la probabilidad de que haya una mano en la imagen. El modelo ha sido previamente entrenado con un gran conjunto de imágenes que contienen manos.
- Modelo de landmarks: Este modelo se utiliza para localizar los puntos clave de cada mano en la imagen. El modelo utiliza una red neuronal convolucional para predecir la ubicación de 21 puntos clave en cada mano, conocidos como landmarks, que se utilizan para rastrear y estimar la posición de las manos en tiempo real.
- Modelo de seguimiento de la mano: Este modelo se utiliza para realizar un seguimiento de las manos en tiempo real a medida que se mueven en el video. El modelo utiliza los landmarks de la mano detectados por el modelo anterior como entrada y utiliza técnicas de seguimiento óptico para estimar la posición y orientación de las manos en el cuadro actual.

- Modelo de filtrado de landmarks: Este modelo se utiliza para suavizar y estabilizar la posición y orientación de las manos estimadas por el modelo de seguimiento de la mano. El modelo utiliza un filtro de Kalman para estimar la posición y velocidad de las manos y reducir el ruido y las fluctuaciones en las estimaciones.
- Filtro de Kalman[10]: es un algoritmo matemático que sirve para identificar un sistema dinámico lineal. Al tratarse de un algoritmo recursivo puede utilizarse en tiempo real utilizando únicamente la entrada actual, el estado previo y su matriz de incertidumbre.

Esta librería se usa en varios de los proyectos que se han encontrado relacionados con la detección de gestos por lo que es la principal herramienta que se usará para esto.

OpenPose

Esta librería[11] es parecida a MediaPipe pero está enfocada principalmente en la detección y seguimiento de cuerpos en tiempo real. También permite la integración con distintas herramientas de procesamiento de imágenes por lo que puede resultar muy útil en este proyecto. Aun así, al no centrarse en la detección de manos puede tener un peor rendimiento que MediaPipe, se estudiarán ambas a la hora de crear el proyecto y se decidirá cuál es la mejor.

OpenCV

OpenCV[12] es una biblioteca de software libre y código abierto que se utiliza para el desarrollo de procesamiento de imágenes en tiempo real. Para ello utiliza algoritmos de procesamiento de imagen y técnicas de aprendizaje automático para analizar y procesar imágenes y videos.

Esta librería se puede usar en distintos sistemas operativos usando varios lenguajes por lo que puede ser útil en el procesamiento de la imagen en el proyecto.

CameraX

CameraX[13] permite acceder y modificar características avanzadas de la cámara, como pueden ser el control de la exposición o el zoom.

Es una librería de Android que se podrá usar para la gestión de las acciones (zoom, contraste y brillo) asociadas a cada gesto.

Chacuopy

Chacuopy[14] es una interfaz de lenguaje Java/Python, que permite acceder desde Java a Python y viceversa. Permite realizar una integración completa mediante el sistema de compilación Gradle estándar de Android Studio.

Incluye además una gran variedad de paquetes Python de terceros como OpenCV o TensorFlow.

Esta librería nos permitirá conectar una aplicación Python encargada de la Inteligencia Artificial para la detección de gestos con la aplicación Android.

TensorFlow

TensorFlow[15] es una librería de código libre utilizada en proyectos de aprendizaje automático.

Es utilizada por MediaPipe Hands para detectar y rastrear las manos en tiempo real.

3. Desarrollo de la aplicación

A continuación, se analiza el proceso de creación de la aplicación Android. Explicando el proceso previo de gestión de riesgos, planificación del plan de pruebas e investigación, además del proceso realizado para el desarrollo del código de la aplicación.

3.1. Gestión de riesgos

Para la gestión de riesgos se ha realizado un estudio clásico en el desarrollo del software basándose en una tabla de relaciones entre probabilidad de que ocurra el riesgo y la gravedad del mismo, tal como se puede observar en la imagen 3.1.1. En esta tabla se van a clasificar tres tipos distintos de riesgo: alto, medio y bajo.

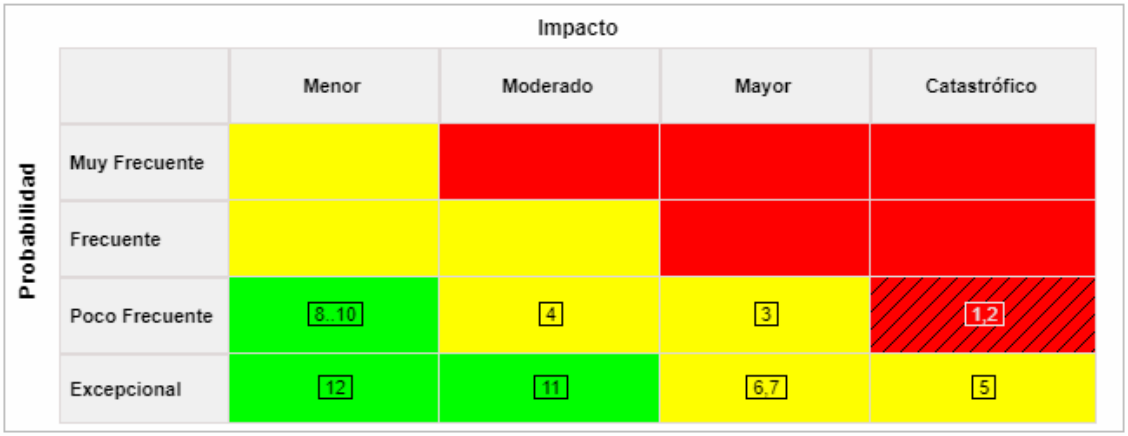

Imagen 3.1.1: Matriz de probabilidades y impacto de los riesgos[16]

Además de esta clasificación en los distintos tipos de riesgos se ha añadido una descripción del mismo. Aunque en un principio se iban a detallar sólo aquellos riesgos clasificados como riesgo alto o medio, dado que obtuvimos un número bastante limitado y este proyecto tiene una gran importancia para todos los integrantes del equipo se decidieron analizar todos, tratando de solucionar cualquier problema que surgiera en el transcurso del proyecto. La tabla 3.1.1 muestra el estudio de los riesgos asociados a este trabajo con un identificador único y en la tabla 3.1.2 asocia cada uno de estos riesgos a dos tipos de soluciones: una solución para prevenir el riesgo y otra para actuar en caso de que el riesgo se haga realidad.

ID	DESCRIPCIÓN DEL RIESGO	PROBABILIDAD	IMPACTO
1	Uno de los integrantes del trabajo de fin de grado deja el proyecto debido a la imposibilidad para compaginarlo con otras asignaturas, haciendo que haya que volver a planificar el proyecto y dar una mayor carga de trabajo a los demás integrantes del grupo.	Excepcional	Mayor
2	Existe una falta de información con la que documentarse para desarrollar las distintas funcionalidades del proyecto, haciendo que las fases de investigación duren más tiempo y comprometiendo la entrega del proyecto en la primera	Poco Frecuente	Catastrófico

	convocatoria.		
3	La falta de experiencia de un estudiante al desarrollar una actividad concreta hace que no se desarrolle a tiempo y retrase la planificación.	Excepcional	Moderado
4	Pérdida del código desarrollado debido a un fallo en la plataforma de control de versiones usada, haciendo que se pierda todo aquello que no se encuentre almacenado de forma local, retrasando el proyecto debido a la necesidad de rehacer las partes de código que se hayan perdido.	Excepcional	Mayor
5	Pérdida del código desarrollado debido a un error al combinar dos subidas paralelas al repositorio, provocando un retraso en la planificación debido al tiempo empleado en recuperar el código (este tiempo aumentará con la cantidad de subidas que se realicen sobre el repositorio sin advertir dicho error).	Excepcional	Mayor
6	Al desarrollar la aplicación utilizando las distintas opciones investigadas se encuentra una incompatibilidad y es necesaria la restructuración del proyecto de manera íntegra, comprometiendo la fecha de entrega propuesta.	Excepcional	Catastrófico
7	Al desarrollar la aplicación nos damos cuenta de que una de las funcionalidades no se puede realizar sin usar Internet, comprometiendo uno de los objetivos principales de la aplicación, prescindir de esta herramienta, haciendo necesario buscar una alternativa o desestimar esta característica para la entrega.	Poco Frecuente	Mayor
8	La incompatibilidad de horarios de los estudiantes hace que sea muy complicado realizar tantas reuniones como se deberían, haciendo que cualquier duda en la implementación o nueva idea para mejorar la aplicación se alarguen en el tiempo más de lo debido, comprometiendo así la calidad del software producido.	Poco Frecuente	Moderado

Tabla 3.1.1: Identificación de los riesgos

ID	RESPUESTA ANTE EL RIESGO
1	**Prevención del riesgo:** en todas las reuniones internas que se realicen se preguntará a cada miembro su situación actual con respecto al proyecto para poder ajustar la carga de trabajo de cada uno, de esta manera se intentará que se pueda compatibilizar casi siempre este proyecto con otras asignaturas manteniendo la motivación de todos los integrantes. Además, desde el principio del proyecto se invitará a convocar reuniones para avisar de cualquier problema relacionado con esto por medio de cualquiera de los canales de comunicación disponibles, haciendo que en caso de que el riesgo se vuelva real los demás integrantes tengan suficiente tiempo como para poder planificar el proyecto en base a esto. **Protocolo si el riesgo se vuelve real:** se realizará una reunión con los tutores para debatir acerca de la viabilidad del proyecto y si los demás integrantes del grupo pueden asumir la carga de trabajo añadida. En caso de poder asumir la carga de trabajo se seguirá la planificación inicial asignando las tareas de la persona que ha abandonado el proyecto a otra persona y en caso de que no se pueda asumir este riesgo se planteará el entregar el proyecto en otra convocatoria o buscar una solución alternativa (como reducir los requisitos funcionales del proyecto).
2	**Prevención del riesgo:** desde el inicio del proyecto se va a realizar un control sobre las fuentes que se están usando para la investigación de los distintos módulos de tal manera que si uno de los integrantes del proyecto está utilizando, por ejemplo, la documentación oficial de una librería concreta, otro estudiante busque otra fuente de información para luego contrastarlas y hacer que este proceso abarque tantas fuentes como sea posible. **Protocolo si el riesgo se vuelve real:** en caso de encontrarse con algún problema de este tipo se planteará una reunión con los tutores para buscar una solución alternativa a dicha funcionalidad.

3	**Prevención del riesgo:** cada vez que se repartan tareas se preguntará a cada miembro acerca de sus preferencias y este será uno de los factores determinantes a la hora de asignarlas. De esta manera se obtendrá un mayor rendimiento de cada uno de los integrantes del proyecto ya que estarán trabajando sobre una funcionalidad conocida. En caso de que ninguno de los integrantes del grupo tenga afinidad por una tarea concreta se incitará a que el estudiante al que se le asigne la tarea a que convoque reuniones cada vez que sea necesario para poder solucionar cualquier problema que pueda surgir durante el desarrollo. **Protocolo si el riesgo se vuelve real:** en caso de encontrarse con una tarea que está llevando más tiempo del esperado se realizará una reunión para que el estudiante encargado de la misma explique el porqué del retraso y pueda debatirse la asignación de dicha tarea a otra persona o asignar más personas al desarrollo de la misma.
4	**Prevención del riesgo:** cada uno de los integrantes del proyecto tendrá una copia local de la rama en la que esté trabajando, aunque no se realice una subida de los cambios o se esté trabajando en una tarea distinta se realizará una copia de esta versión de forma local cada dos días. De esta manera aunque se produzca un problema con la plataforma usada se podrá recuperar el proyecto de manera sencilla. **Protocolo si el riesgo se vuelve real:** dependiendo de la cantidad de código que se haya perdido se planteará una reunión con los tutores para reorganizar el proyecto y valorar si se puede continuar con la entrega manteniendo las mismas funcionalidades para la primera convocatoria o, debido al tiempo, se debería retrasar la entrega a la próxima convocatoria.
5	**Prevención del riesgo:** cada vez que se realice una subida al repositorio se revisará cada uno de los ficheros que se han cambiado para comprobar que sólo se están subiendo los cambios pertinentes y no entra en conflicto con otra funcionalidad. Además se realizarán pruebas cada vez que se termine un hito para comprobar que todo lo que se ha implementado funciona correctamente. También se intentará que al repartir el trabajo se minimice la necesidad de que varias personas trabajen en una misma funcionalidad para evitar cualquier conflicto (en caso de que sea posible). **Protocolo si el riesgo se vuelve real:** dependiendo de la cantidad de código que se haya perdido se planteará una reunión con los tutores para reorganizar el proyecto y valorar si se puede continuar con la entrega manteniendo las mismas funcionalidades para la primera convocatoria o, debido al tiempo, se debería retrasar la entrega a la próxima convocatoria.

6	**Prevención del riesgo:** cada vez que se use una nueva librería se intentará realizar una pequeña prueba de su funcionamiento en una clase independiente. De esta manera además de corroborar que funciona como se espera, existirá un código de ejemplo accesible desde nuestro propio proyecto que agilizará el uso de dicha librería. **Protocolo si el riesgo se vuelve real:** se propondrá una reunión con los tutores para explicar el problema y se estudiará la viabilidad del proyecto para realizar la entrega en la primera convocatoria.
7	**Prevención del riesgo:** Todas las pruebas que se realicen de la aplicación se realizarán en un dispositivo sin conexión a internet para corroborar el buen funcionamiento de la aplicación sin hacer uso de esta característica. **Protocolo si el riesgo se vuelve real:** se propondrá una reunión con los tutores para explicar el problema y se estudiará la viabilidad de buscar una solución sin conexión para esta funcionalidad (teniendo en cuenta los plazos del proyecto) o descartar dicha funcionalidad.
8	**Prevención del riesgo:** pese a tener un horario fijo de reuniones se habilitarán varios medios de comunicación (*Discord* y *WhatsApp*) para informar de cualquier cambio necesario en dichas reuniones, de esta manera podremos tener un horario flexible, facilitando así la comunicación entre los miembros del grupo. **Protocolo si el riesgo se vuelve real:** se realizará una revisión de la aplicación para ver si realmente el proyecto cumple con los estándares de calidad propuestos y en caso negativo se planteará la posibilidad de retrasar el proyecto para otra convocatoria.

Tabla 3.1.2: Descripción de las soluciones de los riesgos

3.2. Plan de pruebas

Para poder probar los desarrollos de cada iteración se generó un plan de pruebas, reflejado en la tabla 3.2.1. Esta tabla muestra el identificador asociado a la prueba junto con un nombre, el grupo al que pertenece y una descripción. La ejecución de las mismas y los resultados están asociados a cada una de las iteraciones por lo que se explicarán en las secciones correspondientes. Algunas de estas pruebas no se pueden realizar en todas las iteraciones ya que reflejan implementaciones asociadas a una iteración al completo. Dichas pruebas se han añadido posteriormente y no pertenecen al plan de pruebas original, pero se han añadido a esta sección para que queden

reflejadas todas las pruebas realizadas. Además, cabe destacar que en cada iteración se vuelven a realizar todas las pruebas aunque correspondan a una funcionalidad implementada en otra iteración. De esta manera se asegura que el desarrollo de las nuevas funcionalidades no afecte de forma negativa a las funcionalidades ya desarrolladas.

ID	NOMBRE DE LA PRUEBA	GRUPO	DESCRIPCIÓN DE LA PRUEBA
1	Gesto de zoom	Detección	Se probará que el gesto se reconoce correctamente. Para ello se pondrá el gesto asociado al zoom veinte veces y deberá reconocerlo de manera satisfactoria al menos en un 90% de los casos, es decir, en 18 de las 20 veces que se pruebe. Se considera como una detección correcta que la etiqueta de texto asociada al gesto indique de manera correcta la acción realizada y el narrador lea este resultado.
2	Gesto de brillo	Detección	
3	Gesto de contraste	Detección	
4	Gesto de aumentar	Detección	
5	Gesto de disminuir	Detección	
6	Gesto de resetear	Detección	
7	Aumentar zoom	Acción	Una vez detectado el gesto, se usa el gesto asociado a aumentar y la característica asociada al gesto aumenta.
8	Aumentar brillo	Acción	
9	Aumentar contraste	Acción	
10	Disminuir zoom	Acción	Una vez detectado el gesto, se usa el gesto asociado a disminuir y la característica asociada al gesto disminuye.
11	Disminuir brillo	Acción	
12	Disminuir contraste	Acción	
13	Aumentar zoom más del máximo	Acción	Una vez detectado el gesto, se usa el gesto asociado a aumentar y la característica asociada no aumenta, mostrando un mensaje de error que también será leído por el narrador.
14	Aumentar brillo más del máximo	Acción	
15	Aumentar contraste más del máximo	Acción	
16	Disminuir brillo más del mínimo	Acción	Una vez detectado el gesto, se usa el gesto asociado a disminuir y la característica asociada no disminuye, mostrando un mensaje de error que también será leído por el narrador.
17	Disminuir zoom más del mínimo	Acción	
18	Disminuir contraste más del mínimo	Acción	

19	Resetear la configuración completa	Acción	Una vez detectado el gesto asociado a resetear, teniendo modificado el zoom, el contraste y el brillo se inicializan todos estos parámetros a su valor original.
20	Resetear la configuración (zoom)	Acción	Una vez detectado el gesto asociado a resetear, teniendo modificado el zoom se inicializan a su valor original.
21	Resetear la configuración (brillo)	Acción	Una vez detectado el gesto asociado a resetear, teniendo modificado el brillo se inicializan a su valor original.
22	Resetear la configuración (contraste)	Acción	Una vez detectado el gesto asociado a resetear, teniendo modificado el contraste se inicializan a su valor original.
23	Guardar configuración anterior (todos los parámetros)	Guardado	Si se cierra la aplicación teniendo modificado el zoom, contraste y brillo y se vuelve a abrir se cargan todos estos valores modificados.
24	Guardar configuración anterior (zoom)	Guardado	Si se cierra la aplicación teniendo modificado el zoom y se vuelve a abrir se carga el valor de zoom modificado.
25	Guardar configuración anterior (contraste)	Guardado	Si se cierra la aplicación teniendo modificado el contraste y se vuelve a abrir se carga el valor del contraste modificado.
26	Guardar configuración anterior (brillo)	Guardado	Si se cierra la aplicación teniendo modificado el brillo y se vuelve a abrir se carga el valor del brillo modificado.
27	Modal permiso de cámara	Permisos	Al iniciar por primera vez la aplicación aparece un modal para pedir permisos para usar la cámara.
28	Modal permiso brillo	Permisos	Al iniciar por primera vez la aplicación aparece un modal para poder modificar los ajustes del sistema.
29	Rechazo permiso cámara	Permisos	Si no se concede el permiso a la cámara la aplicación no ejecuta la misma.
30	Rechazo permiso brillo	Permisos	Si no se concede el permiso a los ajustes del sistema la aplicación funciona de manera normal, pero si se intenta modificar el brillo se obtendrá un mensaje de error que además también es leído por el narrador.
31	Captura de pantalla	Permisos	Al intentar realizar una captura de pantalla la aplicación muestra un mensaje de que no se permite realizar esta acción.

32	Grabar pantalla	Permisos	Al intentar grabar la pantalla de la aplicación se muestra un mensaje de que no se permite realizar esta acción.
33	Bloqueo automático	Interacción	El dispositivo no se bloquea de forma automática, aunque pase el tiempo de inactividad establecido.
34	Rotar pantalla	Interacción	La aplicación funciona de manera correcta al rotar la pantalla.
35	Manual de usuario	Manual	Al pulsar el botón de la pantalla principal se redirige a la pantalla del manual de usuario donde se muestra el mismo de manera correcta. Si se pulsa sobre el botón de atrás se redirigirá de forma correcta a la pantalla principal de nuevo.

Tabla 3.2.1: Plan de pruebas

Tal como vemos en la tabla, todas las pruebas se han realizado de manera manual y no se ha usado ningún sistema para automatizarlas. Esto se debe a que toda la funcionalidad de la aplicación está asociada a la cámara y la detección de gestos por lo que sería mucho más complejo y requeriría mucho más tiempo implementar pruebas automáticas que realizarlas de manera manual.

3.3. Iteraciones del proyecto

3.3.1. Fase de investigación

En la fase de investigación nos centramos en el reconocimiento de gestos.

Estuvimos investigando sobre la librería OpenPose, que se puede utilizar para el reconocimiento de gestos, además tiene otras utilidades como rastrear o detectar cuerpos humanos a través de imágenes y vídeos. Esta biblioteca destaca por su precisión a la hora de realizar la detección de manos y los movimientos que se realizan. Además, ofrece una alta compatibilidad con muchos lenguajes de programación como C++, Python, Matlab y Java. Utiliza algunos marcos de aprendizaje profundo como TensorFlow lo que permite una integración fácil y rápida en proyectos que utilizan estos marcos de trabajo.

El principal problema derivado del uso de esta librería es el requerimiento de un alto costo computacional, una gran cantidad de recursos para el procesamiento de la información, por lo que sería necesario un hardware muy potente para poder utilizar esa librería obteniendo un rendimiento adecuado. Teniendo en cuenta que el entorno de ejecución de nuestra aplicación es un sistema operativo Android, el uso de esta librería podría acarrear problemas de rendimiento. Además, para entrenar esta librería sería necesario disponer de una gran cantidad de datos de prueba, lo que conllevaría un mayor tiempo dedicado al entrenamiento. Además, podría implicar problemas en la detección de gestos en escenarios de baja iluminación o con un fondo poco distinguible de la mano.

Durante la investigación de la librería MediaPipe se comprobó su capacidad de análisis y procesamiento de datos multimedia en casos como la detección de manos, seguimiento de gestos y signos realizados. Esta librería al igual que OpenPose cuenta con un alto porcentaje de acierto en la detección de manos y gestos, permite una amplia personalización y es posible la inclusión de gestos y casos de prueba propios. Es compatible con lenguajes de programación como C++, Python, Java y Javascript. También utiliza marcos de aprendizaje profundo como TensorFlow.

Finalmente se decidió utilizar MediaPipe para el desarrollo del proyecto, ya que además de ahorrar una gran cantidad de recursos con respecto a OpenPose, permitía una mayor adaptabilidad a los requisitos de nuestro proyecto, contaba con una gran cantidad de documentación y una amplia variedad de proyectos que hacían uso de la misma y además no requería de una conexión a Internet para su uso.

Tras realizar una investigación sobre las diversas opciones de lenguaje que permitiera programar una inteligencia artificial, se consideró Python la opción más factible, debido la flexibilidad del lenguaje, facilidad de uso y la disponibilidad de una gran variedad de bibliotecas de código abierto existentes, muchas de ellas relacionadas con el desarrollo de IAs. Además, es un lenguaje con el que todos los integrantes del proyecto están familiarizados y por lo que se preveía un breve periodo de adaptación.

Para el desarrollo de la aplicación Android se optó por una implementación Java debido a su popularidad en el uso de aplicaciones móviles. Otra alternativa era hacer uso de

Kotlin, opción que fue descartada debido a la nula experiencia de los miembros del proyecto.

Considerando que era necesario el uso de un narrador encargado de leer al usuario la información clave de la aplicación, se valoraron las diferentes librerías capaces de realizar esta funcionalidad en Android. Se concluyó que la opción más viable era el uso de TextToSpeech, un módulo integrado en Android que permite aplicar esta funcionalidad de manera muy sencilla.

Tras concluir con la fase de investigación se decidió tomar como una de las principales referencias el video[17] creado por Ivan Goncharov, en el que se expone de manera detallada el proceso de construcción de un sistema capaz de reconocer gestos en tiempo real a partir de los fotogramas de la cámara de un ordenador en Python.

En primer lugar, el video aborda una de las principales cuestiones a tratar en nuestro proyecto, la adquisición de los fotogramas. Para ello se hace uso de la biblioteca OpenCV, que ofrece un amplio abanico de herramientas para el tratamiento de imágenes y vídeos en Python. A través de esta biblioteca, se puede acceder a la cámara del dispositivo y obtener los fotogramas necesarios.

Una vez obtenidos los fotogramas, es necesario su procesamiento y posterior extracción de la información relevante para el reconocimiento de gestos. Para ello, se recurre al uso de técnicas de aprendizaje automático, en particular de aprendizaje profundo (Deep Learning), haciendo uso de redes neuronales convolucionales(CNN), un tipo especial de red neuronal diseñada para trabajar específicamente con imágenes.

La idea central del enfoque del aprendizaje profundo utilizado es la consideración de cada fotograma como una matriz RGB[18], que contiene toda la información necesaria para la identificación de gestos satisfactoria. Esta matriz es analizada mediante una red neuronal convolucional, dicha red neuronal convolucional procesa la matriz RGB y extrae las características más relevantes para el reconocimiento de gestos.

Tras este enfoque inicial el video centra su contenido en la explicación del código[19] implementado originalmente por Kazuhiro Takashi y traducido por Kinivi que mediante el uso de MediaPipe realiza un reconocimiento satisfactorio de gestos establecidos por el

usuario. Usando la matriz RGB mencionada anteriormente y MediaPipe se obtendrán los respectivos landmarks asociados a un fotograma que luego se utilizarán en la red neuronal convolucional.

Este código ofrece la posibilidad de ajustar el número máximo de manos a detectar en un mismo fotograma mediante el parámetro "max_num_hands". La configuración inicial se encontraba definida para la detección de una única mano.

Una de las principales ventajas que ofrece este código es la posibilidad de agregar gestos personalizados definiendo ejemplos de entrenamiento desde la propia aplicación, simplemente ejecutando la aplicación en modo “debug” y pulsando en el teclado el número asociado con el gesto a entrenar, lo que deriva en el guardado de los landmarks asociados al gesto en cuestión.

El vídeo explica también de manera detallada la integración entre las librerías de OpenCv y MediaPipe en Python pero en el caso de nuestra aplicación no es necesario, ya que se utilizan los fotogramas capturados desde un dispositivo Android usando Java.

Para conseguir que la aplicación Python reconociese los gestos necesarios se llevó a cabo el proceso de adición de nuevos gestos descrito anteriormente. Para ello se utilizó la aplicación Android DroidCam[20], que permite la conexión remota a una cámara de un dispositivo Android desde un ordenador mediante el uso de una dirección IP. Usando esta herramienta, se crearon aproximadamente cien ejemplos de entrenamiento por cada gesto y por cada integrante del equipo con el objetivo de obtener un reconocimiento de gestos lo más preciso posible. Tras esto se combinaron todos estos datos en un mismo fichero csv y se entrenó a la IA para que fuera capaz de detectar los gestos personalizados.
Fue necesaria la modificación del código para ajustar los porcentajes de acierto mínimos para considerar un gesto como detectado correctamente ya que los porcentajes preestablecidos eran demasiado bajos, lo que conllevaba que el modelo detectara un gesto incorrectamente de forma habitual antes que simplemente, no detectar el gesto. Esto supone un problema ya que se detectarían gestos no intencionados por parte del usuario de forma frecuente. Además, se realizó otra modificación sobre el código en el que se realiza la detección de manos para detectar la

existencia de una mano sin ningún gesto conocido por la IA, que posteriormente será utilizado en la fase de desarrollo para detectar si el usuario está tratando de realizar un gesto y no está siendo detectado.

3.3.2. Iteración 1

Completado el estudio necesario para el desarrollo de la aplicación se procedió a la creación de un en proyecto de Android Studio teniendo en cuenta las características software que necesitaría la aplicación en un futuro.

Considerando que posteriormente la idea es trasladar la aplicación móvil a una aplicación funcional dentro de unas gafas de realidad aumentada (que poseen un software bastante actualizado) por lo que se decidió usar una versión de api 31 como mínimo y una versión de api 33 como *target,* estas versiones de api corresponden a la versión 12 y a la versión Tiramisú respectivamente, tal como se muestra en la imagen 3.3.2.1 obtenida del propio Android Studio. Es por esto que esta aplicación podrá usarse sin problemas en cualquier dispositivo Android actual. No se añadió ninguna configuración adicional en un inicio debido a que podría derivar en ciertas incompatibilidades con las diferentes librerías a utilizar.

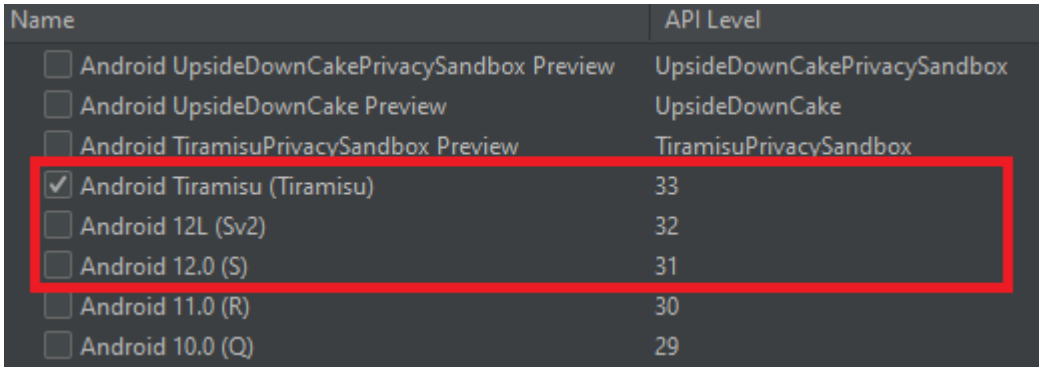

Imagen 3.3.2.1. Lista de correspondencia entre la versión Android y el de API

Una vez se subió el código al gestor de repositorios escogido (*Github*) se realizó un estudio de las distintas librerías que pudieran ser útiles para realizar la conexión entre la aplicación Android y la aplicación Python.

Tras barajar las distintas opciones se concluyó que la librería que más encajaba con los requisitos de la aplicación era *Chacuopy,* ya que el resto de soluciones[21] se enfocaban principalmente en la interfaz gráfica, la cual no es requerida para el desarrollo de nuestra aplicación. Esta librería permite la instalación de dependencias Python desde un repositorio privado que se añaden posteriormente en el paquete de aplicación Android (APK), además de incluir la posibilidad de realizar llamadas a código de Python desde código java. Para su configuración fue necesaria la especificación de la versión de Python a usar (3.10) y la ruta hasta el ejecutable de Python. Una vez configurada se creó un ejemplo simple para probar el correcto funcionamiento de la librería, que consistía en sumar dos valores que recibía por parámetros y devolver el resultado, comprobando que se realiza correctamente el envío y recepción de datos del código Python. Dado que para invocar al código de detección de gestos en principio se usaría una implementación muy parecida (enviando la información necesaria de cada fotograma y obteniendo el gesto asociado) se subió al repositorio como código de ejemplo de uso de esta librería.

Una vez fue posible realizar la llamada a un código Python desde la aplicación Android, se tomó como idea inicial la reutilización del código Python. Para ello era preciso gestionar la cámara desde Android y ser capaces de enviar cada poco tiempo la imagen obtenida.

Debido al conocimiento previo sobre las librerías utilizadas en el proyecto Python para la gestión de la cámara, se trató buscar estas mismas librerías en su versión para Android. Una de las librerías clave dentro del flujo de la aplicación Python era OpenCV, una librería cuya funcionalidad permitía realizar la gestión de la cámara. Al añadirla al proyecto Android se procedió a la creación de una pantalla simple para probar que el uso de la cámara del dispositivo era el esperado. Tomando como referencia la documentación oficial de OpenCV para Android se crearon los métodos básicos del flujo de la aplicación y se mantuvieron en el proyecto para probar la conexión entre ambas aplicaciones.

Previamente a comprobar el buen funcionamiento de la aplicación Python desde nuestra aplicación se debían instalar las librerías necesarias para la misma, estas se añaden directamente a la configuración del proyecto Android al realizar la compilación de la aplicación, ya que *Chaquopy* se encarga de descargarlas desde su repositorio

privado. La lista de librerías que se necesitaban para poder ejecutar el código Python (siendo la mayoría librerías destinadas el procesamiento de datos y entrenamiento de un modelo de predicción) es la siguiente: Wheel, OpenCV, TensorFlow, Scikit-learn, Scipy, Matplotlib, Pandas, Seaborn y MediaPipe. Al tratar de instalar dichas librerías surgió el primer inconveniente, no existía la posibilidad de instalar MediaPipe como librería ya que no se encontraba presente en la biblioteca privada de Chaquopy y tampoco estaba planteada una futura adición, tal como se especifica en una incidencia del repositorio de Github oficial de la librería [22].

Esto conllevó la modificación de la planificación original del proyecto, pues al asumir que esta solución podría funcionar, se había planeado que se podría comenzar a incorporar la funcionalidad de los distintos gestos en un corto periodo de tiempo y finalizar una primera versión de la aplicación. Este cambio provocó que fuera necesario plantear distintas soluciones que se discutieron con los tutores durante una reunión en la que se expuso el inconveniente existente y se llegó a la conclusión de que la solución más viable era añadir la librería de MediaPipe a nuestro proyecto de Android, para procesar la imagen y pasar este resultado al código Python. Esto requería no sólo añadir esta nueva librería sino que sería necesario modificar el código Python para que aceptara el resultado obtenido por MediaPipe, un objeto que especifica las coordenadas de los distintos puntos de la mano tal como se representa en la imagen 3.3.2.2.

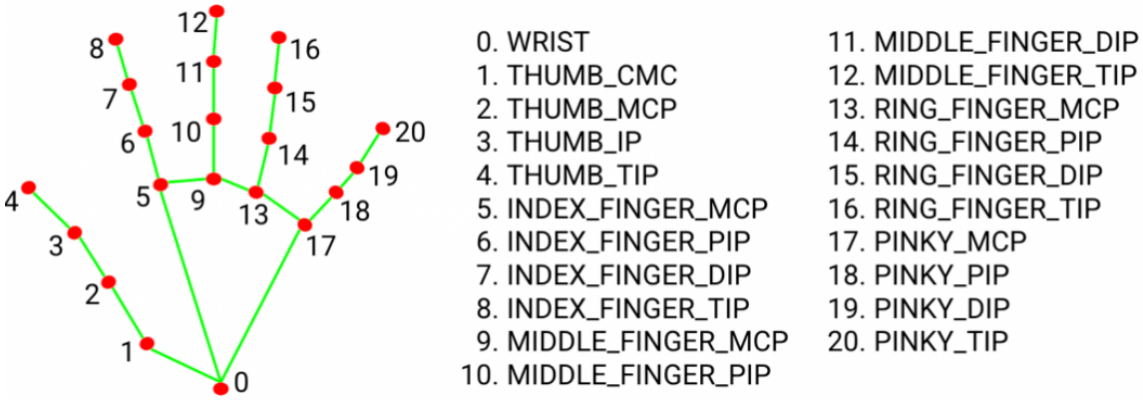

Imagen 3.3.2.2: Puntos de la mano que usa el objeto de la librería MediaPipe[23]

Para conformar esta nueva solución en primer lugar se estudiaron los distintos ejemplos proporcionados por la propia librería de MediaPipe, con el objetivo de comprender el

funcionamiento interno de la librería y el flujo del objeto que contiene las landmarks para adaptarlo a nuestro proyecto y realizar el envío correctamente a la aplicación Python. Durante dicha búsqueda se analizó un repositorio de Github con varios ejemplos de aplicaciones que utilizaban MediaPipe enfocada a diversas áreas, en nuestro caso se investigó el repositorio de la aplicación para la detección de manos dónde se encontraba un proyecto Android[24] que permitía procesar imágenes estáticas, vídeos e incluso el procesamiento en streaming conectado a la propia cámara del dispositivo. La aplicación permite mostrar en tiempo real la imagen reflejada en la figura 3.3.2.3 donde se distinguen los distintos landmarks.

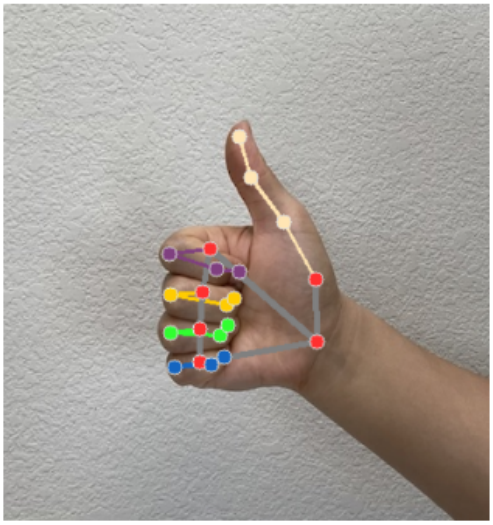

Imagen 3.3.2.3: Resultado visual de la librería MediaPipe con una mano real [25]

Se tomó este proyecto como referencia y aprovechando que permitía el procesamiento en streaming se modificó para que, por cada fotograma procesado, en caso de haber detectado una mano, se enviará el objeto que contiene las landmarks para ser procesado por el código Python.

Este proceso fue algo complejo ya que existían incompatibilidades entre los objetos de Java y Python y los errores impresos en la consola no eran lo suficientemente descriptivos como para seguir la traza de los errores. Se eliminaron varias funciones del fichero “app.py” que no serían necesarias provocado por el uso de la librería MediaPipe en la aplicación Android, pero se producía un error al intentar iterar sobre los puntos del objeto Java que se había obtenido en la aplicación Android. Tras varias pruebas

cambiando los tipos de listas que contenían estos puntos a listas iterables y cambiando las distintas funciones que iteran sobre estos puntos se llegó a la conclusión de que los tipos eran simplemente incompatibles y que se deberían formatear antes de pasarlos a la aplicación Python.

Para obtener un formato universal a todos los lenguajes se decidió convertir el objeto Java con las coordenadas a un objeto JSON. Se usó la librería GSON para Java que permite hacer esta conversión de manera bastante sencilla. Además, se modificó el código de Python para leer y convertir este objeto a un objeto Python sobre el que fuera posible iterar. Fue necesario también realizar ciertas modificaciones en el código ya que algunos nombres de variables no coincidían con los nombres definidos en el código original (por ejemplo, en Java los nombres de las coordenadas tenían una barra baja al final: x_, mientras que en el código Python no).

Una vez realizados estos ajustes fue posible realizar una conexión satisfactoria entre las dos aplicaciones y obtener en Java el identificador del gesto a procesar.

Dado que en este punto ya era posible obtener el gesto detectado se procedió a investigar la posibilidad de modificación de las propiedades de la cámara (zoom, brillo y contraste). La mayoría de soluciones que se encontraron referenciaban a un tipo Camera, el cual en la actualidad está obsoleto ya que había sido sustituido por los tipos Camera2 y CameraX. Tras realizar una investigación sobre el tipo usado dentro del ejemplo obtenido del repositorio de MediaPipe (que había sido modificado para recibir el identificador de los gestos) se encontró que utilizaba una implementación del tipo Camera2 pero con sus propios métodos. Al intentar modificar los valores nos dimos cuenta de que todas estas características estaban creadas de manera interna en la biblioteca y sus valores eran de solo lectura, y por lo tanto, no editables, de tal forma que no fue posible continuar el desarrollo utilizando esta solución y era preciso buscar una alternativa dónde era necesaria la creación de la instancia de Camera2 o CameraX para tener el control sobre la misma y editar sus parámetros.

Tal como se ha explicado anteriormente, este ejemplo contiene tres métodos de procesamiento: imagen, vídeo y streaming. Dado que el streaming usaba directamente una cámara no editable se llegó a la conclusión de que sería necesario crear una instancia de la cámara y añadir un listener que se activa a cada fotograma para enviar la información de la imagen a la aplicación Python. Para poder implementar esta solución

se debía primero escoger entre las dos opciones disponibles de objetos para gestionar la cámara: Camera2 y CameraX. Tras realizar una investigación se llegó a la conclusión de que la primera solución añadía mucha más complejidad en la inicialización pero permitía una mayor personalización y la segunda proponía simplificación de la primera con algo menos de personalización pero con las funciones básicas que serían suficientes para cumplir con los objetivos de la aplicación, por este motivo se decidió usar CameraX.

CameraX tiene varios modos de funcionamiento de los que podemos destacar previsualización (preview) y análisis (analysis), la primera está diseñada para hacer operaciones muy básicas (como por ejemplo una captura de pantalla) y la segunda añade una capa de complejidad para permitir otras operaciones. Tras investigar sobre estas dos implementaciones se llegó a la conclusión de que ambas tenían funcionalidades que necesitábamos: nos interesaba la simplicidad de preview para obtener un mapa de bits de la imagen de manera muy sencilla (mediante la función de getBitmap) y las funciones que añadía el modelo de análisis, como la posibilidad de añadir un listener que se ejecuta a cada fotograma para poder llamar a la aplicación de Python. Es por esto, que se decidió usar una combinación de ambas, usando la previsualización para mostrar la imagen (debido a que al tener una funcionalidad mucho más básica el número de fotogramas por segundo era mucho mayor y lo fácil que era obtener el mapa de bits) y la de análisis para poder crear un listener que se ejecute a cada fotograma.

Una vez creada esta funcionalidad fue añadido en el listener la llamada al método de análisis de imágenes mencionado anteriormente, pasando por parámetros un mapa de bits representando la imagen actual. Se modificaron ciertos parámetros ya que el método de procesamiento de imágenes además de devolver el objeto con las coordenadas de los puntos de la mano también creaba una nueva imagen donde se mostraban de manera visual estos puntos, tal como se ha explicado anteriormente, esta funcionalidad no era necesaria para nuestra aplicación. La aplicación Python se comportaba de la misma manera que con la funcionalidad de streaming, recibía el objeto de coordenadas (que ahora se obtenía del procesamiento que MediaPipe realiza sobre el mapa de bits mencionado anteriormente) y devolvía un identificador representando el gesto indicado.

Tras esto se empezaron a procesar los identificadores asociados a cada uno de los gestos en la aplicación Android: se añadió una etiqueta en la aplicación con el valor por defecto de “gesto no detectado” que cambiaba en base al gesto que se hubiera obtenido para que el usuario pudiera saber de manera visual el gesto detectado (funcionalidad que luego se expandirá en la siguiente iteración para añadir un narrador que leyera el gesto reconocido) y se realizaba la funcionalidad requerida dependiendo del gesto.

Cada vez que se detectaba algún gesto de acción (como puede ser el brillo o contraste) se guarda en una variable y en caso de que el siguiente gesto sea aumentar o disminuir se realizará la funcionalidad correspondiente. Además, dado que el listener se realizaba a cada fotograma, para que el usuario tenga tiempo suficiente como para cambiar entre gestos (por ejemplo, si detecta zoom y se hace el gesto de aumentar, si se analiza a cada fotograma se aumentará el zoom más de lo que probablemente quiera el usuario) se añadió un temporizador de dos segundos a la detección de gestos.

En cuanto a los gestos de contraste y zoom sólo era necesario modificar ciertos parámetros de la cámara mediante distintos setters que nos proporcionaba el modelo de CameraX, incrementando o decrementando en base a un factor definido en la aplicación.

El caso del brillo es algo más particular ya que CameraX no proporcionaba una forma sencilla de modificarlo (tras investigar se encontró que no existía una forma directa de cambiar el brillo en los tipos de cámara estudiados), tras pensar acerca de las posibles soluciones a este inconveniente se llegó a la conclusión de que al ser una aplicación que en un futuro se usará en unas gafas de realidad aumentada y que será la aplicación principal de las mismas, se podría modificar directamente el brillo del dispositivo por lo que se llevó a cabo esta solución en la que en base a un factor se cambiaba el brillo. Esta implementación tenía otro problema ya que para poder cambiar los parámetros de configuración del dispositivo el usuario debe conceder permisos a la aplicación la primera vez que se inicie. Dado que desde un principio se ha querido mantener una aplicación ética (donde todos los permisos y acciones son visibles al usuario) y se había planteado una iteración para implementar esto (iteración 3), se decidió dar una funcionalidad básica de aceptación de estos permisos (al iniciar la aplicación se abrían los ajustes para dar los permisos necesarios a la app) y esta se expandirá en la

iteración mencionada anteriormente, añadiendo una explicación de por qué se necesita dar permisos a la aplicación y cómo hacerlo de manera sencilla.

Todas estas características tienen un valor máximo y mínimo dependiendo del dispositivo por lo que también se añadió una notificación por pantalla para indicar si la característica seleccionada no se puede modificar.

Una vez se habían implementado todas las funcionalidades descritas anteriormente, la aplicación Android era capaz de detectar gestos en tiempo real y cambiar los parámetros objetivo (zoom, contraste y brillo).

Por último, se realizaron las pruebas descritas en el plan de pruebas y cuyos resultados se podrán encontrar en el apéndice A al final del documento.

Aunque la funcionalidad básica de la aplicación cumplía con muchos de los objetivos principales de la aplicación, se detectaron varias mejoras que se planearon para desarrollar en la última iteración.

La primera mejora es la adición de un gesto nuevo que restablezca todos los valores de zoom, brillo y contraste a sus valores iniciales. Esto se propuso debido a ciertos errores que podían ocurrir en la detección de un gesto haciendo que se modificara una característica, aumentando o disminuyendo más de lo que se quería.

A partir de la primera mejora y aprovechando la necesidad de tener que entrenar la inteligencia artificial con nuevos casos de prueba, se propuso añadir una mayor cantidad de datos de los que se tenían inicialmente, esto haría que se pudiera obtener una detección de gestos mucho más fiable disminuyendo en gran medida cualquier error que pudiera ser producido por esta detección.

Por último, se encontró un fallo relacionado con un parámetro por defecto en Android: la pantalla tiene un tiempo máximo de inactividad antes de bloquearse y en nuestra aplicación dicha característica carece de sentido ya que al no precisar de una interacción del usuario con la pantalla este parámetro haría que se bloqueara. Dado que esto es un cambio muy pequeño se decidió arreglar en la próxima iteración.

3.3.3. Iteración 2

Partiendo de la aplicación creada en la iteración anterior se empezaron a construir de forma paralela la escritura de un manual de usuario y la mejora de la aplicación creada

para añadir la posibilidad de configurar ciertos parámetros y el narrador, además de arreglar la mejora mencionada en la iteración anterior.

En primer lugar, se cambió el método de creación de la pantalla principal para que no se bloqueara nunca. Para ello se usó un *flag* de Android que controla si la pantalla está activa siempre o no (*FLAG_KEEP_SCREEN_ON*). Tras añadir esto, mientras se estuviera ejecutando la aplicación en primer plano el dispositivo no se bloquearía, arreglando así el problema detectado en la iteración anterior.

En cuanto al manual de usuario, se intentó realizar una explicación muy sencilla de la aplicación para que cualquier usuario pudiera entender su funcionamiento, además de intentar realizarlo de la manera más breve y esquematizada posible para facilitar la lectura desde la propia aplicación. Dado que este manual de usuario podría cambiar al expandir las funcionalidades de la aplicación, se realizó una implementación en Android para que se pudiera cambiar el contenido de este manual sin modificar código, mediante la lectura de un fichero de texto. De esta manera una vez se consiguiera realizar una lectura de una primera versión del manual de usuario de forma correcta, sólo haría falta cambiar el contenido de dicho fichero cada vez que se creara una nueva versión. Para implementar esta funcionalidad se añadió un botón con el símbolo de información en la esquina superior derecha, al pulsar sobre este botón se redirecciona a una nueva pantalla donde se muestra el manual de usuario que se lee desde el fichero anteriormente mencionado y se muestra un nuevo botón para volver a la vista principal de la aplicación. Tanto el botón de redirección a la pantalla del manual de usuario como la propia pantalla de manual de usuario se muestran en la imagen 3.3.3.1. El manual de usuario completo con todas las funcionalidades desarrolladas se detalla en la sección Resultados 1.

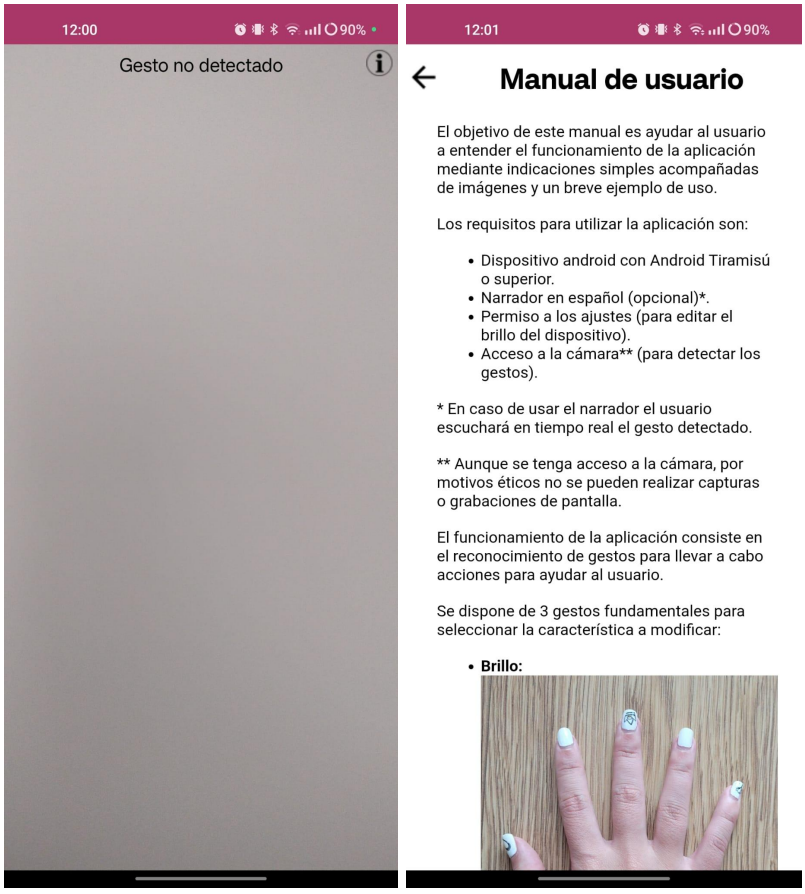

Imagen 3.3.3.1: Pantalla inicial de la aplicación con el botón de información y pantalla de manual de usuario

Después de añadir esta pantalla se comenzó la implementación del narrador. La idea de este es facilitar el uso de la aplicación a personas que no sean capaces de leer los textos que se muestran en pantalla. Para implementar esta funcionalidad se hizo uso del módulo TextToSpeech, módulo que se encuentra incorporado por defecto en Android. Al inicio de la aplicación se realizaba una inicialización de este módulo indicando el paquete de idioma a usar, dado que la aplicación en un inicio está pensada para usarse en español y por ahora no soporta otros idiomas se usó el paquete de este idioma. Una vez se había inicializado el narrador simplemente se usaba la función speak con el valor del gesto que se hubiera detectado para que el usuario pudiera saber qué acción estaba realizando.

Por último, en cuanto a la configuración de la aplicación, en primera instancia, se planteó la posibilidad de cambiar ciertos parámetros de la misma, como son el tamaño de letra, el tipo de letra y la posibilidad de activar o desactivar el narrador. Sin embargo, debido a que la idea era que la aplicación se use en unas gafas como aplicación principal se llegó a la conclusión de que estas variables se podrían modificar directamente en los ajustes generales de Android. Añadir una pantalla para modificar estas características no aportaría ninguna utilidad al proyecto y haría que se tuviera menos tiempo para arreglar cualquier tipo de fallo que pudiéramos encontrar, por lo que se descartó la implementación de este módulo.

Habiendo descartado la implementación anterior, la funcionalidad de configuración se redujo a guardar todos los valores de brillo, contraste y zoom de cada ejecución con el objetivo que, al volver a iniciar la aplicación, se cargara por defecto con estos valores y que el usuario no tuviera que cambiarlos cada vez que inicia la aplicación. Para implementar esto se hace uso del módulo *shared preferences* que proporciona Android para poder guardar y cargar ciertos parámetros (en el caso de esta aplicación, números que reflejaban el nivel de zoom, brillo y contraste actuales). Al cargar la aplicación se haría una llamada a este módulo para cargar las características guardadas y en caso de no encontrarlas se carga un valor por defecto para cada una. Tras esto, cada vez que se cambiaba el valor de una de estas características se usaba el editor de este módulo para guardar esta configuración usando una notación de clave-valor, indicando en la clave el nombre de la característica que había cambiado.

Por último, se realizaron las pruebas descritas en el plan de pruebas y cuyos resultados se podrán encontrar en el apéndice B al final del documento.

Dado que esta Iteración no tenía un gran añadido a la funcionalidad principal de la aplicación (detección de gestos) no se detectaron fallos o mejoras a realizar en la siguiente iteración.

3.3.4. Iteración 3

Esta iteración comenzó con el desarrollo del código para el nuevo gesto que se encargaría de reiniciar todas las características de la aplicación (zoom, contraste y brillo) a un valor por defecto. Este proceso requería la modificación de la funcionalidad

tanto de la aplicación Android para que realizara la funcionalidad asociada a este gesto como añadir los casos de entrenamiento para que la aplicación de python sea capaz de reconocer este gesto.

En cuanto a la funcionalidad de la aplicación Java simplemente fue necesario reutilizar el código usado para establecer los valores de zoom, brillo y contraste a los valores definidos por defecto. Este proceso fue tan sencillo debido a que el código de las acciones asociadas a los distintos gestos fue implementado contemplando una ampliación futura para añadir más gestos. En el caso de Python, aprovechando la necesidad de añadir este nuevo gesto se realizó un nuevo proceso de reentrenamiento de la inteligencia artificial de nuevo y así eliminar cualquier error producido por el reconocimiento de los gestos anteriores. Antes de entrenar la IA se valoró que gesto sería el más adecuado para el restablecimiento de los valores iniciales de zoom, brillo y contraste; dado que debía ser un gesto fácilmente distinguible de los gestos ya existentes se decidió usar el gesto representado en la imagen 3.3.4.1. Tras esto se borraron todos los registros de los gestos anteriores y, usando el modo “debug” (desde la aplicación de Python original) se volvió a generar el fichero csv con todos los casos de prueba para el entrenamiento de la IA. Al realizar la primera versión de este reconocimiento se usaron en total aproximadamente 800 datos para cada uno de los gestos (400 para cada mano) pero en esta ocasión se aumentó este número a 500 para que la IA pudiera distinguir entre todos los gestos con mayor facilidad. Una vez completo el fichero csv se ejecutó el código correspondiente al entrenamiento de la IA con los nuevos datos y se pasaron al proyecto de Android los archivos necesarios para usar esta nueva IA (keypoint_classifier.hdf5 y keypoint_classifier.tflite). Por último, se hicieron diversas pruebas para corroborar que se obtenían todos los gestos de manera correcta.

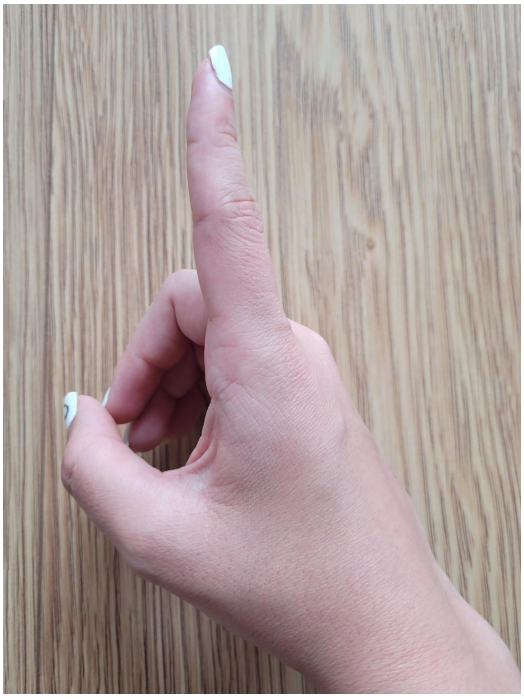

Imagen 3.3.4.1: Gesto restablecer

De forma paralela a esta implementación se desarrollaron todas las funcionalidades relacionadas con la ética de la aplicación: no permitir las capturas ni grabar la aplicación y explicar al entrar a la aplicación por primera vez para qué se necesitan los permisos que se solicitan.

En primer lugar se deshabilitaron la posibilidad de realizar capturas y grabar la pantalla dentro de la aplicación, esto se realizó mediante la activación de un flag llamado *FLAG_SECURE* en todas las pantallas de la aplicación (la pantalla principal donde se muestra la cámara y la pantalla de manual de usuario). Una vez añadido este flag, cada vez que se intentará realizar una captura de pantalla, se mostraba un mensaje explicando al usuario que la aplicación no permite esta operación.

Tras esto se estudió el mecanismo de solicitud de permisos actual en la aplicación. El permiso de acceso a la cámara es solicitado por defecto en Android para cualquier aplicación que haga uso de la misma y dado que se solicita de una manera clara, tal como se muestra en la imagen 3.3.4.2 no se añadió ninguna explicación adicional.

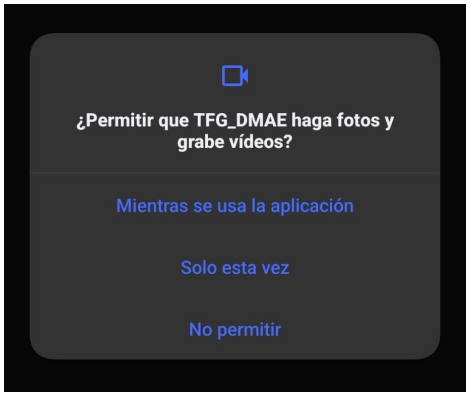

Imagen 3.3.4.2: Diálogo para pedir los permisos de la cámara en la aplicación Android

El caso del permiso para cambiar el brillo del dispositivo es algo diferente, ya que por defecto al requerir este permiso la aplicación redirige al usuario a una pestaña de ajustes en la que es posible conceder los permisos necesarios a la aplicación. Tal como muestra la imagen 3.3.4.3, esta pestaña es algo confusa y no explica con claridad las acciones que debe tomar el usuario: desplazarse hasta la aplicación, pulsar sobre ella y concederle los permisos.

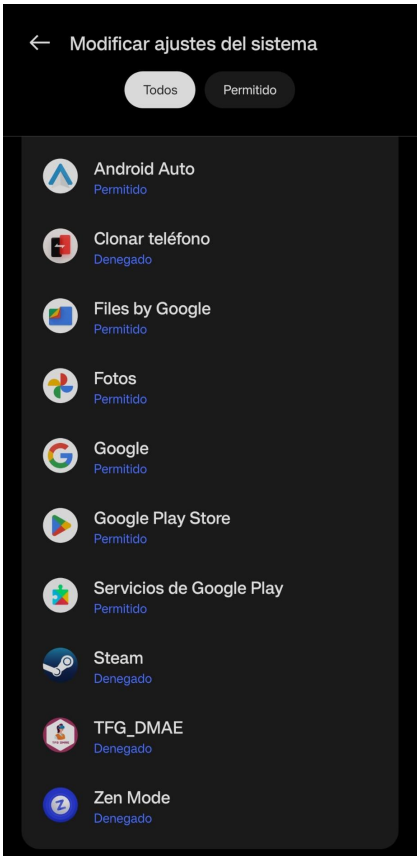

Imagen 3.3.4.3: Ventana para modificar los ajustes del sistema

Es por esto que se decidió añadir un diálogo modal con una breve explicación acerca de cómo conceder los permisos a la aplicación y una vez que se hubiera leído y pulsado el botón de confirmar se mostraría esta pestaña de ajustes para que el usuario pudiera conceder los permisos, este diálogo se puede ver en la imagen 3.3.4.4. Para cambiar esta funcionalidad fue necesario reorganizar mínimamente el flujo de la aplicación ya que en un inicio se daba este permiso en la función inicial (y por tanto nada más abrir la aplicación redirigía al usuario a esta pestaña de ajustes sin llegar a ver el diálogo modal). Por este motivo el permiso de la cámara se mantuvo en esta función principal, pero el de los ajustes fue necesario modificar el orden de aparición para que se ejecute al cerrar el diálogo modal y modificar algunas funciones del flujo principal relacionadas con el brillo para que sólo funcionen en caso que el usuario concediera dicho permiso. Cabe destacar que al tratarse de un ajuste que permite modificar varios parámetros del

sistema, se deja a elección del usuario la concesión o no dicho permiso. En caso de que decida no concederlo, la aplicación funcionará igual, pero al reconocer el gesto del brillo en caso de que se quiera aumentar o disminuir se mostrará un mensaje avisando al usuario de que para modificar este parámetro es necesario otorgar permisos a la aplicación tal como se puede observar en la imagen 3.3.4.5.

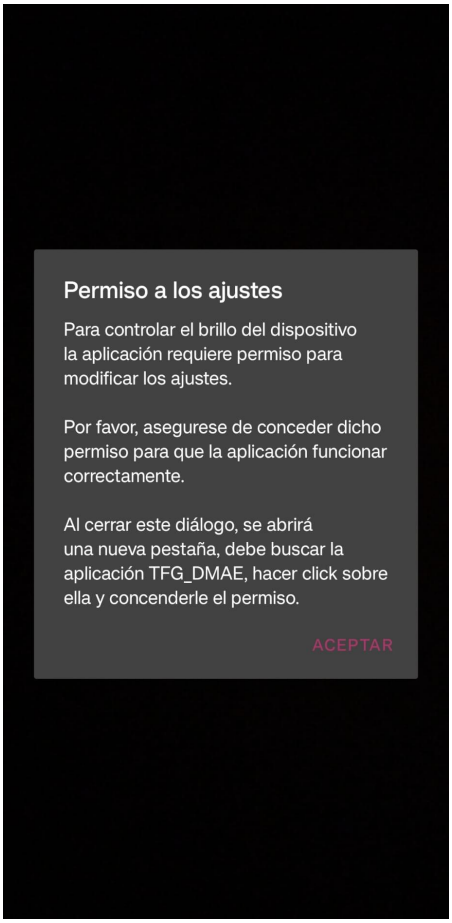

Imagen 3.3.4.4: Diálogo modal con la explicación de acceso a los permisos

Imagen 3.3.4.5: mensaje de no poder editar el brillo en la aplicación

Por último, se realizaron las pruebas descritas en el plan de pruebas y cuyos resultados se podrán encontrar en el apéndice C al final del documento.

4. Resultados

4.1. Manual de usuario

El objetivo de este manual es ayudar al usuario a entender el funcionamiento de la aplicación mediante indicaciones simples acompañadas de imágenes y un breve ejemplo de uso.

Los requisitos para utilizar la aplicación son:

- Dispositivo Android con Android Tiramisú o superior.
- Narrador en español (opcional)*.
- Permiso a los ajustes (para editar el brillo del dispositivo).
- Acceso a la cámara** (para detectar los gestos).

* En caso de usar el narrador el usuario escuchará en tiempo real el gesto detectado.

** Aunque se tenga acceso a la cámara, por motivos éticos no se pueden realizar capturas o grabaciones de pantalla.

El funcionamiento de la aplicación consiste en el reconocimiento de gestos para llevar a cabo acciones para ayudar al usuario.

Se dispone de 3 gestos fundamentales para seleccionar la característica a modificar:

- Brillo:

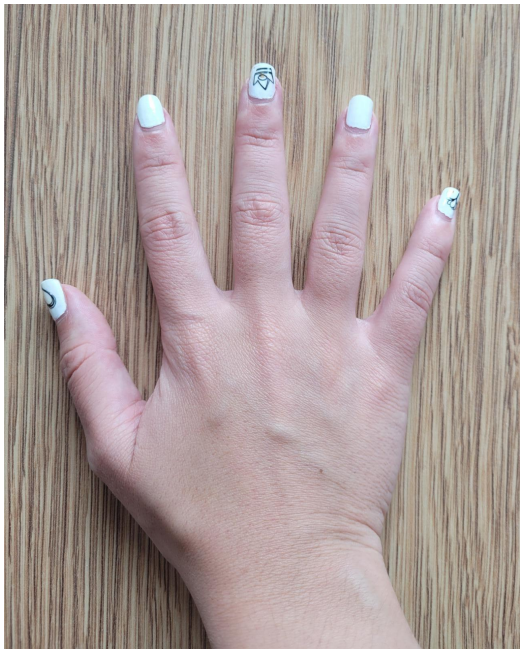

Imagen 4.1.1: Gesto Brillo

- Contraste:

Imagen 4.1.2: Gesto Contraste

- Zoom:

Imagen 4.1.3: Gesto Zoom

Tras realizar cualquiera de estos gestos es posible aumentar o disminuir el valor mediante los siguientes gestos:

- Aumentar:

Imagen 4.1.4: Gesto Aumentar

- Disminuir:

Imagen 4.1.5: Gesto Disminuir

Cabe destacar que los gestos se reconocen cada dos segundos, por lo que si se mantiene el gesto de aumentar durante 4 segundos se producirá dos veces el aumento por ejemplo (ocurre igual para todos los gestos disponibles).

Además, existe un último gesto cuya función es restablecer los ajustes de brillo, zoom y contraste a su valor predefinido:

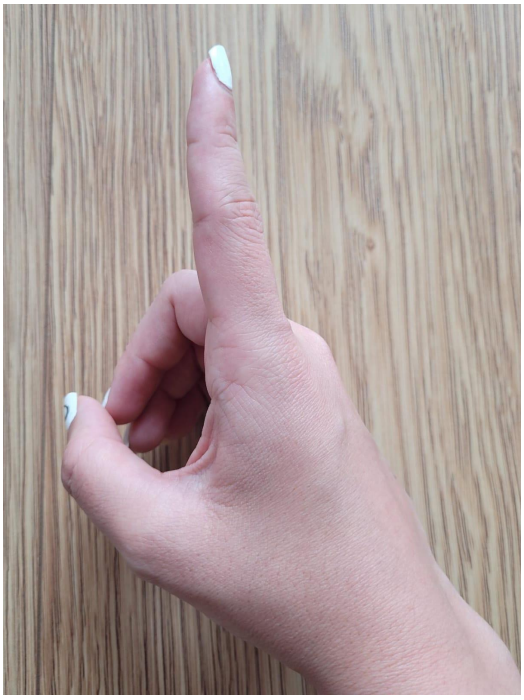

Imagen 4.1.6: Gesto restablecer

Es importante mencionar que al cerrar la aplicación se guarda la configuración actual y se cargará automáticamente al volver a usar la aplicación.

A continuación, se describe un breve ejemplo de uso:

1. El usuario abre la aplicación.
2. Realiza el gesto de contraste (el contraste está seleccionado).
3. Realiza el gesto de aumentar (el contraste aumenta).
4. Realiza el gesto de brillo (el brillo está seleccionado).
5. Realiza el gesto de disminuir (el brillo disminuye).
6. Realiza el gesto de zoom (el zoom está seleccionado).

7. Realiza el gesto de aumentar (el zoom aumenta).
8. Mantiene el gesto de aumentar (el zoom aumenta).
9. Realiza el gesto de restablecer (los niveles de zoom, brillo y contraste se restablecen).

En resumen, el usuario ha aumentado el contraste, ha disminuido el brillo, ha aumentado el zoom dos veces y finalmente ha restablecido los ajustes.

4.2. Aplicación

Para poder evaluar los objetivos logrados con el desarrollo de nuestra aplicación se procede a analizar cuáles de los objetivos iniciales hemos conseguido cumplir:

Objetivo	**Análisis**
Sistema empotrado sin necesidad de conexión a Internet	Cumplido, es posible usar la aplicación sin conexión a Internet
El sistema podrá ser usado como aplicación Android en un smartphone y como aplicación Android en las gafas de realidad aumentada.	Solo se han realizado pruebas de la aplicación en un smartphone
Detección de gestos	Cumplido, la aplicación detecta los gestos con un porcentaje de éxito del 90%
Tiempo de respuesta menor a 3 segundos	Cumplido, la aplicación reconoce los gestos a tiempo real, cada 2 segundos puede reconocer gestos nuevos

Tabla 4.2.1: Objetivos funcionales

En cuanto a los objetivos de accesibilidad que habíamos propuesto:

Objetivo	**Análisis**
Detección de gestos	Cumplido, la aplicación indica de manera clara el gesto reconocido, mediante un texto en la pantalla o si no se ha reconocido ninguno

Manual de usuario	Cumplido, se dispone de un manual de usuario que se puede consultar en caso de dudas sobre la funcionalidad
Narrador	Cumplido, la aplicación dispone de un narrador que lee en voz alta el gesto detectado o si no se ha detectado ninguno
Personalización del texto	Cada usuario puede configurar el tipo de letra que usa en su smartphone, este será usado en la aplicación

Tabla 4.2.2: Objetivos de accesibilidad

Objetivos éticos de la aplicación:

Objetivo	**Análisis**
La cámara	Cumplido, el usuario puede revocar los permisos cuando quiera, no se permite hacer capturas de pantalla dentro de la aplicación y no se guardan las imágenes
La licencia del software	Cumplido, la aplicación tiene una licencia de Software libre

Tabla 4.2.3: Objetivos éticos

Análisis de detección de gestos:

En cuanto al análisis de gestos de forma automatizada se ha realizado una matriz de confusión, reflejada en la imagen 4.2.1 que muestra los resultados obtenidos en el entrenamiento para cada uno de los gestos, acompañado de un informe de clasificación que ayuda a comprender mejor estos resultados, tal como se puede observar en la figura 4.2.2.

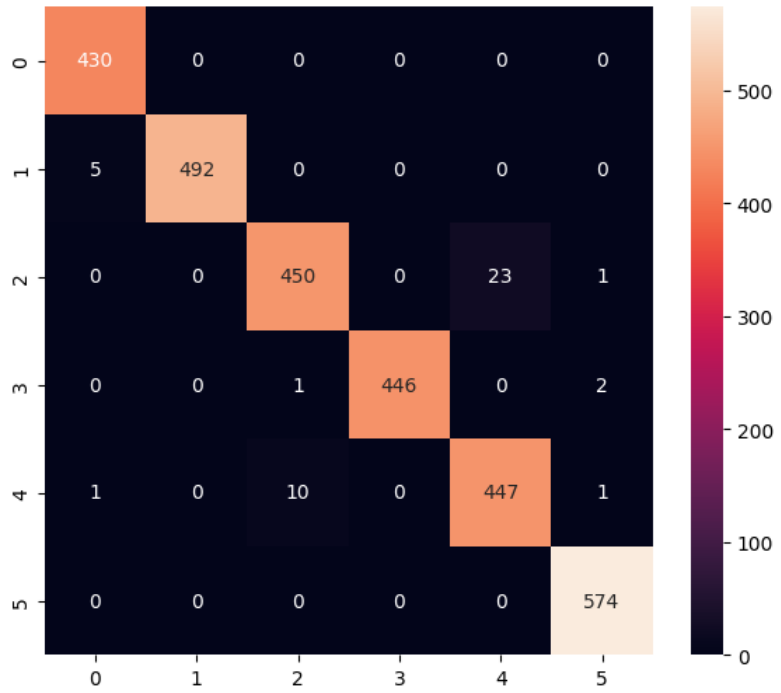

Imagen 4.2.1: Matriz de confusión

Classification Report

	precision	recall	f1-score	support
0	0.99	1.00	0.99	430
1	1.00	0.99	0.99	497
2	0.98	0.95	0.96	474
3	1.00	0.99	1.00	449
4	0.95	0.97	0.96	459
5	0.99	1.00	1.00	574
accuracy			0.98	2883
macro avg	0.98	0.98	0.98	2883
weighted avg	0.98	0.98	0.98	2883

Imagen 4.2.2 Informe de clasificación

En los datos, los gestos se representan teniendo en cuenta que el 0 simboliza zoom, 1 brillo, 2 contraste, 3 aumentar, 4 disminuir y 5 aumentar.

Cabe destacar que se han usado un 75% de los datos para entrenamiento y el 25% restante para realizar los test de forma automática.

El dataset usado para este entrenamiento, tal como se ha explicado en secciones anteriores, se trata de un csv que incluye el identificador del gesto y los datos asociados a los landmarks. Este csv se ha generado manualmente mediante el uso de la aplicación original de Python y el modo de captura, explicado en la sección 3.3.1.

Con esta premisa, puede observarse en los datos que la precisión del modelo es muy alta para todos los gestos, lo que significa que el modelo está funcionando correctamente y que será capaz de reconocer gestos en situaciones normales.

Además, puede entenderse que la precisión mínima es la del gesto 4 (disminuir) con un 95% de acierto, lo cual cumple ampliamente con las expectativas teniendo en cuenta que el resto de gestos se detectan de forma más precisa e incluso en el caso de aumentar se obtiene una detección casi perfecta.

También es importante destacar que el gesto 5 (reestablecer) se ha identificado correctamente el 100% de las veces que se ha realizado, lo cual es importante ya que dicho gesto está pensado para que el usuario lo utilice tras haber modificado la configuración inicial, lo que conlleva que la detección del gesto será más compleja.

En conclusión, el modelo es apto para utilizarlo en una aplicación en tiempo real.

Como información adicional a los resultados obtenidos en el modelo de entrenamiento, se ha realizado una gráfica que se muestra enla figura 4.2.3 que representa los porcentajes de aciertos que se obtuvieron en la realización de las pruebas del último sprint (pruebas manuales reflejadas en el plan de pruebas de la sección 3.2.

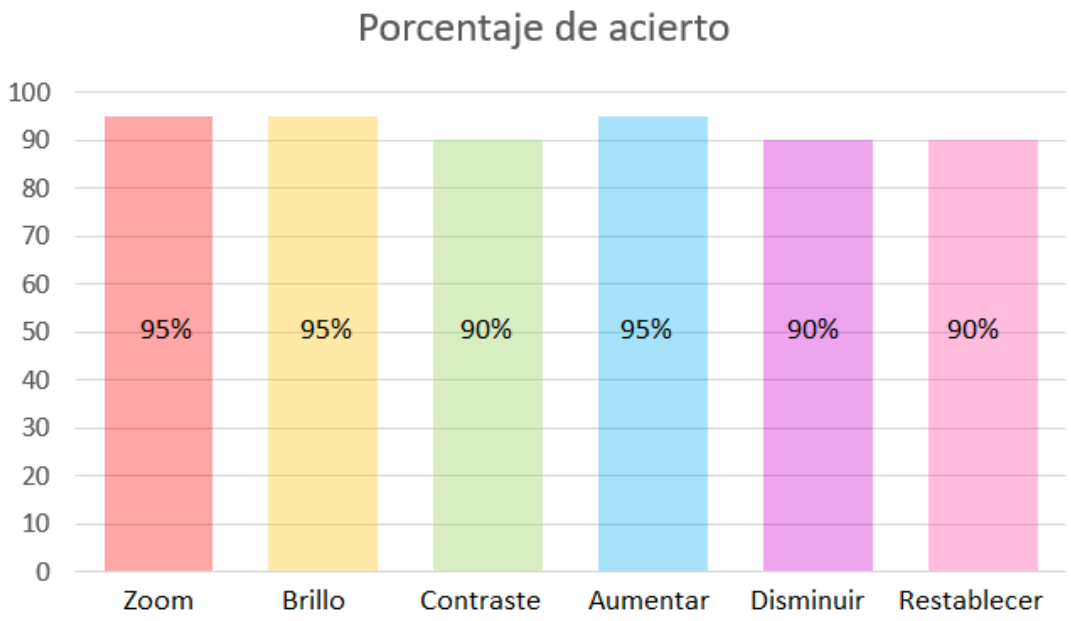

Figura 4.2.3: Porcentaje de acierto de los distintos gestos

4.3. Análisis ético

4.3.1. Licencias de software

Dado que el software que se ha desarrollado en este trabajo tiene como objetivo final su uso en gafas de realidad aumentada para ayudar a personas con problemas de visión, es preciso que la licencia del software permita no sólo editar el código fuente (para que otros desarrolladores puedan usar este proyecto como base para mejorarlo) sino también comercializarlo (para que en caso de que cualquier empresa quiera usar el software pueda hacerlo).

Existen varias licencias de software que cumplen con estas características, una de ellas, la licencia Apache 2.0, se usa además en los otros proyectos software que se han utilizado para implementar la aplicación final (como puede ser el proyecto de Python para la detección de gestos). Debido a la restricción que tiene dicha licencia de añadir una copia de la misma si se usa código protegido bajo esta licencia se decidió dejar como licencia final para la aplicación.

Se ha añadido la copia de dicha licencia en el repositorio de Github para que cualquier persona que quiera usar el código pueda informarse sobre la misma y aplicarla de manera correcta. Además, en el fichero *readme* de este mismo repositorio se ha añadido una explicación de los cambios que se han realizado sobre el software externo que se ha usado en el desarrollo de la aplicación.

4.3.2. Uso de la aplicación: restricciones y soluciones implementadas

Al inicio de la aplicación se detectaron varios problemas éticos relacionados con la aplicación debido principalmente al uso de la cámara y al procesamiento posterior de las imágenes.

El primer problema que se planteó era el uso que se iba a hacer de las imágenes, ya que se debía asegurar que se analizarían únicamente para obtener el gesto asociado a la misma. En relación a esto nos aseguramos de que al desarrollar la aplicación las imágenes se transformaran a mapa de bits y que a partir de este se obtuviera un objeto landmarks que sería leído por la aplicación Python. Esta imagen una vez procesada no se almacenaría de manera local, ya que no se había dado ningún permiso para realizar esto, ni online ya que la aplicación no requiere de ningún tipo de funcionalidad en línea. Además, dado que el uso de la cámara puede involucrar a terceras personas que no estén al tanto de la misma, se añadió una restricción para realizar capturas o grabar la pantalla, tal como se ha explicado en la tercera iteración. De esta manera se restringió el uso de las imágenes tanto dentro de la aplicación en el propio procesamiento de las mismas como en el exterior ante cualquier posible acción del usuario.

El otro problema encontrado para esta aplicación está relacionado con los permisos de la misma: el uso de la cámara y el uso de los ajustes del dispositivo para el ajuste del brillo que se puede realizar en la aplicación. El uso de la cámara, tal como se ha explicado anteriormente, no fue un problema ya que todos los dispositivos Android piden este permiso al iniciar una aplicación que necesite usarla, de tal forma que el usuario cuando inicie la aplicación por primera vez podrá otorgar este permiso. Sin embargo, el uso de los ajustes del sistema era algo poco intuitivo y un usuario podría no

comprender la necesidad de conceder estos permisos, perdiendo así una funcionalidad importante de la aplicación. Es por esto que para solucionar este problema se añadió un diálogo modal en el que se explicaba al usuario la necesidad de conceder este permiso. Además, la aplicación funcionará correctamente independientemente de si el usuario decide dar este permiso o no (sólo cambiará el hecho de poder o no modificar el brillo del dispositivo). Aún después de la adición de dicha solución, sigue existiendo un problema, la posibilidad de encontrar una alternativa para modificar el brillo sin necesidad de concederle el permiso para modificar cualquier ajuste del sistema a la aplicación, pero esto queda fuera del alcance de este proyecto y se explicará como una posible mejora en las siguientes secciones.

Por último, el siguiente problema que se plantea es la accesibilidad. Dado que la aplicación en un futuro se intentará ejecutar en unas gafas de realidad aumentada (que sirven para ayudar a usuarios con problemas de visión) para mejorar la interacción con las distintas características de las mismas (zoom, brillo y contraste) es necesario que cualquier usuario pueda usar la aplicación sin problemas, independientemente del conocimiento técnico que tenga. Para ello se añadió un manual de usuario dentro de la propia aplicación en el que se intentó explicar de la forma más sencilla posible el funcionamiento básico de la aplicación, además de añadir un narrador que fuera capaz de leer cualquier mensaje que apareciera en la pantalla.

5. Conclusiones y trabajo futuro

Como conclusión del proyecto realizado se puede apreciar que la aplicación final cumple con la mayoría de requisitos propuestos, tal como se ha explicado en las secciones anteriores. Esta aplicación es un primer paso para mejorar la calidad de vida de los usuarios de gafas de realidad aumentada que padecen de distintos problemas de visión. Aun así, hay ciertos conceptos que sería interesante implementar en un futuro para poder mejorar la aplicación: aumentar el brillo mediante procesamiento de software, pasar de manera correcta la aplicación a unas gafas de realidad aumentada, optimizar los tiempos lo máximo posible y reentrenar todos los gestos.

Actualmente para poder cambiar al brillo se debe conceder a la aplicación permisos para cambiar cualquier ajuste del sistema, además de que el brillo que se cambia es el del dispositivo y no el de la imagen que se está viendo. Aunque se haya intentado minimizar este problema explicando de manera breve a los usuarios por qué se usa este permiso y permitir usar la aplicación, aunque no se conceda, sería interesante buscar una solución alternativa y debido a que el proceso de buscar una solución software a este problema es muy costoso se plantea como trabajo futuro. Mediante el procesamiento software de la imagen no se requeriría conceder tantos permisos a la aplicación, lo que reduciría los problemas éticos asociados a la misma e incluso si se consigue implementar podría servir como base para añadir otro tipo de filtros a la imagen, que puedan ayudar a los usuarios de la aplicación.

Tal como se ha explicado en varias ocasiones en este documento, el objetivo final de este proyecto es el de utilizar la aplicación en unas gafas de realidad aumentada. Este proyecto es solo la primera fase de este proceso ya que simula el funcionamiento gafas mediante un smartphone. Como trabajo futuro se plantea el, usando esta aplicación como base, probar su funcionamiento en unas gafas de realidad aumentada y cambiar todo lo necesario para optimizar la experiencia de usuario en estas gafas.

Además, al tratarse de una aplicación que está analizando gestos en tiempo de ejecución, es probable que añadir nuevas funcionalidades a la aplicación conlleve un deterioro en el rendimiento de la misma. Es por esto mismo que se debería realizar una revisión exhaustiva de la implementación, para asegurarse de que se ha desarrollado

de la forma más óptima posible, centrándose sobre todo en las nuevas adiciones y cómo estas pueden generar ciertas incompatibilidades con las funcionalidades presentadas en este proyecto.

Por último, siempre existirá un margen de error a la hora de la detección de los gestos ya que se realiza mediante una IA y que dependerá del entrenamiento recibido, para proporcionar mejores o peores resultados. En este proyecto el número de datos que se han usado son bastante limitados y se han utilizado principalmente para un uso estándar de la aplicación, sin embargo, aún se siguen encontrando ciertos fallos en la detección que podrían ser molestos para una persona que usará la aplicación diariamente y que se podrán solventar con un entrenamiento con muchos más datos de prueba de los que se han usado en el proyecto.

Introduction

Motivation

There is a wide variety of programs and applications that are able to identify gestures, however, most of these tools have as their main objective other than helping to solve problems for people with visual impairment or improve their quality of life.

Specifically, this project is focused on improving the daily life of people suffering from AMD (Age-Related Macular Degeneration), a disease that causes degeneration of the macula. As a result, people with AMD experience blurred vision and an increase in a blurred field in the center of their vision, so changing the way they see the world can greatly help them in their daily tasks.

In addition, we believe that the use of gestures as means of interaction with the glasses would be extremely beneficial and timely, as currently, visually impaired people often rely on external devices to record the actions they wish to perform.

Furthermore, this initiative represents a highly interesting project, which can serve as an excellent opportunity to demonstrate the enormous potential of artificial intelligence-based technologies to solve very specific needs and improve the quality of life of people with disabilities.

Goals

To define the main objectives of this project, we will distinguish between three different types: functional objectives (that will allow the application to recognize gestures in the shortest possible time to perform the action associated with it), accessibility objectives (that will allow a user of the application to interact with it without any problem) and ethical objectives (that will allow the user to make decisions about the use of the camera within the application).

As for the functional objectives we find:

- Embedded system without the need for internet connection: the application will be able to be used as an application of the augmented reality glasses themselves and the user will be able to use it without the need to be connected to the internet, since it is an application focused on improving the accessibility of the glasses for people with vision problems, connectivity should not restrict its use.
- The system will be able to be used as an Android application on a smartphone and as an Android application on the augmented reality glasses. Consequently, first the Android application will be developed to achieve gesture detection and later the implementation will be moved to the glasses.
- Gesture detection: the application will detect a series of gestures that will be associated with a characteristic of the image (brightness, zoom and contrast), once detected it will be possible to increase or decrease that characteristic by performing another gesture.
- Response time: since it is an application that detects gestures and performs actions based on this, the response time should be low enough to be an improvement over performing these actions manually using a remote control. The objective of this project is to be able to detect a gesture with a maximum response time of three seconds.

In relation to the accessibility objectives:

- Gesture detection: associated with the functional objective to detect gestures and perform an action based on it, it will be clearly indicated which gesture has been detected so that the user knows which feature of the image is going to be modified. In addition, if the gesture cannot be recognized, the user will be informed.
- User manual: the application will have a user manual that can be consulted in case there is any doubt about the functionality of the app. This manual will be as simple as possible so that a person with minimal knowledge in Information and Communication Technologies (ICT) can understand how the application works.
- Narrator: since the application is intended for visually impaired people, a narrator will be incorporated to all the actions mentioned above (gesture detection, errors...) so that a blind person can use the application without any problem.

- Text customization: the user will be able to change both the typography and the font size to be able to use the application without problems. These settings will be saved so that the next time the application is opened they will be available.

Finally, regarding the ethical objectives we find:

- Regarding the camera: as the application makes use of the camera it will be allowed that the user can revoke the use of the camera at any time. In addition, screenshots will not be allowed within the application itself and no record of the images used for the detection of gestures will be saved.
- Regarding the software license: since it is an application of a certain complexity and with a lot of room for improvement, all the software created will have a free software license. In this way, this application can be used as a basis for multiple applications with the aim of helping people with vision problems.

Work plan

For the development of the project an iterative and incremental methodology has been used, the project is divided into several iterations or short work cycles, obtaining a gradual progress in the development. Each iteration is divided into several processes: planning, design, development, testing and delivery. These iterations are repeated until the project is completed.

The advantages of using this methodology are mainly the facility to adapt the development to changing requirements as more information is obtained, in addition to having continuous deliveries allows to detect and correct errors in early stages of development which translates into an improvement in the quality of the final product.

Regarding workflow management, Kanban, a visual management technique that aims to increase the efficiency and quality of software development, has been used. For this purpose, a Kanban board is used to represent the workflow and the status of the tasks. This board is divided into several columns: "Pending", "In Development", "Testing" and "Finished".

Pending: The task is defined but has not yet started to be developed, therefore, no team member has that task assigned to it.

In Development: The task has been assigned to one or more members of the development team, the task will remain at this stage until it has been completed.

Testing: Once the task has been developed it must be tested before being finalized, the tests must always be performed by a different person than the one who has performed the development of the task, after the completion of the tests the task can be finalized or, in case it has not passed the tests, it must be sent back to the "In development" status to proceed to fix the bugs found. For research tasks the testing phase is not necessary and is terminated once the necessary information has been obtained and documented.

Completed: Upon successful completion of the testing of a task, the status of the task becomes "Completed".

To represent the status of the different tasks we have used the Asana tool, which allows the creation of virtual Kanban boards. Members of the development team can move the tasks between the different columns as they progress through the different stages.

For version control the tool used has been GitHub, a main branch was created to upload completed tasks and additionally, if any task required certain tests, a new branch was enabled for this.

In the specific case of our project, it has been divided into 3 main iterations and an initial research phase prior to development, for none of them a completion or delivery date has been determined except for the last iteration that had to be completed within the deadlines indicated by the Complutense University of Madrid for the delivery of the project.

During the initial phase or research phase, the development team conducted a study of existing research or projects on the main subject of the project, in addition to analyzing the advantages and disadvantages of different technologies that could be used in the development of the project. The objective of this initial phase was to define more precisely the scope of the project and to search for the most suitable technology to meet the proposed objectives.

Initial phase:

In the first instance, the development team was dedicated to the study of existing projects and applications similar to our project, for this purpose different studies related

to the use of artificial intelligence to help visually impaired people were analyzed. This study helped the development team to know the pending research and to be able to define more precisely the requirements of the final application.

Once the existing applications were known and having a panoramic view of the pending research, the next step would be to investigate the technologies that could be used for the development of the project.

Finally, having a list of existing technologies and knowing the pros and cons of each one, small functionality tests were performed on each of the technologies to decide which of them was the most optimal for the characteristics of the project.

A timeline of the initial phase can be seen in image 1.3.1.

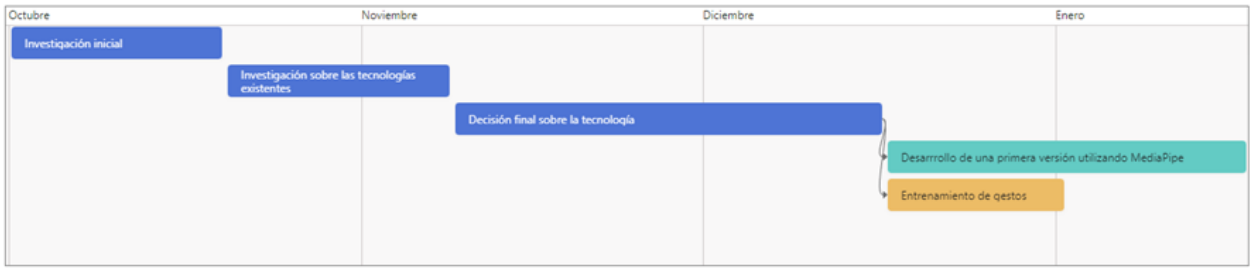

Figure 1.3.1: Timeline of the initial iteration

Iteration 1:

The first iteration of the project was aimed at finalizing the basic functionalities of the application, among which were the detection of the zoom, brightness, contrast, zoom in and zoom out gestures, and the functionality of these, modifying the necessary camera parameters.

This first iteration was undoubtedly the most ambitious of the project, since it not only required the implementation of the detection of the gestures described above and the modification of the camera configuration, but also, as it was our first contact with the project, it was necessary to carry out research parallel to the development, which meant that this iteration was also the most extended in time. But it allowed us to establish a solid functional base on which to develop the following iterations.

To finish the iteration, the required tests were carried out and the errors that arose from them were corrected.

A timeline of iteration 1 can be seen in image 1.3.2.

Figure 1.3.2: Iteration 1 timeline

Iteration 2:

The goal for this iteration was the development of two new functionalities, the access to a user manual from the application itself and the implementation of the voice narrator. Both functionalities were implemented in parallel and a functionality to avoid screen locking while the application is in use was added. Once finalized a new functionality was added that saved the current configuration when closing the application, this way the saved parameters would be loaded when restarting the application.

Finally, all the necessary tests were performed and the resulting bugs were fixed.

A timeline of iteration 2 can be seen in image 1.3.3.

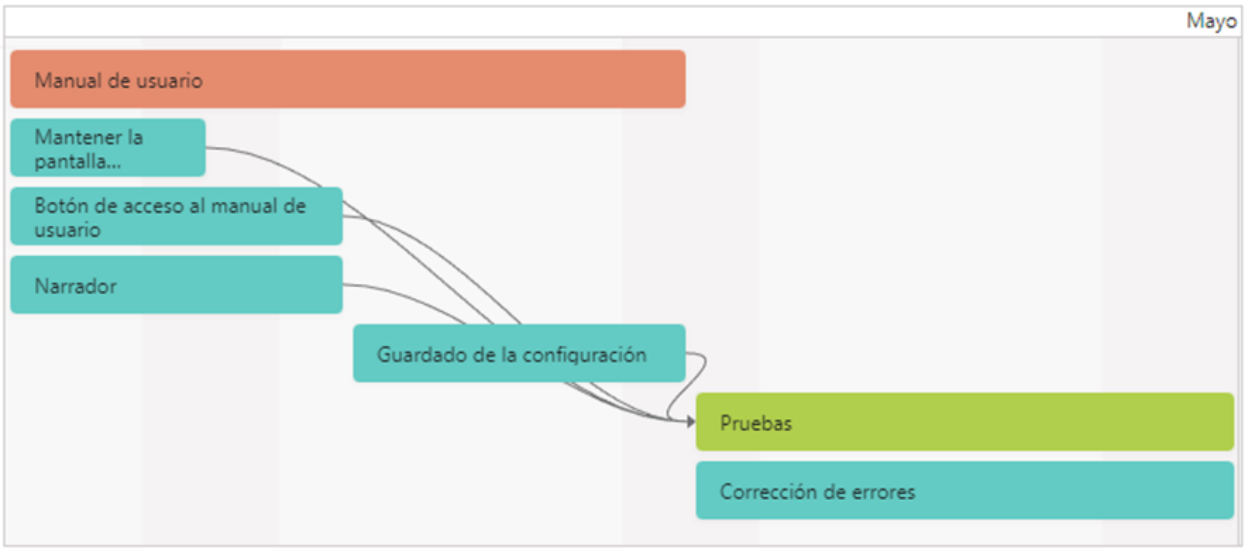

Figure 1.3.3: Timeline for iteration 2

Iteration 3:

The goal of the third and last iteration consisted of implementing a series of warnings and limitations with the objective of guaranteeing the maximum ethical integrity of the application. In addition, a new gesture was included that allowed resetting the configuration to the values defined by default, adding this new gesture did not involve a great effort since the application had been designed to be scalable in terms of detecting new gestures.

Once the development was finished, the associated tests were carried out and the errors found were corrected.

A timeline of iteration 3 can be seen in image 1.3.4.

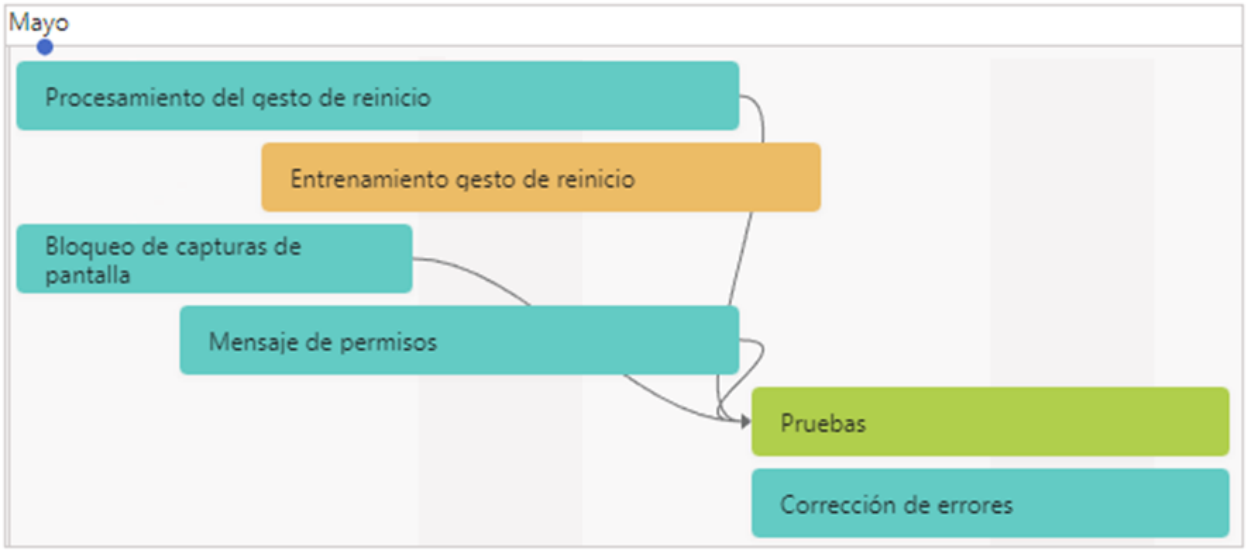

Figure 1.3.4: Timeline of iteration 3

Finally, a pie chart has been created to represent the percentage of time spent on each type of task, as shown in Figure 1.3.5.

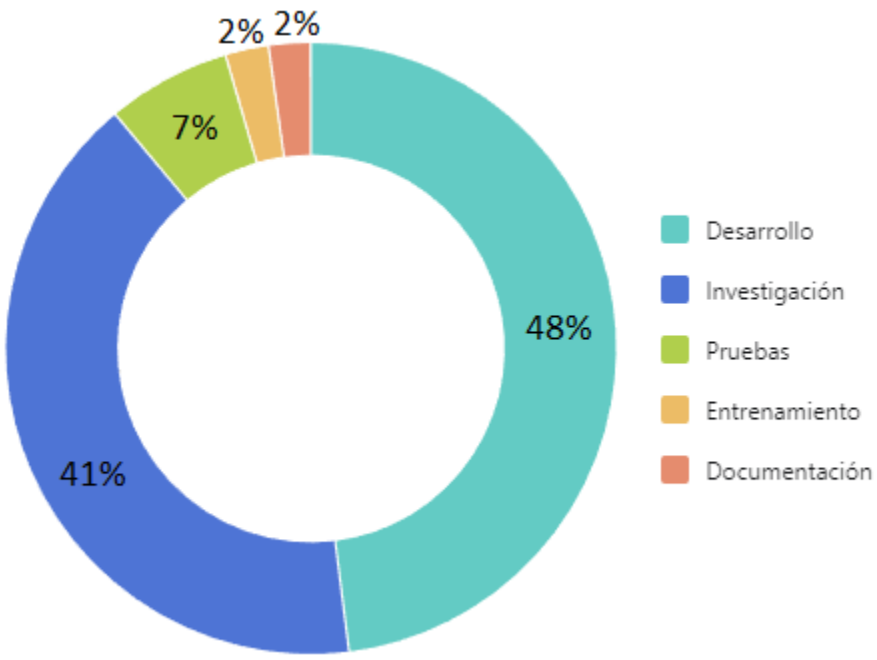

Figure 1.3.5: Graph of percentages of time spent on each type of task

Conclusions and future work

As a conclusion of the project, it can be seen that the final application meets most of the proposed requirements, as explained in the previous sections. This application is a first step to improve the quality of life of augmented reality glasses users who suffer from different vision problems. Even so, there are certain concepts that would be interesting to implement in the future in order to improve the application: increase the brightness through software processing, correctly pass the application to augmented reality glasses, optimize the times as much as possible and retrain all gestures.

Currently, in order to change the brightness, the application must be granted permissions to change any system setting, in addition to the fact that the brightness being changed is that of the device and not that of the image being viewed. Although we have tried to minimize this problem by briefly explaining to the users why this permission is used and allowing to use the application even if it is not granted, it would be interesting to look for an alternative solution and because the process of finding a software solution to this problem requires substantial effort it is proposed as future work. By software processing of the image would not require granting so many permissions to the application, which would reduce the ethical problems associated with it and even if it can be implemented could serve as a basis for adding other types of filters to the image, which can help users of the application.

As explained several times in this document, the ultimate goal of this project is to use the application in augmented reality glasses. This project is only the first phase of this process as it simulates the operation of the glasses using a smartphone. As a future work, using this application as a base, it is proposed to test its operation in augmented reality glasses and change everything necessary to optimize the user experience in these glasses.

In addition, as this is an application that is analyzing gestures at runtime, it is likely that adding new features to the application will lead to a deterioration in its performance. It is for this very reason that a thorough review of the implementation should be carried out, to ensure that it has been developed in the most optimal way possible, focusing

especially on new additions and how these may generate certain incompatibilities with the functionalities presented in this project.

Finally, there will always be a margin of error when detecting gestures as it is done by an AI and will depend on the training received, to provide better or worse results. In this project the number of data that have been used are quite limited and have been used mainly for a standard use of the application, however, there are still some failures in the detection that could be annoying for a person who will use the application daily and that can be solved with a training with much more test data than those that have been used in the project.

Contribuciones Personales

Daniel Castro López

En primer lugar, realizó de forma paralela una investigación del estado del arte junto con sus compañeros. Debido a que la temática del proyecto cambió en los primeros meses, sirvió para familiarizarse con las distintas tecnologías a usar y luego se compartió dicha información en reuniones para proponer distintas soluciones a los problemas que se planteaban en un inicio (como era el propio tema del proyecto y las tecnologías que se deberían aplicar). Dentro de esta investigación se centró en sopesar la diversidad de opciones de inteligencias artificiales que ofrecían la posibilidad de implementar reconocimiento de gestos y su funcionamiento. Tras haber investigado junto con todos sus compañeros y decidir usar una aplicación de Python ya hecha para la detección de gestos, tal como se decidió en una reunión, realizó su parte correspondiente para el entrenamiento de los distintos gestos, usando la aplicación de python generó un fichero csv con los datos de entrenamiento asociados a cada uno de los gestos (100 casos por cada gesto y por cada mano) y subió el fichero a un repositorio común.

Realizó una primera aproximación al uso de la cámara de Android a partir del proyecto de Python inicial, para ello utilizó una aplicación para poder conectar de forma remota el dispositivo Android al ordenador a través de una conexión a Internet y desde Python capturó la información de la cámara del teléfono. Esta primera aproximación tenía como objetivo medir la viabilidad de utilizar un dispositivo Android para detección de gestos en tiempo real, con conocimiento de que su rendimiento en dichas condiciones iba a ser inferior al que se iba a obtener en la aplicación final, debido a que la conexión entre el algoritmo de python y la cámara se realizaban con mayor latencia debido a la conexión remota.

Investigó sobre el proceso de transmisión de información entre la aplicación Android y la inteligencia artificial con la ayuda de Iñigo Rolando, para ello se barajaron varias posibilidades en función de las limitaciones que tenían las bibliotecas utilizadas. El principal conflicto era la diferencia de funcionamiento entre el formato de las listas de Java y Python. La solución que considero más viable era tras la detección de las

landmarks en Android, enviar la información a Python en forma de json para en Python invertir el proceso y recuperar la lista de landmarks.

Para realizar las funcionalidades básicas de la aplicación, realizó una investigación sobre el uso de las distintas apis que ofrecía Android para capturar la cámara del dispositivo. Las opciones se redujeron al uso de Camera, Camera 2 o Camera X y optó por el uso de Camera X debido a que la primera opción estaba obsoleta, y Camera X era una implementación de Camera 2 más simple y moderna que cumplía con las características necesarias para la implementación de la aplicación. Entre las opciones que ofrece Camera X, destacan la posibilidad de editar el zoom y contraste a través de distintos parámetros que podían modificarse de forma satisfactoria desde Java. Por otro lado, la api no incluía un atributo para modificar el brillo, por lo que se barajaron distintas posibilidades, como mostrar al usuario una imagen tras ser procesada pero con el brillo modificado, pero esta idea quedó rápidamente descartada debido al aumento drástico de complejidad al ser necesario el reestructurado de la aplicación para un correcto funcionamiento. Además, aunque se procesara la imagen y se modificara el brillo de la misma, no lo haría el dispositivo por lo que el resultado era incierto y conlleva una disminución del rendimiento de la aplicación. Finalmente la opción más realista era la modificación del brillo propio del dispositivo y no el de la cámara, esto derivó en que hubiera que tener en cuenta el uso de un nuevo permiso en la aplicación y sobre el cúal se debe informar al usuario y solicitar su consentimiento al respecto. También estableció un factor que consideró adecuado para el aumento y disminución del brillo con el objetivo de que el cambio fuera notable para el usuario, pero de forma lo más gradual posible.

En adición, realizó junto con Víctor de la Fuente una limpieza y reestructuración del código completo de la aplicación, ya que quedaban varias funciones no utilizadas, o con posibilidad de ser optimizadas. También eliminó y editó varios archivos del código en python con el objetivo de que solamente quedarán los archivos estrictamente relacionados con el reconocimiento de gestos y no con el entrenamiento de la inteligencia artificial, con lo que se consiguió reducir el tamaño de la aplicación.

Realizó la implementación del manual de usuario de la aplicación que es accesible mediante un botón ubicado en la esquina superior derecha realizado por Víctor de la

Fuente y que tiene como objetivo presentar al usuario de forma simple el funcionamiento de la aplicación.

Realizó las pruebas asociadas a la iteración 2 descritas en el plan de pruebas con la ayuda de Víctor de la Fuente.

Asociado a la funcionalidad del brillo, como se comentó anteriormente, era necesaria la implementación de un nuevo diálogo para advertir al usuario sobre la necesidad de permiso para la edición de los ajustes del dispositivo con el fin de regular el brillo y una breve explicación sobre qué acciones debe realizar el usuario para otorgarlo. Además, tras este diálogo, implementó una función que abriera una instancia de los ajustes para facilitar al usuario conceder dicho permiso. Si el usuario no concede el permiso puede seguir usando la aplicación, pero con la funcionalidad asociada al brillo limitada, por lo que también implementó un mensaje para informar al usuario de que no puede modificar el brillo si trata de modificarlo sin haberle otorgado permiso a la aplicación primero.

Junto al resto de sus compañeros, realizó una serie de pruebas para verificar que el comportamiento de la aplicación era el deseado y que no se produjeran fallos inesperados. Además, también realizó la corrección de errores asociados a dichas pruebas.

Sonia Marcos de Benito

En primer lugar, realizó de forma paralela una investigación del estado del arte junto con sus compañeros. Debido a que la temática del proyecto cambió en los primeros meses, sirvió para familiarizarse con las distintas tecnologías a usar y luego se compartió dicha información en reuniones para proponer distintas soluciones a los problemas que se planteaban en un inicio (como era el propio tema del proyecto y las tecnologías que se deberían aplicar). Dentro de esta investigación incidió sobretodo en el estudio de las gafas de realidad aumentada que se iban usar, para conocer qué requisitos software y hardware debía cumplir la aplicación. Tras haber investigado junto con todos sus compañeros y decidir usar una aplicación de Python ya hecha para la detección de gestos, tal como se decidió en una reunión, realizó su parte correspondiente para el entrenamiento de los distintos gestos. Usando la aplicación de Python generó un fichero csv con los datos de entrenamiento asociados a cada uno de los gestos (100 casos por cada gesto y por cada mano) y subió el fichero a un repositorio común.

Al inicio de la primera iteración, durante la instalación de las librerías necesarias para realizar la compilación del proyecto, surgió un problema, ya que Chaquopy no contaba con MediaPipe entre las librerías que se podían instalar. Realizó una investigación sobre las alternativas existentes adaptables al proyecto que no requiriesen una remodelación significativa del mismo. Junto a su compañero Víctor de la Fuente, investigó una posible forma de modificar Chaquopy para poder añadir MediaPipe entre sus librerías disponibles.

Tras acabar la investigación, junto con Víctor de la Fuente, decidió que la mejor forma de solucionarlo sería añadir la librería MediaPipe al proyecto Android, que se encargaría de procesar las imágenes y enviar el listado de landmarks a la aplicación Python. Esto a su vez, provocó varios problemas derivados ya que fue necesario analizar de manera exhaustiva el código de MediaPipe y conocer cómo se guardaban y generaban los valores de los landmarks para poder enviarlos a la aplicación Python.

Una vez implementado el listener de imágenes, se procedió a la implementación de la funcionalidad para los tres gestos de la aplicación (brillo, zoom y contraste), cada uno

de ellos con sus respectivos casos de aumentar y disminuir. Desarrolló la funcionalidad de aumentar y disminuir el zoom, gracias a que la librería utilizada CameraX contaba con un parámetro existente para modificarlo. Tan solo fue necesario aumentar o disminuir el valor de ese parámetro y definir un valor adecuado para el factor de aumento o disminución, de manera que cada cambio fuera apreciable, pero permitiera ajustar esta configuración de la mejor manera posible.

Realizó las pruebas asociadas a la primera iteración especificadas en el plan de pruebas junto con su compañero Íñigo Rolando para comprobar el correcto desempeño de la funcionalidad desarrollada en esta iteración.

Tras acabar de hacer las pruebas de la primera iteración se detectó que la pantalla del dispositivo Android se apagaba debido a la inactividad. Lo cual no era un comportamiento adecuado para nuestra aplicación ya que está diseñada para que el usuario no tenga que intervenir durante su funcionamiento tocando el dispositivo.

Por ello era necesario configurar la aplicación para que se mantuviera encendida. Android contempla el uso de una flag específica que conlleva que el dispositivo no se bloquee, aunque pasen largos periodos de inactividad. Esta flag se activa al inicio del flujo de la aplicación Android y permite que siga funcionando con normalidad.

Junto con su compañero Íñigo Rolando se encargó de realizar las correcciones asociadas a las distintas pruebas que se hicieron en la segunda iteración, centrándose sobretodo en las correcciones asociadas al narrador (como por ejemplo, añadiendo el narrador en algunos nuevos errores propios de esta iteración).

Tras corregir los errores de la segunda iteración, se decidió que sería útil implementar una nueva funcionalidad asociada a un nuevo gesto, que se encargase de reiniciar la configuración a los valores predeterminados de brillo, contraste y zoom. Además eligió unos valores predeterminados que consideró óptimos. Para ello utilizó el mismo flujo que el resto de gestos de la aplicación.

Finalmente, junto con todos sus compañeros se encargó de realizar las pruebas asociadas a la tercera y última iteración para comprobar que la aplicación funcionaba de

manera correcta en base a los requisitos iniciales y se encargó de pequeños arreglos que surgieron en base a estas pruebas.

Víctor de la Fuente Sabalete

En primer lugar, realizó de forma paralela una investigación del estado del arte junto con sus compañeros. Debido a que la temática del proyecto cambió en los primeros meses, sirvió para familiarizarse con las distintas tecnologías a usar y luego se compartió dicha información en reuniones para proponer distintas soluciones a los problemas que se planteaban en un inicio (como era el propio tema del proyecto y las tecnologías que se deberían aplicar). Tras haber investigado junto con todos sus compañeros y decidir usar una aplicación de python ya hecha para la detección de gesto, tal como se decidió en una reunión, realizó su parte correspondiente para el entrenamiento de los distintos gestos, usando la aplicación de python generó un fichero csv con los datos de entrenamiento asociados a cada uno de los gestos (100 casos por cada gesto y por cada mano) y subió el fichero a un repositorio común. Una vez subidos todos los ficheros fue el encargado de unificarlos y entrenar la red neuronal con el fichero unificado.

Al comenzar la primera iteración, durante la instalación de las librerías necesarias para realizar la compilación del proyecto, surgió un problema, ya que Chaquopy no contaba con MediaPipe entre sus librerías instalables.

Realizó una investigación sobre las alternativas existentes adaptables al proyecto que no requirieran una remodelación significativa de la estructura del proyecto. Junto a su compañera Sonia Marcos investigó una posible forma de modificar Chaquopy para poder añadir MediaPipe a sus librerías instalables.

Tras concluir la investigación, junto con Sonia Marcos, decidió que la solución más viable sería añadir la librería MediaPipe al proyecto Android, que se encargaría de procesar las imágenes y enviar el listado de landmarks a la aplicación Python.

Esto a su vez generó varios problemas derivados ya que fue necesario analizar de manera exhaustiva el código de MediaPipe y conocer cómo se guardaban y generaban los valores de los landmarks para poder enviarlos a la aplicación Python.

Conocimiento el funcionamiento de MediaPipe para la creación de los landmarks y convertidos los mismos en un objeto JSON para ser utilizados en Android, implementó con la ayuda de Íñigo Rolando un listener en la aplicación Android que envía a cada fotograma este objeto con la información de la imagen al código de Python.

Una vez implementado el listener se procedió a la implementación de la funcionalidad para los tres gestos existentes en ese momento en la aplicación, cada uno de ellos con sus respectivos casos de aumentar y disminuir.

Desarrolló la funcionalidad de aumentar y disminuir el contraste, gracias a que la librería utilizada CameraX contaba con un parámetro existente para modificar el contraste, tan solo fue necesario aumentar o disminuir el valor de ese parámetro.

También estableció un valor adecuado para el factor de aumento o disminución, de manera que cada cambio fuera apreciable, pero permitiera ajustar esta configuración de una manera óptima.

Paralelamente a la realización de las pruebas de la iteración 1, realizó junto con Daniel Castro una limpieza y reestructuración del código completo de la aplicación, ya que quedaban varias funciones no utilizadas, o con posibilidad de ser optimizadas. También eliminó y editó varios archivos del código en python con el objetivo de que solamente quedarán los archivos estrictamente relacionados con el reconocimiento de gestos y no con el entrenamiento de la inteligencia artificial, con lo que se consiguió reducir el tamaño de la aplicación.

Realizó la creación de un botón situado en la esquina superior derecha. El botón se añadió con un fondo transparente y con un tamaño adecuado para que pudiera ser visto con facilidad, pero lo suficientemente pequeño para no entorpecer la visión del usuario. Al pulsar sobre el botón se redirige al usuario a una pantalla donde se muestra el manual de usuario creado por Daniel Castro, una vez este estuviera creado, leyéndolo directamente de un archivo para evitar tener que modificar el código de esta pantalla ante una posible actualización del manual.

Realizó las pruebas asociadas a la iteración 2 descritas en el plan de pruebas con la ayuda de Víctor de la Fuente.

Durante el desarrollo de la tercera iteración detectó un problema con uso del narrador en español en caso de que este no estuviera ya instalado en el dispositivo. Por ello, implementó un aviso para informar al usuario en caso de que en el dispositivo no se encuentre instalado el narrador en español de que esta funcionalidad se encontrará deshabilitada mientras no se instalen los paquetes necesarios. Tras la confirmación de este aviso el usuario puede seguir usando la aplicación con el resto de funcionalidades disponibles.

Íñigo Rolando García

En primer lugar, realizó de forma paralela una investigación del estado del arte junto con sus compañeros. Debido a que la temática del proyecto cambió en los primeros meses, sirvió para familiarizarse con las distintas tecnologías a usar y luego se compartió dicha información en reuniones para proponer distintas soluciones a los problemas que se planteaban en un inicio (como era el propio tema del proyecto y las tecnologías que se deberían aplicar). Dentro de esta investigación se centró sobre todo en las distintas librerías existentes en Android que se necesitarían para cumplir con éxito los objetivos iniciales de la aplicación y que luego se encargaría de integrar en el propio proyecto de Android. Tras haber investigado junto con todos sus compañeros y decidir usar una aplicación de Python ya hecha para la detección de gestos, tal como se decidió en una reunión, realizó su parte correspondiente para el entrenamiento de los distintos gestos, usando la aplicación de Python generó un fichero csv con los datos de entrenamiento asociados a cada uno de los gestos (100 casos por cada gesto y por cada mano) y subió el fichero a un repositorio común.

Tras esto se encargó de la creación del proyecto Android teniendo en cuenta las características hardware que se habían establecido en las reuniones anteriores (como por ejemplo el número de versión de Android) y de subir este proyecto al repositorio compartido en GitHub de manera correcta (añadiendo todos los ficheros claves de la aplicación Android como puede ser el fichero build.gradle). Además, se encargó de añadir a esta aplicación las librerías necesarias para el funcionamiento de la misma: Chacuopy y MediaPipe. Se encargó de configurar estas librerías para la versión de Android que se estaba usando y en el caso de Chacuopy estudió de manera exhaustiva el cómo se configuraban las distintas librerías de Python necesarias para la conexión entre la aplicación Android y la aplicación Python. Esto supuso un gran problema debido a que Chacuopy usa su propio repositorio de librerías Python y esto genera ciertas incompatibilidades entre algunas librerías. Al añadir esta librería y entender bien el funcionamiento de la misma también realizó una investigación en cómo integrar la aplicación de detección de gestos en Python con la aplicación Android y junto con su compañero Daniel Castro se implementaron distintas soluciones para poder llevar a

cabo esta conexión, centrándose en gran medida en cómo poder pasar el objeto de landmarks obtenido del procesamiento de la imagen de MediaPipe al código de Python.

Una vez entendido de manera correcta el funcionamiento de este objeto landmarks y habiendo implementado la forma correcta de pasarlo a la IA en Python (mediante un formato JSON con la librería GSON), implementó con la ayuda de Víctor de la Fuente un listener en la aplicación Android que envía a cada fotograma este objeto con la información de la imagen al código de Python.

Realizó las pruebas asociadas a la primera iteración especificadas en el plan de pruebas junto con su compañera Sonia Marcos para determinar el correcto desempeño de la funcionalidad desarrollada en esta iteración.

Tras esto se encargó de la implementación del narrador en la aplicación, tras una pequeña investigación inicial decidió usar TextToSpeech, una funcionalidad añadida directamente en Android y que podría aplicar el narrador de manera muy sencilla. Hizo que este narrador se ejecutará cada vez que se detectara un gesto y cada vez que se obtuviera un error en la aplicación.

Mediante el uso de SharedPreferences añadió la posibilidad de guardar los valores de zoom, brillo y contraste que cambia el usuario para que cuando la aplicación se abra de nuevo se puedan cargar estos valores. Esto haría que la aplicación fuera mucho más útil para el usuario ya que no tendrá que cambiar las características en cada uso.Para ello primero intentó guardar estas preferencias sólo cuando se cerrara la aplicación pero debido a que esto daba ciertos problemas al cerrarse de forma inesperada finalmente implementó este guardado cada vez que se cambiaba el valor de uno de los parámetros mencionados anteriormente. Además, en caso de haber algún error al cargar la información añadió un valor por defecto a todas estas características.

Junto con su compañera Sonia Marcos se encargó de realizar las pequeñas correcciones asociadas a las distintas pruebas que se hicieron en la segunda iteración, centrándose sobretodo en las correcciones asociadas al narrador (como por ejemplo, añadiendo el narrador en algunos nuevos errores propios de esta iteración).

Debido a los objetivos éticos que se propusieron al inicio del proyecto se encargó de buscar una solución a las capturas de pantalla dentro de la aplicación. Android permite añadir o eliminar permisos mediante la activación de distintas flags por lo que se encargó de añadir esta flag a la aplicación Android para que no se permitieran hacer capturas ni grabar la pantalla mientras estuviera esta aplicación en primer plano.

Por último, junto con todos sus compañeros se encargó de realizar las pruebas asociadas a la tercera y última iteración para corroborar que la aplicación funcionaba de manera correcta en base a los requisitos iniciales y se encargó de pequeños arreglos que surgieron en base a estas pruebas.

Bibliografía

[1] *¿Por qué utilizar la Metodología kanban?método básico y Principios Kanban para ayudarle a empezar.* Metodología Kanban | Kanban Tool. (n.d.). https://kanbantool.com/es/metodologia-kanban

[2] Asana. (n.d.-a). *Gestiona el Trabajo, Los Proyectos y las tareas de tu equipo en línea • asana*. Asana. https://asana.com/es

[3] *Rolando, I., Castro, D., de la Fuente, V., & Marcos, S. (n.d.). Inigorolando/TFG_DMAE. GitHub. https://github.com/InigoRolando/TFG_DMAE*

[4] *Porzi, L., Messelodi, S., Modena, C. M., & Ricci, E. (n.d.). A smart watch-based gesture recognition system for assisting people ... https://www.researchgate.net/profile/Lorenzo-Porzi/publication/262233058_A_smart_watch-based_gesture_recognition_system_for_assisting_people_with_visual_impairments/links/54f082a20cf24eb87940ba1e/A-smart-watch-based-gesture-recognition-system-for-assisting-people-with-visual-impairments.pdf*

[5] *Balasuriya, B. K., Lokuhettiarachchi, N. P., Ranasinghe, A. R. M. D. N., Shiwantha, K. D. C., & Jayawardena, C. (n.d.). Learning platform for visually impaired children through artificial ... https://www.researchgate.net/profile/Chandimal-Jayawardena/publication/323281352_Learning_platform_for_visually_impaired_children_through_artificial_intelligence_and_computer_vision/links/63681d5c431b1f530076e6c9/Learning-platform-for-visually-impaired-children-through-artificial-intelligence-and-computer-vision.pdf?origin=publication_detail*

[6] Tejashri J. Joshi, N. Z. Tarapore, Shiva Kumar, & Vivek Mohile. (2015, June 21). Static hand gesture recognition using an Android device - researchgate. https://www.researchgate.net/profile/Vivek-Mohile/publication/281415512_Static_Hand_Gesture_Recognition_using_an_Android_Device/links/619cded1d7d1af224b1b33a4/Static-Hand-Gesture-Recognition-using-an-Android-Device.pdf

[7] Wang, T., Wan, Y., Peers, C., Sun, J., & Zhou, C. (2022, July 20). *Vision-based gesture tracking for teleoperating mobile manipulators*. White Rose Research Online. https://eprints.whiterose.ac.uk/189808/

[8] authors, A., Yang, C.-Y., Additional informationFundingThis work was supported by the Ministry of Science and Technology, & ReferencesRelay. [Online] Available: https://tutorials.webduino.io/zh-tw/docs/basic/component/relay.html. May 2022a. [Google Scholar]Alemuda. (2022, May 1). *Smart control of home appliances using hand gesture recognition in an IOT-enabled system*. Taylor & Francis. https://www.tandfonline.com/doi/full/10.1080/08839514.2023.2176607

[9] Hand landmarks detection guide. (s. f.). Google Developers. https://developers.google.com/MediaPipe/solutions/vision/hand_landmarker

[10] colaboradores de Wikipedia. (2023). Filtro de Kalman. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Filtro_de_Kalman

[11] *CMU-Perceptual-Computing-Lab. (n.d.). CMU-Perceptual-Computing-Lab/openpose: OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation. GitHub. https://github.com/CMU-Perceptual-Computing-Lab/openpose*

[12] OpenCV. (2023, 26 abril). Home - OpenCV. https://opencv.org/

[13] Descripción general de CameraX. (s. f.). Android Developers. https://developer.android.com/training/camerax?hl=es-419

[14] Smith, M. (s. f.). The easiest way to use Python in an Android app. Chaquopy. https://chaquo.com/chaquopy/

[15] TensorFlow. (s. f.). TensorFlow. https://www.tensorflow.org/?hl=es-419

[16] *Software para la gestión de riesgos y controles: SoftExpert Riesgo - SoftExpert: Soluciones Para Excelencia en la gestión y conformidad empresarial*. SoftExpert. (2021, March 25). From https://www.softexpert.com.es/todos-los-productos/gestion-riesgos-controles/

[17] YouTube. (2022). *Custom Hand Gesture Recognition with Hand Landmarks Using Google's MediaPipe + OpenCV in Python*. *YouTube*. https://www.youtube.com/watch?v=a99p_fAr6e4.

[18] Wikimedia Foundation. (2023, March 17). *RGB*. Wikipedia. https://es.wikipedia.org/wiki/RGB

[19] Kinivi. (n.d.). *Kinivi/hand-gesture-recognition-MediaPipe: This is a sample program that recognizes hand signs and finger gestures with a simple MLP using the detected key points. Handpose is estimated using MediaPipe.* GitHub. Retrieved May 4, 2023, from https://github.com/kinivi/hand-gesture-recognition-MediaPipe

[20] *Google. (n.d.). DROIDCAM - webcam for PC - aplicaciones en google play. Google. https://play.google.com/store/apps/details?id=com.dev47apps.droidcam&hl=es&gl=US&pli=1*

[21] *Python on Android*. Android - Python Wiki. (n.d.). From https://wiki.python.org/moin/Android

[22] Chaquo. (n.d.). *MediaPipe package · issue #479 · Chaquo/Chaquopy*. GitHub. From https://github.com/chaquo/chaquopy/issues/479

[23] Google. (n.d.). *Hand landmarks detection guide | MediaPipe | google developers*. From https://developers.google.com/MediaPipe/solutions/vision/hand_landmarker

[24] Google. (n.d.). *MediaPipe/mainactivity.java at master · google/MediaPipe*. GitHub. From https://github.com/google/mediapipe/blob/master/mediapipe/examples/android/solutions/hands/src/main/java/com/google/mediapipe/examples/hands/MainActivity.java

[25] Google. (n.d.). *Hand landmarks detection guide for android | MediaPipe | google developers*. From https://developers.google.com/MediaPipe/solutions/vision/hand_landmarker/android

APÉNDICES

Apéndice A - Pruebas primera iteración

ID	NOMBRE DE LA PRUEBA	Resultado	Observaciones

1	Gesto de zoom	OK	No se aplica la condición del narrador
2	Gesto de brillo	KO	No se aplica la condición del narrador. No se llega al 90%, se arreglará en la última iteración al añadir el nuevo gesto.
3	Gesto de contraste	KO	No se aplica la condición del narrador. No se llega al 90%, se arreglará en la última iteración al añadir el nuevo gesto.
4	Gesto de aumentar	OK	No se aplica la condición del narrador
5	Gesto de disminuir	OK	No se aplica la condición del narrador
6	Gesto de restablecer	-	No aplica
7	Aumentar zoom	OK	
10	Disminuir zoom	OK	
8	Aumentar brillo	OK	
11	Disminuir brillo	OK	
9	Aumentar contraste	OK	
12	Disminuir contraste	OK	
13	Aumentar zoom más del máximo	OK	No se aplica la condición del narrador
16	Disminuir zoom más del mínimo	OK	No se aplica la condición del narrador
14	Aumentar brillo más del máximo	OK	No se aplica la condición del narrador
17	Disminuir brillo más del mínimo	OK	No se aplica la condición del narrador
15	Aumentar contraste más del máximo	OK	No se aplica la condición del narrador
18	Disminuir contraste más	OK	No se aplica la condición del narrador

	del mínimo		
19	Restablecer la configuración completa	-	No aplica
20	Restablecer la configuración (zoom)	-	No aplica
21	Restablecer la configuración (brillo)	-	No aplica
22	Restablecer la configuración (contraste)	-	No aplica
23	Guardar configuración anterior (todos los parámetros)	-	No aplica
24	Guardar configuración anterior (zoom)	-	No aplica
25	Guardar configuración anterior (contraste)	-	No aplica
26	Guardar configuración anterior (brillo)	-	No aplica
27	Modal permiso de cámara	-	No aplica
28	Modal permiso brillo	-	No aplica
29	Rechazo permiso cámara	-	No aplica
30	Rechazo permiso brillo	-	No aplica
31	Captura de pantalla	-	No aplica
32	Grabar pantalla	-	No aplica
33	Bloqueo automático	-	No aplica

34	Rotar pantalla	-	No aplica
35	Manual de usuario	-	No aplica

Apéndice B - Pruebas segunda iteración

ID	NOMBRE DE LA PRUEBA	Resultado	Observaciones
1	Gesto de zoom	OK	
2	Gesto de brillo	-	No aplica ya que se detectó un error en la iteración anterior y no se arreglará hasta la próxima.
3	Gesto de contraste	-	No aplica ya que se detectó un error en la iteración anterior y no se arreglará hasta la próxima.
4	Gesto de aumentar	OK	
5	Gesto de disminuir	OK	
6	Gesto de restablecer	OK	
7	Aumentar zoom	OK	
10	Disminuir zoom	OK	
8	Aumentar brillo	OK	
11	Disminuir brillo	OK	
9	Aumentar contraste	OK	
12	Disminuir contraste	OK	
13	Aumentar zoom más del máximo	OK	
16	Disminuir zoom más del mínimo	OK	
14	Aumentar brillo más del máximo	OK	

17	Disminuir brillo más del mínimo	OK	
15	Aumentar contraste más del máximo	OK	
18	Disminuir contraste más del mínimo	OK	
19	Restablecer la configuración completa	OK	
20	Restablecer la configuración (zoom)	OK	
21	Restablecer la configuración (brillo)	OK	
22	Restablecer la configuración (contraste)	OK	
23	Guardar configuración anterior (todos los parámetros)	OK	
24	Guardar configuración anterior (zoom)	OK	
25	Guardar configuración anterior (contraste)	OK	
26	Guardar configuración anterior (brillo)	OK	
27	Modal permiso de cámara	-	No aplica
28	Modal permiso brillo	-	No aplica
29	Rechazo permiso cámara	-	No aplica

30	Rechazo permiso brillo	-	No aplica
31	Captura de pantalla	-	No aplica
32	Grabar pantalla	-	No aplica
33	Bloqueo automático	OK	
34	Rotar pantalla	OK	
35	Manual de usuario	OK	

Apéndice C - Pruebas tercera iteración

ID	NOMBRE DE LA PRUEBA	Resultado	Observaciones
1	Gesto de zoom	OK	
2	Gesto de brillo	OK	
3	Gesto de contraste	OK	
4	Gesto de aumentar	OK	
5	Gesto de disminuir	OK	
6	Gesto de restablecer	OK	
7	Aumentar zoom	OK	
10	Disminuir zoom	OK	
8	Aumentar brillo	OK	
11	Disminuir brillo	OK	
9	Aumentar contraste	OK	
12	Disminuir contraste	OK	
13	Aumentar zoom más del máximo	OK	
16	Disminuir zoom más del mínimo	OK	
14	Aumentar brillo más del máximo	OK	
17	Disminuir brillo más del mínimo	OK	
15	Aumentar contraste más del máximo	OK	

18	Disminuir contraste más del mínimo	OK	
19	Restablecer la configuración completa	OK	
20	Restablecer la configuración (zoom)	OK	
21	Restablecer la configuración (brillo)	OK	
22	Restablecer la configuración (contraste)	OK	
23	Guardar configuración anterior (todos los parámetros)	OK	
24	Guardar configuración anterior (zoom)	OK	
25	Guardar configuración anterior (contraste)	OK	
26	Guardar configuración anterior (brillo)	OK	
27	Modal permiso de cámara	OK	
28	Modal permiso brillo	OK	
29	Rechazo permiso cámara	OK	
30	Rechazo permiso brillo	OK	
31	Captura de pantalla	OK	
32	Grabar pantalla	OK	

33	Bloqueo automático	OK	
34	Rotar pantalla	OK	
35	Manual de usuario	OK	