Desarrollo de una aplicación de Android para la identificación de setas y hongos basada en el reconocimiento de imágenes por machine learning

GRADO EN INGENIERÍA INFORMÁTICA

Realizador por:

Álvaro Trius Béjar

Especialidad: Ingeniería del Software

Director: Silverio Martínez-Fernández

Codirectora: Cristina Gómez Seoane

Barcelona 2021

Resumen

Machine learning es una rama de la IA que consiste en el desarrollo de algoritmos capaces de aprender relaciones entre ideas y de mejorar a base de entrenamiento continuo. Debido a los pesados costes de recursos y tiempo que esto requiere, uno de los grandes desafíos de este campo es poder integrar estos algoritmos en un dispositivo móvil. En este trabajo nos ponemos el desafío de realizar esta integración, poniendo como objetivo el desarrollo de una aplicación Android capaz de identificar clases de setas y hongos mediante reconocimiento de imágenes usando redes neuronales.

Resum

Machine learning és una branca de la IA que consisteix en el desenvolupament d'algoritmes capaços d'aprendre relacions entre idees i de millorar a base d'entrenament continu. A causa dels pesats costos de recursos i temps que això requereix, un dels grans desafiaments d'aquest camp és poder integrar aquests algoritmes en un dispositiu mòbil. En aquest treball ens atrevim a realitzar aquesta integració, posant com a objectiu el desenvolupament d'una aplicació Android capaç d'identificar classes de bolets i fongs mitjançant reconeixement d'imatges usant xarxes neuronals.

Abstract

Machine learning is a branch of AI that consists of the development of algorithms capable of learning relationships between ideas and improving based on continuous training. Due to the heavy costs of resources and time that this requires, one of the great challenges in this field is to be able to integrate these algorithms into mobile devices. In this work, we challenge ourselves to carry out this integration, setting the objective of developing an Android application capable of identifying classes of mushrooms and fungi through image recognition using neural networks.

Índice

Índice de tablas

Índice de ilustraciones

1. INTRODUCCIÓN

El trabajo de fin de grado (TFG) suele ser la última asignatura que se realiza al cursar el grado de Ingeniería Informática en la Facultad de Informática de Barcelona (FIB). En este trabajo se pone a prueba todo el conjunto de conocimientos y prácticas adquiridos durante la carrera acorde a la especialidad cursada. Existen diversas modalidades para realizar el trabajo, pero en mi caso, como estudiante de la especialidad de Ingeniería del Software, he decidido realizar un proyecto dentro del marco de la UPC (modalidad A) que consistirá en realizar una *Aplicación de software para la identificación de setas y hongos basada en reconocimiento de imágenes por machine learning*. Considero que este trabajo me permitirá poner en práctica y demostrar los conocimientos adquiridos durante la carrera en esta especialidad.

1.1. CONTEXTUALIZACIÓN DEL PROBLEMA

1.1.1 Sobre la recogida de setas

Hoy en día la recogida de setas se ha convertido para muchos en una afición o una distracción inesperada durante una excursión pero, a causa de esto, no es infrecuente oír noticias de paseantes o recolectores de setas que, sin tener mucha experiencia, deciden consumirlas y acaban intoxicados. Así, tal y como dice el artículo de El Heraldo, cada año se producen 400 intoxicaciones por hongos en España de las cuales al menos una o dos acaban en muerte **[1]**.

La solución a este problema, por un lado, pasa por incentivar a los excursionistas a que tomen conciencia de los riesgos que implica recolectar y consumir hongos desconocidos. Por otro lado, también pasa por proporcionarles las herramientas necesarias para saber identificarlos adecuadamente.

Nuestra primera intención en este TFG es colaborar hacia la segunda parte de la solución. En concreto, sabiendo que hoy en día el uso del teléfono móvil y las aplicaciones está tan extendido, creemos que podríamos desarrollar una aplicación que, a partir de una foto, fuera capaz de identificar con precisión de qué clase de hongo se trata e informar al usuario de sus características. Así proveeríamos a los usuarios de una herramienta de más para poder disfrutar de esta afición sin someterse a riesgos innecesarios.

Aparte de este problema, que consideramos el más grave, también consideramos que siendo la búsqueda de setas una afición tan saludable y entretenida es una lástima que no haya tanta gente interesada en ella. Nuestra idea hacer que este hobby sea algo más atractivo.

1.1.2 Sobre machine learning

Como ya fue indicado en el resumen, machine learning es una de las ramas de la IA (Inteligencia Artificial) que nos parecen más atípicas de cara al diseño y el desarrollo del software. Porque, así como se indica en el artículo *Towards Guidelines for Assessing Qualities of Machine Learning Systems*, su funcionalidad no queda tan solo establecida por el programador, sino por los datos de entrada que recibe **[2]**. Esto hace que sea una tecnología software con unos requisitos no funcionales atípicos que se acaba convirtiendo en un auténtico reto para desarrolladores de software.

Nuestra segunda intención con este proyecto es intentar ayudar a resolver el problema erigiéndolo como una guía de ejemplo para otros desarrolladores de este campo. Así, a través de este trabajo no sólo llevaremos las prácticas clásicas de desarrollo de software, sino también las específicas para el desarrollo de aprendizajes profundos, entre los cuales nos enfrentaremos al problema de cómo medir los requisitos no funcionales en aplicaciones de machine learning.

1.2. DESCRIPCIÓN DEL PROYECTO

El proyecto a implementar es una aplicación móvil que será capaz de identificar la clase de seta a partir de una foto. Para ello utilizará una Inteligencia Artificial de tipo machine learning que podrá mejorar con el tiempo en su precisión para identificar setas y hongos.

Tendrá las siguientes funcionalidades:

1) Reconocimiento de setas a partir de imagen: la funcionalidad principal del proyecto. A través de una imagen de una seta la aplicación será capaz de identificar cuál es de entre las clases que conozca.

2) Enciclopedia de setas: Al reconocer una seta la aplicación proporcionará al usuario de información completa acerca de ésta, como las características, algunas curiosidades y, sobre todo, si es comestible o no. Esta enciclopedia también estará al alcance del usuario como funcionalidad aparte.

3) Capacidad de auto aprendizaje en caso de fallo: El usuario podrá corregir la identificación de la IA. Al hacerlo, esos datos serán incorporados al dataset de aprendizaje. Puesto que una ampliación del número de datos en el dataset, colaborará hacia la mejora de la precisión del modelo.

4) Gamificación: Aunque esto es un objetivo más secundario, con tal de incentivar al usuario a utilizar esta aplicación utilizaremos técnicas de gamificación mediante objetivos y recompensas digitales.

1.3. TERMINOLOGÍA DEL PROYECTO

En este apartado se muestran algunos de los conceptos más importantes de este TFG:

1) Inteligencia Artificial: Es campo de la informática que combina algoritmos planteados con el propósito de crear máquinas para simular las mismas capacidades inteligentes de un ser humano. Existen muchas disciplinas y una de ellas es machine learning.

2) Machine Learning [3] [4]: Machine learning es una disciplina del campo de la Inteligencia Artificial que crea sistemas que son capaces de aprender comportamientos automáticamente. Se trata de algoritmos que a partir de un entrenamiento basado en aprender a establecer relaciones entre datos, son capaces de identificar relaciones entre datos futuros.

Por lo general hay tres tipos de algoritmos machine learning que se distinguen por el modo que realizan este proceso de aprendizaje y el resultado que se espera de ellos:

2.1) Aprendizaje supervisado: La máquina es adiestrada con el ejemplo. Al principio se le enseña a la máquina un conjunto de entradas y se le enseña cuáles son las salidas esperadas. El modelo tiene que aprender la relación entre ambas. Una vez entrenado, debería ser capaz de establecer una relación entre una entrada nunca vista y una salida. Por ejemplo: un modelo entrenado a distinguir hombres y mujeres a partir de archivos de voz de habitantes de la ciudad de Madrid debería ser capaz de hacer lo mismo con un archivo de voz de un habitante de Badajoz.

2.2) Aprendizaje no supervisado: No se esperan salidas concretas, el modelo es adiestrado a identificar patrones. A partir de esto se pueden hacer estudios y sacar conclusiones. Por ejemplo: un algoritmo que sitúa en un plano 2D los clientes de un

banco, sus datos y la propensión que tienen ingresar dinero nos permitiría analizar tendencias y, a partir de ellas, como banco, podríamos tomar decisiones.

2.3) Aprendizaje por refuerzo: La máquina es adiestrada a encontrar la combinación óptima de acciones para resolver una situación. En primer lugar, a la máquina se le enseña un contexto con unas reglas, unos objetivos y las acciones que puede tomar para alcanzar esos objetivos. Luego, la máquina mediante prueba y error debe de explorar todas las opciones hasta encontrar la combinación óptima para alcanzar su objetivo dentro de ese contexto. Por ejemplo: una máquina adiestrada a aprender a conducir un coche en un circuito virtual.

En mi caso, mi intención es utilizar una Inteligencia Artificial entrenada a identificar setas y hongos a partir de imágenes, por tanto, emplearé algoritmos de aprendizaje supervisado.

3) Deep Learning o aprendizaje profundo: Enfoque dentro de machine learning que usa estructuras lógicas que utilizan arquitecturas similares a las encontradas en los sistemas nerviosos de los mamíferos especializado en detectar características de objetos percibidos. El uso de estas arquitecturas similares a la materia gris es uno de los motivos por los que también se les llama “redes neuronales”.

4) Datos de entrenamiento y de prueba o validación: Para el desarrollo machine learning, se necesita un set de datos. Este set de datos se divide en dos: el primer set es para entrenar al modelo y el segundo para probar que funciona correctamente y da los resultados esperados.

5) Micología: La micología es la parte de la biología que se encarga del estudio de los hongos. Es un área extensa y diversificada. Entre las cosas que abarca es la idoneidad de un hongo o seta para su consumición.

6) Transfer Learning: Esta es una técnica de machine learning en la que un modelo desarrollado para desarrollar una tarea se reutiliza para el desarrollo de otra tarea. Por ejemplo: un modelo entrenado para identificar formas en la naturaleza a través de una imagen puede ser reutilizado como punto de partida para un modelo que tenga que reconocer, por ejemplo, plantas. Es una técnica muy popular puesto que entrenar un modelo desde cero requiere mucho tiempo y recursos.

1.4. ACTORES IMPLICADOS

1) Usuarios: Recoge todo el conjunto de clientes finales: son los que principalmente se verán beneficiados por esta aplicación. Serán usuarios de teléfono móvil Android interesados en conocer e identificar setas y hongos.

2) Potenciales usuarios: Teniendo en cuenta que una de las intenciones del proyecto es hacer la recogida de setas más atractiva, considero que esta aplicación les afectará en cuanto a que será mostrada como oferta tentadora.

3) Organizaciones de Micología: Para construir la aplicación será imprescindible entrenar a la IA dándole resultados correctos, para ello, utilizaremos bases de datos y catálogos creados por organizaciones de micología. No se verán afectados por el proyecto directamente, pero su input determinará el estado inicial de la IA y serán acreditados debidamente, pudiendo influir de algún modo en su credibilidad.

4) Competencia: Existen otras muchas aplicaciones en las diversas tiendas online que ofrecen una funcionalidad cuanto menos similar. Se verán afectadas en cuanto a que entrará un nuevo competidor en el mercado.

5) Director y codirectora del TFG: Silverio Martínez-Fernández y Cristina Gómez Seoane serán el director y codirectora respectivamente. Como el nombre indica, ambos estarán implicados en la dirección y el asesoramiento del proyecto.

6) Realizador del TFG: Yo, Álvaro Trius Béjar, también me considero stakeholder del proyecto porque seré el desarrollador principal. Seré el que le dedique las horas necesarias al proyecto con tal de llevarlo a su correcto término, puesto que de ello depende que finalice mi aprendizaje y pueda acceder al mundo laboral con un título.

1.5. ANÁLISIS DEL MERCADO

Ahora convendría justificar el desarrollo de nuestra aplicación dentro del contexto del mercado actual. Para ello, vamos a examinarlo y buscar las aplicaciones más similares y analizaremos sus funcionalidades.

1.5.1 Mushroom Identify

Mushroom Identify **[5]** es una de las aplicaciones más populares y más completas. La interfaz es muy sencilla e intuitiva y directamente ofrece las funcionalidades que se espera

de ella. Puedes pedir que identifique un hongo tanto a través de la cámara como a través de una imagen de la galería.

De cara a la identificación de hongos, la aplicación te permite corregirla en caso de error y te ofrece otros resultados similares. En caso de que se tengan dudas, también ofrece la posibilidad de crear un mensaje anónimo en un chat público para pedir ayuda al resto de la comunidad para identificarlo. Además de todo esto, la aplicación da toda la información de la seta escaneada, sirviendo además de manual portátil.

Para acabar considero que convendría señalar que la aplicación está llena de avisos en los que se advierte al usuario en contra de confiar del todo en la aplicación, debido al alto riesgo que lleva confundir un tipo de hongo con otro.

1.5.2 Shroomify

Shroomify **[6]** es la más diferente de entre las que señalo aquí, pero considero oportuno mencionarla teniendo en cuenta que es una de las más populares para cumplir la funcionalidad de "identificar setas".

Es una aplicación sencilla, no utiliza algoritmos de Machine Learning ni de reconocimiento a partir de imágenes para identificar setas. Más bien es un manual de setas portátil con la funcionalidad añadida de poder describir a la aplicación el tipo de seta que se ha encontrado a partir de introducir sus características físicas. Una vez ha sido descrito el hongo, la aplicación muestra todos los tipos de hongo que son similares a la descripción y permite consultar sus características biológicas en mayor detalle.

A parte de esto, la aplicación ofrece un acceso directo a la consulta de la enciclopedia, mucho más completa que la de la anterior aplicación y además permitiéndote consultar información sobre setas sin necesidad de encontrarlas.

1.5.3 Picture mushroom

Picture mushroom **[7]** recoge la mayoría de las funcionalidades de la primera: permite identificar setas usando la cámara o mediante una imagen subida de la galería del móvil y se te da toda su información. Además de esto, la aplicación proporciona al usuario con artículos de aprendizaje general sobre la recogida de setas.

La información que se te da con respecto al hongo identificado es mucho más extensa respecto a la primera aplicación que hemos comparado, pero no se la puede corregir en

caso de error y las otras funcionalidades están bloqueadas como características Premium para las cuales hay que pagar 21,99 €/año.

1.5.4 Fungus - identificación de hongos

Fungus **[8]** es la última aplicación que vamos a analizar. Como todas las anteriores puedes identificar una seta u hongo a partir de tanto una imagen de la galería como una foto sacada en el mismo momento. Se puede confirmar o rechazar la identificación y también se te proporciona toda la información acerca de la seta u hongo identificado. Aparte de esto, no tiene más funciones relevantes.

1.5.5 Conclusión

En resumen, podemos comprobar que esta idea no es novedosa y el mercado parece ya estar cubierto. Sin embargo, al analizar todas estas aplicaciones (véase Tabla 1, más abajo) nos damos cuenta que lo que cada una aporta en un área falta en las otras. Así, las que ofrecen un manual de setas accesible a la vez que completo son limitadas en el modo de usar la funcionalidad de "identificar setas a partir de una imagen" y viceversa. También, a pesar del alto riesgo que conlleva la identificación de setas, algunas de estas aplicaciones son bastante descuidadas a la hora de notificar al usuario con respecto a confiar en los resultados de la aplicación. Por último, las aplicaciones que te permiten corregir los resultados dan poco incentivo al usuario para hacerlo.

Por ello podemos concluir que hay espacio en el mercado para una aplicación sencilla, capaz de identificar setas mediante la cámara in situ o mediante una imagen, que además recoja adecuadamente la funcionalidad de servir como enciclopedia portátil de setas y que además incentive a los usuarios a aportar sus conocimientos para mejorar la aplicación.

Aplicación/ funcionalidad	Identificar setas a partir de una imagen	Capacidad de corregir y mejorar la app	Notifica al usuario respecto a los riesgos	Enciclopedia portátil	Incentiva al usuario a utilizarla
Mushroom Identify	Sí	Sí	Sí	Sí, pero solo respecto a la seta identificada	No
Shroomify	No	No	Sí	Sí, siempre	No
Picture mushroom	Sí	No	Sí	Sí, pero limitado por *premium*	No
Fungus	Sí	Sí	No	Sí, pero solo respecto a la seta identificada	No
Nuestra aplicación	Sí	Sí	Sí	Sí, siempre	Sí

Tabla 1: Comparativa de aplicaciones presentes en el mercado

2. ALCANCE

El proyecto completo sería la realización de una aplicación capaz de identificar de forma razonablemente correcta setas y hongos, capaz de ser mejorada y que a la vez proporcione al usuario una enciclopedia completa de información. La aplicación tomará la forma de aplicación de móvil.

Hay que tener en cuenta que, para realizarla, serán necesarios conocimientos de Machine Learning y reconocimiento de imágenes, conocimientos de los que carezco, por lo que el TFG abarcará el aprendizaje de los conceptos requeridos para realizar el proyecto.

2.1. OBJETIVOS Y SUB-OBJETIVOS

El objetivo final del proyecto es el de la correcta implementación de esta aplicación. Para poder cumplir con este objetivo será necesario cumplir los siguientes sub-objetivos:

1) Proporcionar una herramienta útil de identificación para aficionados a esta práctica: Esta herramienta ha de ser útil y precisa para los usuarios.

2) Proporcionar una herramienta educativa: Esta herramienta debe contener una enciclopedia de referencia de hongos y setas, útil tanto para aficionados como para veteranos.

3) Advertir de los peligros que conlleva la recolección de hongos y setas: Debe ser una herramienta que, a la vez que no debe de coartar la exploración y el descubrimiento, incentive la cautela.

4) Atraer a nuevos potenciales aficionados: Mediante técnicas de gamificación, deberá de hacer la práctica más entretenida y divertida.

5) Servir como caso de estudio sobre cómo medir requisitos no funcionales en aplicaciones basadas en machine learning: Tal y como habíamos especificado en el apartado de contextualización del problema, una aplicación basada en machine learning es muy atípica desde la perspectiva de identificación de requisitos no funcionales. Este proyecto servirá como un ejemplo para futuros proyectos.

2.2. RIESGOS

A pesar de nuestros esfuerzos por asegurarnos de que todo salga bien, es inevitable que ocurran sucesos incontrolables que nos obliguen a cambiar el plan inicial. Es por ello que conviene ser consciente de los posibles riesgos. En este apartado incluyo una pequeña recopilación de los riesgos posibles con una pequeña descripción de todos ellos.

1) Bugs: Los bugs son todo el conjunto de errores informáticos con los que todo programador se encuentra a la hora de llevar a cabo su profesión. Básicamente, es cuando la máquina "no funciona del modo esperado". Suele ser por algún despiste del propio programador que al escribir el código comete un desliz, no instala software de la forma correcta, etc. Es un riesgo casi inevitable.

2) Inexperiencia con las tecnologías utilizadas: Teniendo en cuenta que partiremos sin conocimiento alguno sobre machine learning para realizar este proyecto, hay ciertas posibilidades de que esto suponga un obstáculo para la consecución del mismo.

3) Falta de acceso a las herramientas necesarias: Durante el desarrollo de este proyecto necesitaré acceso a ciertas bases de datos y galerías de imágenes que podría no estar al alcance de mi mano. No disponer de una base de datos sería fatal para el proyecto.

4) Calendario cerrado: El desarrollo del TFG tiene unas claras fechas de inicio y de final, por lo que es importante no excederse de éstas.

3. METODOLOGÍA Y HERRAMIENTAS

3.1. METODOLOGÍA

La metodología que usaremos será *Agile Scrum*, pero teniendo en cuenta que el equipo estará formado por solo una persona, yo mismo haré tanto de *scrum* master como de desarrollador. Mi director y co-directora, por otro lado, actuarán como *product owners*.

El trabajo será dividido en *sprints* y cada sprint estará delimitado por tres revisiones. La primera revisión marcará el inicio del sprint: en ésta definiré junto a los *product owners* las historias de usuario a ser desarrolladas (*sprint planning*). La segunda y la tercera marcarán el final: en éstas revisaré de la iteración la implementación de las funcionalidades (*sprint review*) y haré una breve revisión (*sprint retrospective*) de nuevo, con los *product owners*. La idea es que al final de cada iteración obtenga un incremento en el producto.

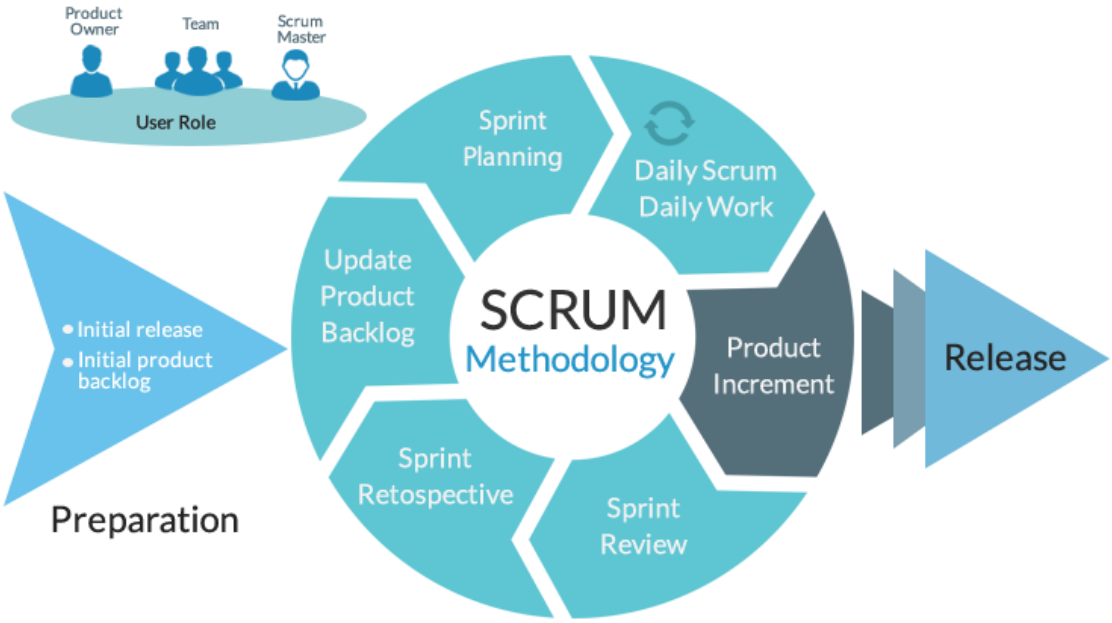

Ilustración 1: Diagrama de metodología scrum ***[9]***

Para la implementación de código utilizaré el método gitflow, este consiste en dividir las diversas tareas en diversas ramas modulares e ir incorporándolas a la implementación del código a medida que vaya completando las historias de usuario.

3.2. HERRAMIENTAS PARA VALORAR EL CORRECTO SEGUIMIENTO DE LA METODOLOGÍA.

3.2.1 Trello

[10] Para la creación y el seguimiento de las historias de usuario. Se creará un tablero dividido en 4 columnas (*To-Do*, *Doing*, *Testing*, *Done*) y las historias de usuario se irán desplazando a medida que vayan cambiando.

3.2.2 Github

Para el mantenimiento y la gestión del código utilizaremos Github **[11]**, una herramienta en la nube que permite dividir el código a través de ramas modulares y permite hacer el control de versiones a mediante repositorios.

Se usarán sobre todo dos tipos de ramas: la rama *master* y la rama de *features*. En la rama master tendremos la versión más reciente del proyecto. Cada vez que queramos añadir una nueva *feautre* al proyecto, crearemos una rama aparte a partir de la master de tal manera que una vez acabemos de implementar esa *feature*, la podremos testear inmediatamente con todo lo desarrollado hasta el momento. Una vez esté todo listo la *feature* será incorporada la rama *master*.

Con tal de poder gestionar las ramas con facilidad, utilizaremos Github Desktop, una extensión de Github que ofrece una interfaz gráfica que permite una gestión de ramas más limpia y eficiente.

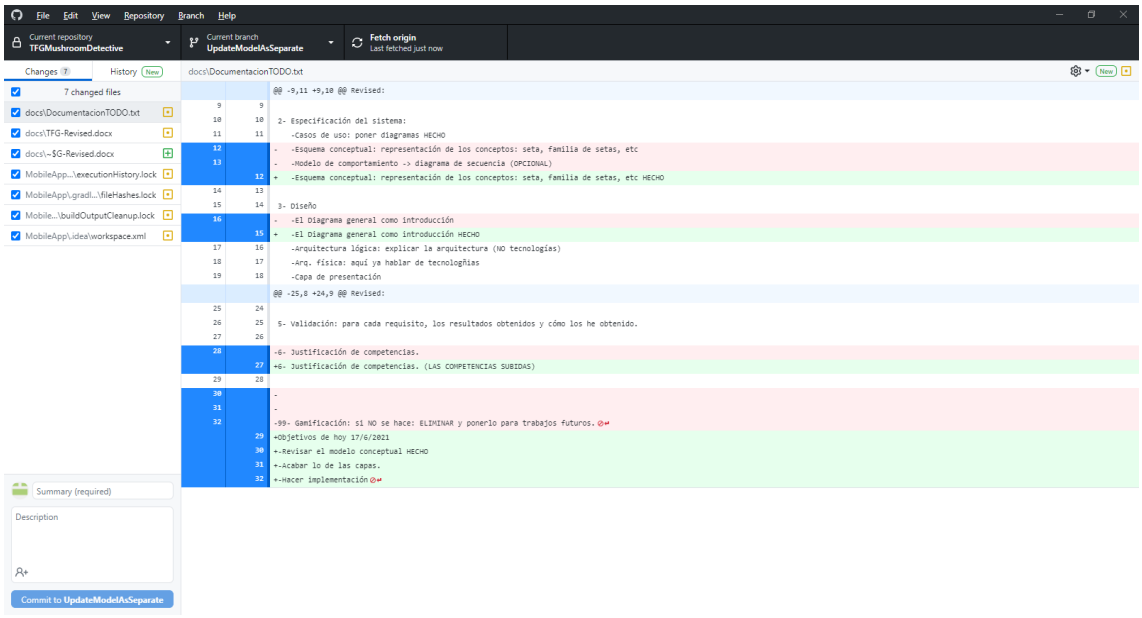

Ilustración 2: Entorno Github Desktop, que permite la gestión de ramas de forma limpia y eficiente

3.2.3 Dropbox

[12] Para guardar la documentación relacionada, utilizaremos la herramienta Dropbox. Así podremos guardar la documentación en la nube y no preocuparnos en caso de perder algo localmente.

4. PLANIFICACIÓN INICIAL

En este apartado describiremos las tareas a realizar en el tiempo del que disponemos. Las fechas límite del proyecto son:

1) Inicio: 15 de febrero del 2021.

2) Final: 27 de junio del 2021, teniendo en cuenta que las presentaciones serán alrededor del 28 y la documentación se deberá de presentar la semana anterior.

Son 18 créditos, estimados como 30 horas por crédito suman un total de 540 horas totales a dedicar.

Es importante tener en cuenta que esta es una estimación inicial y, teniendo en cuenta, de nuevo, el desconocimiento en este ámbito, es muy posible que haya muchos cambios de cara a la planificación final ejecutada.

4.1. GP - GESTIÓN DEL PROYECTO

GP1- Contextualización y alcance: Consiste en la elaboración de un documento en el que se sitúa el contexto del proyecto y los objetivos a alcanzar. (Dependencias: Ninguna, horas estimadas: 25).

GP2 - Planificación: Se elaborará un documento en el que se enumerarán las tareas a ejecutar durante el proyecto y su distribución a lo largo del calendario, con sus cargas de tiempo y trabajo respectivas. (Dependencias: GP1. Horas estimadas: 20).

GP3 - Gestión económica y de sostenibilidad (Dependencias: GP2): Se elaborará un documento en el que se tratará de cuestiones de gastos y se analizará el impacto medioambiental. (Dependencias: GP2. Horas estimadas: 25).

GP4 - Integración del documento final (Dependencias: GP1, GP2 y GP3): Todos los documentos anteriores se someterán a revisión y serán integrados en un documento que formará parte de la documentación final del TFG. (Dependencias: GP3. Horas estimadas: 20).

4.2. FASE DE PREPARACIÓN

4.2.1 AC - Aprendizaje de los conocimientos necesarios

Debido a mi poca práctica con el campo del Machine Learning y el reconocimiento de imagen, será necesario invertir unas cuantas horas en realizar una adecuada investigación que consistirá en consultas con profesores y lectura de artículos. Además convendrá programar una prueba de concepto.

AC1 - Aprendizaje de conocimientos básicos de Machine Learning: (Dependencias: ninguna) Aquí nos centraremos en aprender los conceptos básicos de Machine Learning. No sólo se hará un estudio teórico, sino que además se buscará referencias para aprender a implementar una IA de estas características. (Dependencias: GP3. Horas estimadas: 30).

AC2 - Prueba de concepto: (Dependencias: AC1) Aquí desarrollaremos una prueba de concepto: una pequeña demo para comprobar que realmente hemos sido capaces de adquirir los conocimientos necesarios para realizar este proyecto. (Dependencias: AC1. Horas estimadas: 30).

4.2.2 I - Inception

I1 - Especificación de los requisitos: (Dependencias: AC) Se describirán todos los requisitos del sistema, tanto funcionales como no funcionales. De este modo podremos especificar con claridad de qué se trata nuestro proyecto. (Dependencias: AC2. Horas estimadas: 22).

I2 - Preparación del entorno: (Dependencias: I1) Será el proceso a partir del cual nos encargaremos de descargar todos los programas y las librerías necesarias para desarrollar el programa. Debido a la naturaleza de auto-aprendizaje de este proyecto, esta tarea no estará limitada sólo al comienzo sino que es posible que el desarrollo del mismo nos lleve a descubrir nuevas necesidades que nos lleven a tener que requerir de diferentes programas. A pesar de todo, intentaremos que esto no suceda. (Dependencias: I1. Horas estimadas: 5).

I3 - Diseño de la arquitectura: (Dependencias: I1) Para hacernos una idea acerca de cómo deberemos diseñar el programa, se usarán las pautas de las metodologías *agile* para hacer una estructura de la base de datos. (Dependencias: I2. Horas estimadas: 5).

4.3. DESARROLLO DEL PROYECTO

4.3.1 IA - Implementación de la Inteligencia artificial

IA1 - Implementación de una base de datos: (Dependencias: I) Implementación de una base de datos de hongos y setas que tanto la IA como la enciclopedia usarán para consultar. Deberá de incluir no sólo el nombre de las setas sino también todas sus características, entre ellas, la colección de imágenes que le corresponde. (Dependencias: I3. Horas estimadas: 48).

IA2 - Implementar la IA: (Dependencias: IA1) Implementación y testeo de una Inteligencia Artificial que sea capaz de reconocer una imagen y clasificarla en una clase específica. (Dependencias: IA1. Horas estimadas: 48).

4.3.2 Integración de la aplicación

Este apartado incluye todas las tareas relacionadas con la implementación de la aplicación en sí. Algunas de estas tareas se pueden hacer aparte de implementar la IA, por eso he considerado interesante separarlo en un apartado distinto.

4.3.2.1 API - Implementación de la interfaz

API1 Diseño de la interfaz: (Dependencias: I2) Antes de implementar la interfaz convendrá dejar clara la idea del diseño que nos interesa para asegurarnos que acertamos con el *look and feel* de la aplicación. (Dependencias: I3. Horas estimadas: 16).

API2 Creación de la interfaz: (Dependencias: API1) Una vez tengamos claro el *look and feel*, habrá que implementar la interfaz. Esta es una tarea que desarrollaremos a medida que vayamos implementando cada funcionalidad. (Dependencias: API1. Horas estimadas: 16).

4.3.2.2 PER - Gestión de perfil

PER1 - Crear perfil: implementación de la funcionalidad perfil que servirá para guardar los logros resultados de la gamificación. Al principio sólo deberá permitir guardar los datos personales. Éstos serán guardados localmente. (Dependencias: I2. Horas estimadas: 10).

PER2 - Modificar perfil: Se programará la posibilidad de modificar los datos personales. (Dependencias: PER1. Horas estimadas: 10).

PER3 - Implementación de logros: Habrá que programar la funcionalidad de que cuando el usuario haya alcanzado una serie de metas y objetivos, reciba una medalla virtual que será visible en su perfil**.** No estará del todo lista hasta que todas las funcionalidades se hayan implementado (Dependencias: PER2, GE1, IS2. Horas estimadas: 12).

4.3.2.3 GE - Gestión de enciclopedia

GE1 - Consultar entrada de enciclopedia: implementación de la funcionalidad de enciclopedia, que consistirá en permitir al usuario que consulte una entrada en la enciclopedia. (Dependencias: IA1. Horas estimadas: 32).

4.3.2.4 IS - Identificación de setas

IS1 - Inserción y reconocimiento de imágenes (dependencia: IA2) La aplicación deberá de poder aceptar imágenes, conectarse a la base de datos y retornar el resultado. (Dependencias: IA2. Horas estimadas: 48).

IS2 - Corrección de imagen reconocida errónea: (dependencia: IS1) Una vez se haya identificado una seta, el usuario deberá de poder corregir a la aplicación y decirle si se ha equivocado, en ese caso, la imagen deberá de enviarse a la base de datos para actualizarse. (Dependencias: IS1. Horas estimadas: 48).

4.4. DC - DOCUMENTACIÓN Y COMUNICACIÓN

DC1 - Seguimiento: A medida que avance el proyecto se irán apuntando todos los hitos conseguidos para poder incluirlos más tarde en la documentación. (Dependencias: Ninguna. Horas estimadas: 10).

DC2 - Documentación: Durante el desarrollo del proyecto se harán tareas de revisión y actualización de la documentación. (Dependencias: GP4. Horas estimadas: 50).

DC3 - Comunicación: Se realizarán reuniones con los tutores del proyecto para analizar el trabajo realizado y resolver dudas y problemas surgidos. (Dependencias: Ninguna. Horas estimadas: 10).

4.5. ESTIMACIONES Y GANTT

En esta sección mostraremos la tabla de estimaciones y el diagrama de Gantt. En la primera se muestran las estimaciones en lo referente a las horas requeridas para desarrollar cada tarea y los recursos necesarios. Además también se incluyen las dependencias con respecto a otras tareas. Puesto que son estimaciones, puede no corresponderse con la realidad. La segunda tabla nos muestra la distribución temporal del desarrollo de cada tarea en el calendario.

4.5.1 Tabla de estimaciones

	Horas estimadas	Total	Dependencias	Recursos
1- GP - Gestión del proyecto		90		
GP1- Contextualización y alcance	25			PC, Drive, Word
GP2 - Planificación	20		GP1	PC, Drive, Word, Excel
GP3 - Gestión económica y de sostenibilidad	25		GP2	PC, Drive, Word, Excel
GP4 - Integración del documento final	20		GP3	PC, Drive, Word, Excel
2- AC - Aprendizaje de los conocimientos necesarios		60		
AC1 - Aprendizaje de conocimientos básicos de Machine Learning	30		GP3	PC, Drive, Artículos
AC2 - Prueba de concepto	30		AC1	PC, Drive, Artículos
3- I - Inception		32		
I1 - Especificación de los requisitos:	22		AC2	PC, Word
I2 - Preparación del entorno:	5		I1	PC, Android Studio, Git
I3 - Diseño de la arquitectura:	5		I2	PC, Draw.io
4- IA - Implementación de la Inteligencia artificial		96		
IA1 - Implementación de una base de datos:	48		I3	PC y lo que defina AC2
IA2 - Implementar la IA:	48		IA1	PC y lo que defina AC2
5- API - Implementación de la interfaz		32		
API1 Diseño de la interfaz	16		I3	PC, Adobe XD
API2 Creación de la interfaz:	16		API1	PC, Android Studio, Git
6- PER - Gestión de perfil		32		
PER1 - Crear perfil:	10		I2	PC, Android Studio, Git
PER2 - Modificar perfil:	10		PER1	PC, Android Studio, Git
PER3 - Implementación de logros:	12		PER2, GE1, IS2	PC, Android Studio, Git
7- GE - Gestión de enciclopedia		32		
GE1 - Consultar entrada de enciclopedia:	32		IA1	PC, Android Studio, Git
8- IS - Identificación de setas		96		
IS1 - Inserción y reconocimiento de imágenes	48		IA2	PC, Android Studio, Git
IS2 - Corrección de imagen reconocida errónea:	48		IS1	PC, Android Studio, Git
9- DC - Documentación y comunicación		70		
DC1 - Seguimiento	10			PC, Taiga, Word
DC2 - Documentación:	50		GP4	PC, Taiga, Word
DC3 - Comunicación:	10			PC, Meet, Gmail
	540	540		

Tabla 2: Tabla de tareas con sus dependencias, coste estimado y recursos necesarios

4.5.2 Diagrama de Gantt

	FEBRERO		MARZO			ABRIL						MAYO				JUNIO		
						Sprint1			Sprint2			Sprint3			Sprint 4			
Semanas:	**15/2 - 21/2**	**22/2 - 28/2**	**1/3 - 7/3**	**8/3 - 14/3**	**15/3 - 21/3**	**22/3 - 28/3**	**29/3 - 4/4**	**5/4 - 11/4**	**12/4 - 18/4**	**19/4 - 25/4**	**26/4 - 2/5**	**3/5 - 9/5**	**10/5 - 16/5**	**17/5 - 23/5**	**24/5 - 30/5**	**31/5 - 6/6**	**7/6 - 13/6**	**14/6 - 20/6**	**21/6 - 27/6**
1- GP - Gestión del proyecto																			
GP1- Contextualización y alcance																			
GP2 - Planificación																			
GP3 - Gestión económica y de sostenibilidad																			
GP4 - Integración del documento final																			
2- AC - Aprendizaje de los conocimientos necesarios																			
AC1 - Aprendizaje de conocimientos básicos de Machine Learning																			
AC2 - Prueba de concepto																			
3- I - Inception																			
I1 - Especificación de los requisitos:																			
I2 - Preparación del entorno:																			
I3 - Diseño de la arquitectura:																			
4- IA - Implementación de la Inteligencia artificial																			
IA1 - Implementación de una base de datos:																			
IA2 - Implementar la IA:																			
5- API - Implementación de la interfaz																			
API1 Diseño de la interfaz																			
API2 Creación de la interfaz:																			
6- PER - Gestión de perfil																			
PER1 - Crear perfil:																			
PER2 - Modificar perfil:																			
PER3 - Implementación de logros:																			
7- GE - Gestión de enciclopedia																			
GE1 - Consultar entrada de enciclopedia:																			
8- IS - Identificación de setas																			
IS1 - Inserción y reconocimiento de imágenes																			
IS2 - Corrección de imagen reconocida errónea:																			
9- DC - Documentación y comunicación																			
DC1 - Seguimiento																			
DC2 - Documentación:																			
DC3 - Comunicación:																			

Tabla 3: Diagrama de Gantt

4.6. RIESGOS Y GESTIÓN DE LOS MISMOS

Como habíamos mencionado previamente, es muy probable que se den eventos imprevisibles que impidan que el proyecto se desarrolle acorde a los tiempos previstos. Antes ya habíamos citado los riesgos, ahora presentaremos un análisis más exhaustivo: intentando predecir el coste que conllevarían y explicaremos el plan que seguiremos para atenuarlos.

Riesgo	**Probabilidad**	**Coste**
Bugs	Alta (75%)	30 horas
Inexperiencia con las tecnologías utilizadas	Media (50%)	30 horas
Falta de acceso a las herramientas necesarias	Media (50%)	10 horas
Calendario cerrado	Baja (25%)	10 horas

Tabla 4: Lista de riesgos con su probabilidad y coste estimado

4.6.1 Bugs

Los bugs son algo casi inevitable en el desarrollo del software, más aún en el proyecto actual teniendo en cuenta que trabajaremos con conceptos nuevos. Aunque sean inevitables sus consecuencias pueden ser atenuadas.

Nosotros atenuaremos este riesgo mediante el continuo testeo de las funcionalidades que implementemos, asegurándonos de que superan todos los tests antes de integrar sus ramas en la rama principal y más adelante asegurándonos que siguen superando los tests una vez unidos con el resto de las funcionalidades. La clave será detectar todos estos errores durante la fase de desarrollo.

4.6.2 Inexperiencia en las tecnologías utilizadas

Como dijimos, la posibilidad de no poder proceder debido a falta de experiencia es existente. Con tal de minimizar este riesgo hemos introducido dentro del proceso de desarrollo de la aplicación una fase de aprendizaje antes de la fase de inception en la que nos aseguraremos de documentarnos bien y hacer una prueba de concepto. Además de eso, nos aseguraremos de usar el contacto con el tutor para hacer reuniones y resolver todas las dudas y problemas que podrían surgir.

4.6.3 Falta de disposición de acceso a las herramientas necesarias

En este proyecto nos comprometemos a utilizar muchas bases de datos con información de hongos y setas y pudiera darse el caso de algunos de éstos no estén disponibles para uso público o que sea necesario pedir permiso para utilizarlo. Por un lado, con tal de mitigar este riesgo ya nos hemos asegurado de encontrar algunas bases de datos a las que nos podemos referir como mínimo para realizar este trabajo. Por el otro, también nos aseguraremos de buscar más alternativas.

4.6.4 Calendario cerrado

Este riesgo consiste en que es posible que no llegue a tiempo para entregar el trabajo. Es un riesgo bajo porque, aunque es existente, toda la preparación que se está llevando a cabo tiene como objetivo evitar que esto ocurra.

5. PRESUPUESTO

Una vez hemos realizado la planificación, convendrá hacer el cálculo de presupuesto de este proyecto. Para ello, en primer lugar convendrá conocer quiénes son los miembros del equipo y el sueldo que les corresponde. En realidad este proyecto va a estar realizado una sola persona, pero consideramos interesante explorar la posibilidad de que fuera desarrollado por un equipo completo. Es por eso que primero presentaremos los costes estimados reales y los costes en caso de realizar el proyecto en un equipo.

5.1. ESTIMACIÓN DE COSTES: CASO REAL

Los costes de desarrollo de esta aplicación serán muy pequeños, el proyecto está realizado por una sola persona con un portátil. No se espera ningún gasto mayor, ya que todas las herramientas utilizadas serán gratuitas.

Modelo	Precio (€)	Dedicación diaria (horas)	Días laborales (/año)	Duración proyecto (horas)	Vida útil (años)	Amortización (€)
ASUSPRO Essential	1050	5	220	540	4	128,86

Consumo por hora (Kwh)	Precio (€/Kwh)	Precio total de consumo (€)
1,1	0,14	83,16

Tabla 5: Costes estimados reales

5.2. ESTIMACIÓN DE COSTES: CASO FALSO

En este apartado exploraremos los costes estimados en caso de que el proyecto fuera realizado por un equipo.

En la tabla siguiente se muestra quiénes son los miembros del equipo y sus sueldos correspondientes basados en Indeed **[13]**, una plataforma de búsqueda de trabajo.

ROL/Persona	Sueldo (€/hora)	Sueldo + SS (€/hora)
Jefe de proyecto	19,91	25,88
Analista	13,50	17,55
Arquitecto de Software	19,60	25,48
Diseñador gráfico y UX	12,56	16,33
Desarrollador Software	14,61	18,99
Tester	10,27	13,35

Tabla 6: Miembros del equipo con sus correspondientes sueldos.

5.2.1 Costes de actividad

Ahora calcularemos cuanto tiempo deberá dedicarle cada uno de los miembros del equipo a cada una de las tareas. Todas ellas seguirán la misma lógica excepto la tarea de aprendizaje de conocimientos: aunque haya establecido que durará 60 horas, son calculadas en base a que es investigación realizada individualmente por el autor. El aprendizaje de conocimientos lo añadiremos más tarde sobre el coste de tareas como el precio de contratar un profesor durante 1 semana, de ahí que no salga en esta tabla.

Se podría asumir que los trabajadores ya tienen los conocimientos necesarios para realizar la aplicación pero, teniendo en cuenta que esta actividad va a ser tan importante para el proyecto, hemos considerado que había que incluirla en los costes, y este es el modo que se nos ha ocurrido.

	Horas estimadas	Jefe de Proyecto	Analista	Arquitecto Software	Diseñador Grafico y UX	Desarrollador	Tester
1- GP							
GP1	25	25					
GP2	20	20					
GP3	25	25					
GP4	20	20					
2- AC							
AC1	30						
AC2	30						
3- I							
I1	22		15	7			
I2	5					2,5	2,5
I3	5			5			
4- IA							
IA1	48		5	5		19	19
IA2	48		5	5		19	19
5- API							
API1	16				16		
API2	16				16		
6- PER							
PER1	10		1,5	1,5	3'5	3'5	
PER2	10		1,5	1,5	3'5	3'5	
PER3	12		1,5	1,5	3'5	3'5	
7- GE							
GE1	32		4	4		13	13
8- IS							
IS1	48		5	5		19	19
IS2	48		5	5		19	19
9- DC							
DC1	10	10	0				
DC2	50	20	10	10		10	
DC3	10	2	2	2	2	2	2
Total horas	540	122	55,5	52,5	34	103,5	93,5
Sueldo + SS		25,88	17,55	25,48	16,33	18,99	13,35
Coste		3157,36	974,025	1337,7	555,22	1965,465	1248,225
Coste Total	9237,995						

Tabla 7: Lista de tareas con las horas asignadas a cada miembro con sus respectivos costes.

El coste total es de 9237,99. Sobre este le añadimos el coste de formación que se calcula de la siguiente forma:

Horas dedicadas al curso (1 semana, 5 horas diarias)	25
Sueldo del formador (€) (calculado según Indeed **[14]**)	12,41
Miembros que realizarán el curso	Analista, Arquitecto de Software, *Developer* y Tester
Total coses de formación (€)	2195,50
Coste total de actividad (€)	11433,49

Tabla 8: Costes de curso de aprendizaje de machine Learning añadidos a los costes totales de actividad

5.2.2 Costes genéricos

Los costes genéricos son todos aquellos que no dependen de la actividad. Asumiendo que seguimos con la idea de que estamos con un equipo de 6 personas, los gastos se resumen en 6 ordenadores (uno para cada uno) y el Works pace con sus correspondientes costes de alquiler y consumo eléctrico.

De cara al equipo informático, vamos a calcular su amortización a partir de la fórmula que aparece a continuación. Luego se multiplicarán esos costes por el número de trabajadores. Con tal de facilitar el cálculo, asumimos que todos los ordenadores son el mismo que el que el autor auténtico usará para desarrollar el proyecto.

$$\frac{Coste\ (euros)}{Vida\ útil\ (años) * Días\ laborables\ al\ año * Dedicación\ diaria\ (horas)} * Duración\ del\ proyecto\ (horas)$$

Modelo	Precio (€)	Dedicación diaria (horas)	Días laborales (/año)	Duración proyecto (horas)	Vida útil (años)	Amortización (€)
ASUSPRO Essential	1050	5	220	540	4	128,86

Tabla 9: Costes de amortización de equipo informático

De cara al Works pace, vamos a asumir que alquilamos un espacio de coworking en Zamness **[15]**. Según la página web, podemos estimar:

Precio por mes (€)	125
Meses de trabajo	5
Total (€)	625

Tabla 10: Costes de Works pace

Por tanto, los costes genéricos totales son:

Nº de ordenadores	Nº de personas Works pace	Total CG (€)
6	6	4523,16

Tabla 11: Costes generales totales

5.2.3 Contingencia

A la hora de desarrollar proyectos, pueden suceder imprevistos que encarezcan la realización del proyecto. Como es imposible predecirlos todos, normalmente se añade un porcentaje presupuesto a cada uno de los costes con tal de suavizar el golpe. Este porcentaje se suele situar entre el 10% y el 20%.

Respecto al coste de actividad le asignaremos un margen de contingencia bajo del 10% ya que, aunque se puede dar el caso que algunas de las actividades tarden más de lo esperado y ello implique que los trabajadores deban dedicarle más horas, también es verdad que se trata de un equipo que puede respaldarse mutuamente.

Respecto a los costes genéricos, teniendo en cuenta la situación de pandemia actual, es probable que acabemos despilfarrando el dinero en la contratación de un Works pace. Por el otro lado, en un principio no esperamos un fallo en el material que requiera de costes de reparación, por lo que le asignaremos un margen de contingencia intermedio, de un 15%.

Por lo que los costes de contingencia quedarían así:

	Contingencia a aplicar	Contingencia aplicada (€)
Costes de Actividad	+10%	12576,84
Costes Genéricos	+15%	5201,63

Tabla 12: Contingencia aplicada a costes de actividad y genéricos

5.2.4 Coste de riesgos

Ahora, basándonos en los riesgos que planificamos varios apartados atrás, vamos a calcular sus costes. Para hacerlo tomaremos cada uno de ellos, estimaremos el tiempo que deberá dedicar cada una de las personas implicadas para solucionarlo, calcularemos el precio de las horas y le aplicaremos un porcentaje en función de la probabilidad.

Riesgo	**Probabilidad**	**Tiempo (horas)**	**Coste/Hora (€/h)**	**Coste (€)**
R1	Alta (75%)	30h -15 h Desarrollador -15 h Tester	D: 18,99 T: 13,35	363,82
R2	Media (50%)	30h -10h Desarrollador -10h Tester -5h Analista -5h Arquitecto	D: 18,99 T: 13,35 An: 17,55 Ar: 25,48	269,27
R3	Media (50%)	10h -5h Desarrollador -5h Tester	D:18,99 T: 13,35	81,35
R4	Baja (25%)	10h -5h Desarrollador -5h Tester	D:18,99 T: 13,35	81,35
Total				795,80

Tabla 13: Coste de riesgos

5.2.5 Coste final

El coste final será determinado por la suma de los costes anteriores, que es:

Costes de actividad (€)	Costes Genéricos (€)	Coste de riesgos(€)	Total (€)
12576,84	5201,63	795,8	18574,27

Tabla 14: Costes totales

5.3. CONTROL DE GESTIÓN

A pesar de nuestros esfuerzos por hacer la adecuada planificación de los costes, no sirve de nada si no se hace un adecuado seguimiento para ver si se está cumpliendo o no, es por eso que conviene establecer unos mecanismos de control para hacer seguimiento de ellos.

Para ello, usaremos la siguiente fórmula:

$$Desviación = Coste\ estimado - coste\ real$$

Siendo:

-Coste estimado: el coste predicho en el apartado anterior.

-Coste real: el coste real de la tarea o actividad.

-Desviación: la diferencia entre el coste estimado y el coste real.

Conviene indicar aquí dos cosas. En primer lugar, que esta fórmula la aplicaremos tanto para calcular la desviación en horas como la desviación en líquido, así podremos usarla análogamente para ver si una tarea requiere más tiempo de lo esperado o si una actividad es más barata de lo que parecía. En segundo lugar, el signo de la desviación será un buen indicador para comprobar qué decisión tomar: si es positivo, significa que he sobreestimado la actividad y esos recursos los debería enfocar en algo distinto, si es negativo, significa que debo de usar los recursos de contingencia para suplir este defecto.

6. ANÁLISIS DE REQUISITOS

6.1. REQUISITOS FUNCIONALES

Estos requisitos son los que definen el comportamiento de nuestra aplicación:

-Identificación de setas: La aplicación podrá tomar una imagen e identificar, de entre las clases que conoce, de qué clase de seta se trata. Si no conoce la clase de seta, la aplicación dará la estimación más cercana.

-Corrección de identificación de setas: La aplicación deberá de poder ser corregida en caso de equivocación al identificar una seta.

-Mejora en la identificación de setas: El sistema deberá de poder tomar las correcciones anteriores, incluirlas en su dataset y así mejorar sus predicciones. Para ello, se irán recogiendo las solicitudes de corrección de los usuarios cada semana y se hará una actualización del modelo en el servidor. Los usuarios más adelante podrán descargarse el nuevo modelo.

-Consulta de enciclopedia de setas: El sistema deberá de poder ofrecer al usuario la posibilidad de consultar todo el catálogo de setas a su disposición.

-Consulta de datos de setas: Una vez que la seta haya sido identificada, el sistema inmediatamente permitirá al usuario conectarse a la enciclopedia y consultar los datos de esa seta.

-Gestión de usuario: El sistema deberá de poder permitir al usuario crear su propio perfil en el que guardar sus datos personales y su progreso. También deberá poder permitir consultarlo, modificarlo y borrarlo.

-Gestión de logros: El sistema deberá de ser capaz de seguir la actividad del usuario y otorgarle recompensas virtuales.

6.2. REQUISITOS NO FUNCIONALES

Una de las cosas que hemos ido indicando ya es lo distinto que va a resultar nuestra aplicación. En cuanto a que es una aplicación software, podemos plantearnos algunos requisitos no funcionales que son típicos, pero en cuanto a que utiliza machine learning habrá otros cuantos que serán únicos. En este apartado nos disponemos a describir los dos tipos de requisitos no funcionales.

6.2.1 Requisitos no funcionales típicos de una aplicación software

Nombre	**Eficiencia**
Descripción	Tanto que la aplicación sea capaz de identificar la seta a partir de la imagen como que el algoritmo de reaprendizaje reentrene el modelo serán ejecutados en un tiempo razonable.
Criterio de satisfacción	Para la identificación: 3 segundos como máximo. Para el reentreno: 20 minutos como máximo.
Modo de validación	Para identificación: al implementar, colocaremos cronómetros en la aplicación y nos aseguraremos de que, después de 50 identificaciones, la media sea menor de 3 segundos (usaremos el método Instant.now() al inicio y al final de la identificación). Para el reentreno: al implementar, lanzaremos 10 reentrenos del modelo y nos aseguraremos que la media sea menor a 20 minutos (usaremos el comando %%time para calcular el tiempo que tarda).

Nombre	**Usabilidad**
Descripción	La interface será asequible y fácil de usar. Usará una combinación de colores y elementos gráficos agradable y apetecible.
Criterio de satisfacción	Los usuarios deberán de encontrar la interfaz fácil y asequible.
Modo de validación	Distribuiremos la aplicación finalizada a 5 usuarios y, después de utilizarla, le haremos un cuestionario. Entre las preguntas que le haremos le pediremos que evalúe de 1 a 5 lo fácil que les ha sido aprender a moverse por los menús. Deberemos conseguir un 3,5 de media como mínimo.

Nombre	**Disponibilidad**
Descripción	La aplicación deberá de estar disponible aún y cuando no se disponga de conexión a internet. En caso de no tener conexión lo único que no funcionará será las funcionalidades de red.
Criterio de satisfacción	Salvo la corrección de setas y la actualización del modelo local, todo lo demás tiene que funcionar el 100% de las veces.
Modo de validación	La aplicación funcionará sin conexión a internet con las funcionalidades especificadas.

Nombre	**Salud y seguridad**
Descripción	Teniendo en cuenta que trataremos con asuntos tan arriesgados como lo es la posibilidad de que una seta sea o no comestible, nos aseguraremos de dejar bien claro que esta aplicación es, por ahora, un prototipo, y que por lo tanto no conviene fiarse de los resultados que dé. Para ello pondremos unos avisos en la aplicación para recordar al usuario de que es preferible que consulte a un especialista. En particular, como mínimo pondremos uno en el menú principal.
Criterio de satisfacción	Tiene que existir un aviso relativo al uso seguro de la aplicación accesible desde el menú principal en el que se debe explicar que: 1- La aplicación es un prototipo, por lo que no hay que fiarse de los resultados. 2- No consumir ninguna de las setas. 3- Tener como prioridad el criterio de los expertos.
Modo de validación	Comprobación que existe el aviso anterior.

Nombre	**Apariencia**
Descripción	La aplicación deberá de ser amigable a los ojos, con un diseño atractivo para los usuarios que la utilicen.
Criterio de satisfacción	Los usuarios tendrán que ver la aplicación como amigable y agradable a los ojos. En una encuesta a potenciales usuarios la nota otorgada a la apariencia debe ser como mínimo 3,5.
Modo de validación	Con un cuestionario, después de distribuir la aplicación a 5 usuarios, les pediremos que evalúen la apariencia de la aplicación de 1 a 5. La nota deberá de ser mínimo un 3,5.

Nombre	**Aprendizaje**
Descripción	El diseño de la interfaz debe estar diseñada de tal manera que el usuario lo encuentre fácil desplazarse por los menús.
Criterio de satisfacción	El usuario deberá de aprender a usar la aplicación fácilmente. En una encuesta a potenciales usuarios la nota otorgada a la facilidad de aprendizaje debe ser como mínimo 3,5.
Modo de validación	Con un cuestionario, después de distribuir la aplicación a 5 usuarios, les pediremos que evalúen lo fácil que les ha resultado aprender a usar la aplicación de 1 a 5. La nota deberá de ser mínimo un 3,5.

Nombre	**Fiabilidad**
Descripción	La aplicación deberá de ser razonablemente fiable a identificar setas,
Criterio de satisfacción	De todas las clases clasificables, la aplicación deberá de acertar como mínimo el 85%.
Modo de validación	El modelo deberá de tener un *accuracy* de 85%.

6.2.2 Requisitos no funcionales específicos de machine learning

Como hemos ido mencionando a lo largo de este estudio, machine learning exige un cambio en el modo con el que enfocamos el diseño de software y, por extensión, la identificación de requisitos no funcionales. El artículo *Towards Guidelines for Assessing Qualities of Machine Learning Systems* **[2]** propone en la tabla 1 un conjunto de atributos

de calidad de los cuales tomaremos aquellos que consideramos aplicables a nuestro problema. También La página *Engineering best practices for Machine Learning* **[16]** plantea un conjunto de buenas prácticas a la hora de desarrollar un modelo; de las cuales también vamos a tomar algunas.

Nombre	**Modelo apropiado**
Descripción	El modelo que utilizaremos deberá de ser el apropiado para el problema que queremos resolver. Así, las tareas deberán de poder trabajar con los tipos de datos de los dataset.
Criterio de satisfacción	El modelo deberá de usar imágenes de tamaño 224x224x3 (altura, anchura y RGB) y deberá de ser optimizado para móviles.
Modo de validación	El modelo deberá de usar las imágenes con las características que hemos especificado y deberá de tener una arquitectura optimizada para móviles.

Nombre	**Capacidad de la consecución de la tarea (o *goodness of fit*)**
Descripción	El modelo deberá de desarrollar la tarea que queremos realizar bajo unos criterios mínimos de aceptación: deberá de ser mínimamente preciso.
Criterio de satisfacción	El modelo debe alcanzar un 85% de precisión (*accuracy*) y ha de estar por debajo de 0.6 en pérdidas (*loss*).
Modo de validación	Entrenaremos el modelo 10 veces y nos aseguraremos de que estas mediciones siempre sean superadas satisfactoriamente.

Nombre	**Datos de entrada completos y equilibrados**
Descripción	A la hora de introducir los datos de entrada para el primer entrenamiento, nos interesa que estos estén bien equilibrados: demasiados datos de una clase podrían acabar haciendo que el modelo acabara sesgado a favorecer la clasificación de clases de un modo u otro.
Criterio de satisfacción	Independientemente del número de clases que haya, ninguna puede tener un 10% más o menos de muestras que la media.
Modo de validación	Auto-explicativo.

Nombre	**Relevancia**
Descripción	El modelo debe de estar razonablemente equilibrado, asegurándose de no sobre ajustar ni desajustar datos y, como mínimo, alcanzar cierto equilibrio.
Criterio de satisfacción	El modelo debe alcanzar las mediciones indicadas en *goodness of fit* sin importar el set de validación ni el de entrenamiento.
Modo de validación	Entrenaremos el modelo 10 veces y, en cada uno de esos entrenos, mezclaremos las imágenes aleatoriamente en los sets de validación y de entreno. Luego nos aseguraremos de que se cumplen los objetivos. Especificados.

Nombre	**Eficiencia del entrenamiento y de la ejecución**
Descripción	Recursos usados para entrenar y ejecutar el modelo. Para ello mediremos el tiempo que se trata en ejecutar ambos.
Criterio de satisfacción	Para el entrenamiento esperamos que al principio se pueda entrenar en menos de 20 minutos, pero para la ejecución esperamos que la identificación se haga en menos de 5 segundos.
Modo de validación	(Análogo a la eficiencia del apartado anterior) Para identificación: al implementar, colocaremos cronómetros en la aplicación y nos aseguraremos de que, después de 50 identificaciones, la media sea menor de 3 segundos. Para el reentreno: al implementar, lanzaremos 10 entrenos del modelo y nos aseguraremos que la media sea menor a 20 minutos. En ambos casos usaremos la medición %%time de Python.

Nombre	**Calidad de código**
Descripción	El código que implementa y entrena el modelo debe de ser fácil de mantener y expandir.
Criterio de satisfacción	El código seguirá razonablemente los criterios PEP8.
Modo de validación	Se usará una herramienta de Python llamada pycodeStyle y seguiremos como mínimo el 80% de las recomendaciones que se nos haga (sin contar aquellas que sean contradictorias). Además cada celda del código estará comentada explicando qué se hace en cada paso.

Nombre	**Medición continua del rendimiento y la eficiencia del modelo**
Descripción	Tanto mientras que se entrene el modelo como durante su despliegue se tiene que hacer medición continuo de su rendimiento y eficiencia.
Criterio de satisfacción	Durante todo el desarrollo del proyecto no se producirán caídas de loss ni *accuracy* ni de eficiencia a la hora de entrenar y de identificar imágenes.
Modo de validación	Durante el desarrollo del proyecto se realizarán entrenos continuos y se asegurará de que a cada paso se cumplen los requisitos de eficiencia y fiabilidad (85% *accuracy*, >0.6 loss, identificación en >3 segundos, 20 minutos como límite de entreno). Se hará estos entrenamientos al final de cada Sprint y se apuntarán los resultados en un documento Excel.

7. ESPECIFICACIÓN DEL SISTEMA

En esta sección se hace la descripción de la especificación del sistema. En la primera parte se describen los casos de uso mediante un diagrama y su correspondiente explicación. En la segunda parte se describe el modelo conceptual, en el que se presenta una abstracción de los conceptos principales del sistema.

7.1. DIAGRAMA DE CASOS DE USO

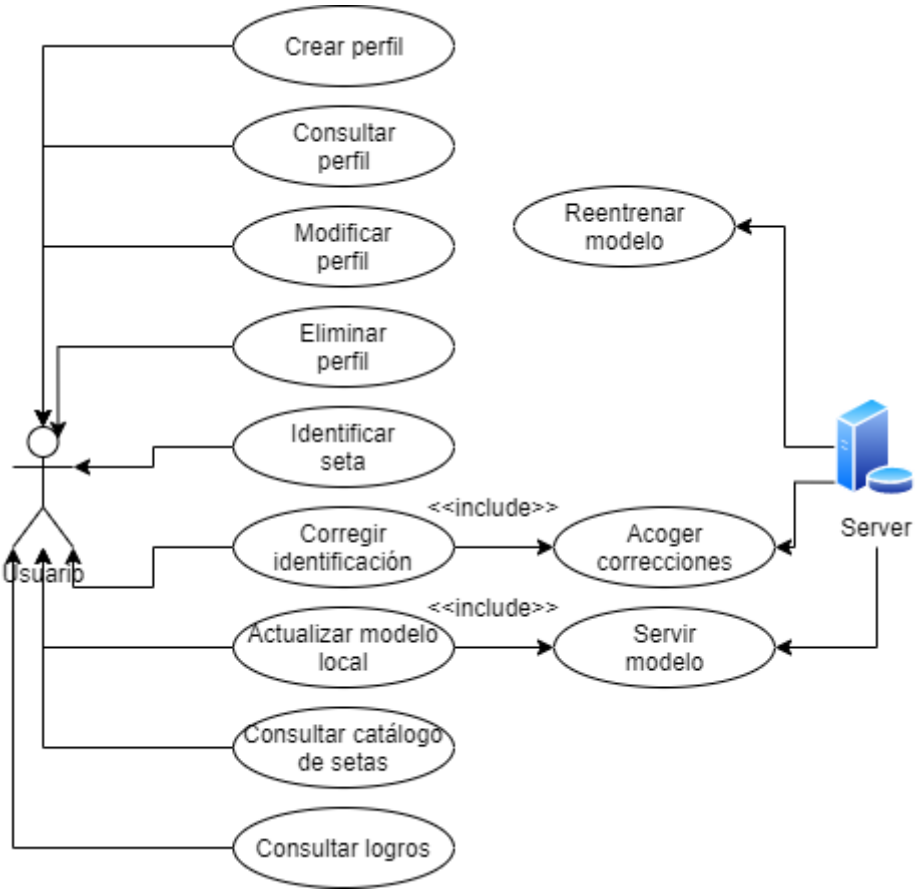

Ilustración 3: Diagrama de casos de uso

7.2. DESCRIPCIÓN DE LOS CASOS DE USO

7.2.1 Sistema servidor

El sistema servidor se refiere al sistema externo al que se conectarán todas las instancias de aplicación móvil a modo de referencia.

-Reentreno del modelo:

-Precondición: Ha llegado el día del reentreno semanal.

-Postcondición: El modelo se reentrena con las nuevas entradas.

-Descripción: Con tal de conseguir un modelo que mejore con el tiempo, es indispensable que éste se pueda actualizar con el tiempo con nuevos datos. Por eso, el servidor se encargará de recopilar todos estos datos y usarlos para reentrenar el modelo cada semana.

-Acoger correcciones:

-Precondición: Una instancia de la aplicación sube una nueva foto con una nueva clasificación.

-Postcondición: El sistema acoge esa petición, obtiene una nueva imagen y etiqueta y la guarda dentro de su dataset de imágenes y etiquetas de entreno.

-Descripción: Para que el modelo pueda mejorar con el tiempo, tiene que estar abierto a nuevos datos. Es por eso que el sistema deberá de estar preparado para acoger nuevos datos.

-Servir el modelo:

-Precondición: Una instancia de la aplicación solicita el nuevo modelo.

-Postcondición: El sistema de servidor envía a esa instancia el nuevo modelo.

-Descripción: En cuanto el servidor reciba una solicitud para descargarse una nueva versión del modelo, el sistema deberá de servir la versión más actualizada.

7.2.2 Usuario

El usuario es el encargado de usar el sistema para ejecutar y observa el funcionamiento de los servicios. Los casos de uso del rol Usuario son los siguientes.

-Creación de perfil:

-Precondición: El usuario ha iniciado la aplicación y no se ha registrado aún.

-Postcondición: El usuario se ha registrado indicando sus datos personales y su nombre de usuario.

-Descripción: El usuario se registra dentro del sistema como un usuario. A partir de ahora podrá emplear el resto de funcionalidades de la aplicación.

-Consulta de perfil:

-Precondición: El usuario está registrado.

-Postcondición: Se muestran los datos personales del usuario registrado por pantalla.

-Descripción: El usuario registrado en cualquier momento puede decidir consultar sus datos personales.

-Modificación de perfil:

-Precondición: El usuario está registrado.

-Postcondición: Se ha cambiado alguno de los campos de los datos personales.

-Descripción: El usuario registrado en cualquier momento decide cambiar alguno de los datos personales que tiene guardado en su perfil

-Realización de identificación:

-Precondición: El usuario está registrado y tiene una foto de una seta.

-Postcondición: El usuario sube una foto a predecir al sistema.

-Descripción: El usuario entrega una foto de una seta al sistema y el sistema retorna las posibilidades de que la seta a la que corresponde la foto pertenezca a alguna de las clases que conoce.

-Corrección de identificación:

-Precondición: El usuario registrado ha realizado la identificación de una seta y cree que el sistema se ha equivocado.

-Postcondición: El usuario indica al sistema cuál es la clase real de la seta.

-Descripción: Una vez que el sistema ha intentado identificar la seta, el usuario puede creer que la predicción es errónea, por lo que tiene la posibilidad de indicárselo a sí al sistema.

-Actualizar modelo local

-Precondición: Hay una nueva versión del modelo lista, el usuario desea actualizarlo y el dispositivo está conectado a internet

-Postcondición: Se envía una petición al servidor central para descargarse un nuevo modelo tflite, reentrenado con nuevos inputs.

-Descripción: Mediante la aplicación el usuario podrá descargarse una nueva versión del modelo reentrenado con las correcciones de los otros usuarios y las suyas, con lo que en teoría debería de poder adquirir un modelo más preciso.

-Consultar catálogo de setas:

-Precondición: El usuario está registrado.

-Postcondición: El usuario hace click en el catálogo para consultar los datos de una seta.

-Descripción: El usuario quiere conocer más acerca de algún tipo de seta, por lo que se decide ir al catálogo donde podrá recibir más información acerca de esa clase de seta específica.

-Consultar logros:

-Precondición: El usuario está registrado.

-Postcondición: El usuario hace click en el botón de logros para consultar su progreso.

-Descripción: El usuario quiere ver el progreso en logros por lo que se decide ir a la ventana correspondiente para comprobarlo.

7.3. MODELO CONCEPTUAL

En este apartado procederemos a mostrar el modelo conceptual del proyecto, que consiste en una abstracción de los conceptos más importantes del dominio. Como nuestro proyecto tiene dos partes, primero mostraremos el modelo conceptual de la parte de la aplicación móvil y después el de la parte del servidor.

7.3.1 Parte aplicación

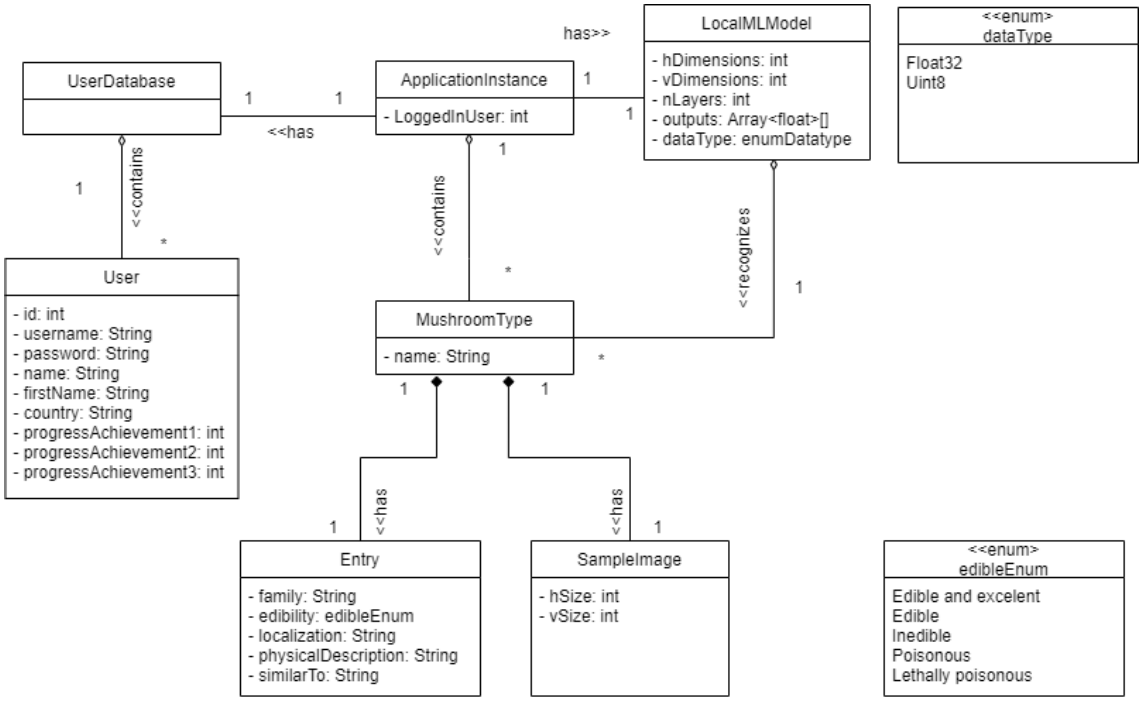

Ilustración 4: Esquema conceptual de la parte aplicación

ApplicationInstance: Representa cada una de las instancias de la aplicación ejecutándose en los dispositivos móviles de los clientes.

-LoggedInUser: variable que mantiene el identificador del usuario que actualmente ha iniciado la sesión.

LocalMLModel: El modelo ML instalado en la aplicación al cual se le harán consultas para realizar inferencias sobre imágenes. Hay un solo modelo local por instancia de aplicación.

-hDimension: Longitud horizontal esperada de las imágenes de entrada sobre las que se espera hacer inferencias.

-vDimension: Longitud vertical esperada de las imágenes de entrada sobre las que se espera hacer inferencias.

-nLayers: Número de capas de las imágenes de entrada sobre las que se espera hacer inferencias.

-outputs: Array de float del mismo tamaño que el número de clases a clasificar que muestra los porcentajes de que la imagen de entrada pertenezca a un tipo de setas u otro.

-dataType: Enum que especifica el tipo de números sobre los que se realizan los cálculos.

MushroomType: Cada una de las clases de setas presentes en la aplicación: ya sea las que el modelo es capaz de clasificar como las que están en la enciclopedia. Cada una de estas tendrá asignada una imagen de ejemplo y una entrada textual de la enciclopedia.

-name: String que indica el nombre del tipo de seta.

SampleImage: Imagen de seta de ejemplo que se corresponde a una de las clases que la aplicación es capaz de clasificar.

-hSize: Longitud de la dimensión horizontal de la imagen.

-vSize: Longitud de la dimensión vertical de la imagen.

Entry: Entrada de la enciclopedia de cada una de las clases de seta. Contiene toda la información relacionada respecto a esa especie de seta.

-family: Nombre de la familia de la familia a la que corresponde la clase de seta.

-edibility: Enum que indica lo comestible que es la seta.

-localization: Las áreas del mundo donde se puede encontrar dicha seta.

-physicalDescription: Descripción general del aspecto físico de la seta.

-similarTo: Especies de setas a las que esta especie en concreto es similar.

UserDatabase: Base de datos principal de la aplicación: contiene las tablas con los usuarios registrados y sus atributos.

User: Representa a los usuarios registrados en la aplicación.

-id: Identificador entero único.

-username: Nombre de perfil único.

-password: Contraseña para iniciar la sesión.

-name: Nombre real del usuario.

-firstName: Apellido real del usuario.

-country: País de origen del usuario.

-progressAchievement1: Progreso total que ha realizado el usuario en el logro 1.

-progressAchievement2: Progreso total que ha realizado el usuario en el logro 2.

-progressAchievement3: Progreso total que ha realizado el usuario en el logro 3.

Enum: EdibleEnum: Enumeración que clasifica cada seta en función de lo comestible que sea.

-Edible and excelent: Comestible y sabrosa, un manjar para el paladar.

-Edible: Comestible.

-Inedible: No comestible. No venenosa, pero no tiene valor nutricional alguno y puede causar malestar general.

-Poisonous: Venenosa. Consumir esta seta causa daños al cuerpo que pueden llevar a la muerte.

-Lethally poisonous: Mortalmente Venenosa. Consumir esta seta causa graves daños al cuerpo con certeza de muerte.

Enum: dataType: Enumeración que representa cada uno de los tipos de números con los que el modelo puede realizar su inferencia.

-Float32: Números reales que ocupan 32 bits.

-Uint8: Número representado en 8 bits, idóneo para representación de imágenes puesto que el rango de valores va de 0 a 255.

7.3.2 Restricciones de integridad: Parte usuario

MushroomType:

-name: El nombre debe de ser único a cada clase de seta (clave primaria).

User:

-id: Debe ser único (clave primaria).

-username: Debe ser único.

-password: No puede estar vacío.

-name: No puede ser vacío.

-firstName: No puede ser vacío.

-country: No puede ser vacío.

7.3.3 Parte servidor

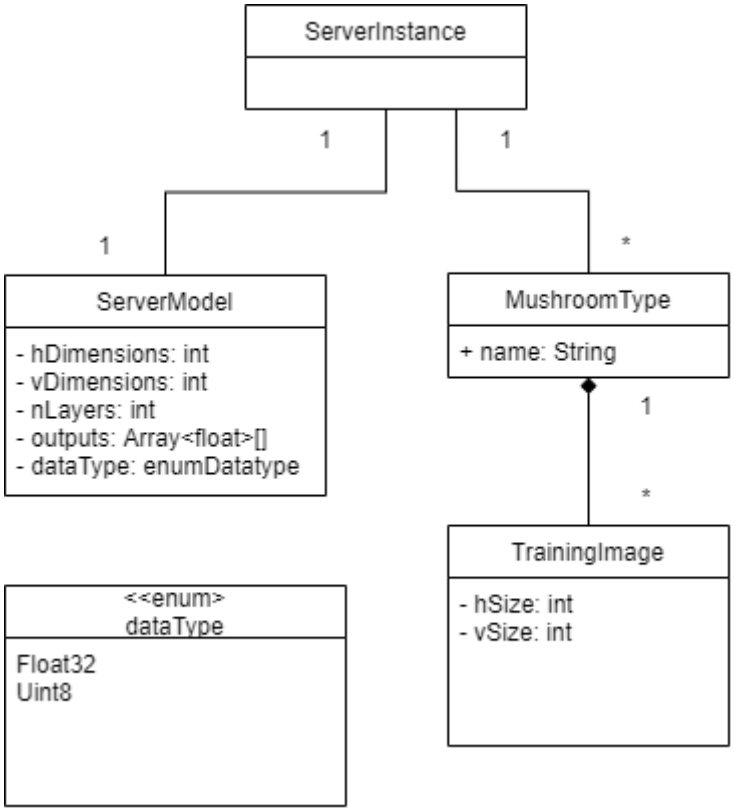

Ilustración 5: Modelo conceptual de la parte servidor

-ServerInstance: Representa al servidor. Contiene el modelo descargable del servidor y todas las utilidades necesarias para mantenerlo y entrenarlo.

-**ServerModel**: Modelo global entrenado por el servidor y descargable desde la aplicación. Es idéntico al modelo ML de la aplicación móvil.

-hDimension: Longitud horizontal esperada de las imágenes de entrada sobre las que se espera hacer inferencias.

-vDimension: Longitud vertical esperada de las imágenes de entrada sobre las que se espera hacer inferencias.

-nLayers: Número de capas de las imágenes de entrada sobre las que se espera hacer inferencias.

-outputs: Array de float del mismo tamaño que el número de clases a clasificar que muestra los porcentajes de que la imagen de entrada pertenezca a un tipo de setas u otro.

-dataType: Enum que especifica el tipo de números sobre los que se realizan los cálculos.

MushroomType: Cada una de las clases que el modelo se entrenará para reconocer. Coinciden con las clases de la aplicación. Al contrario que en la aplicación, cada tipo de seta tiene un paquete asignado con imágenes de esa clase.

-name: String que indica el nombre del tipo de seta.

Enum: dataType: Enumeración que representa cada uno de los tipos de números con los que el modelo puede realizar su inferencia.

-Float32: Números reales que ocupan 32 bits.

-Uint8: Número representado en 8 bits, idóneo para representación de imágenes puesto que el rango de valores va de 0 a 255.

7.3.4 Restricciones de integridad: Parte servidor

MushroomType:

-name: El nombre debe de ser único a cada clase de seta (clave primaria).

8. DISEÑO

En este apartado explicaremos el diseño. Para ello primero comenzaremos con un esquema general del sistema, en el que procuraremos ilustrar de forma entendible una descripción general de lo que se quiere hacer. Luego mostraremos la arquitectura lógica, en la que mostraremos la abstracción de la arquitectura que vamos a implementar. Más adelante describiremos la arquitectura física, en la que describiremos cómo los componentes de la arquitectura lógica se distribuyen físicamente. Seguido de esto explicaremos las operaciones de los componentes de la arquitectura. Por último, mostraremos un ejemplo de una consecución de casos de uso en la que mostraremos la interacción entre capas.

8.1. ESQUEMA GENERAL

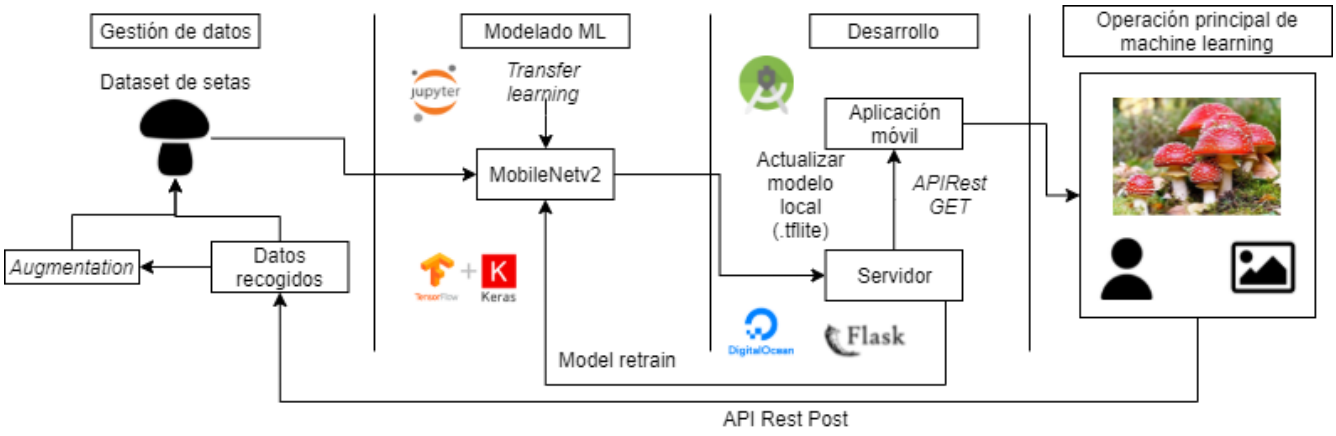

Ilustración 6: Esquema general de la aplicación

Con tal de poder entender mejor cómo va a funcionar la aplicación, hemos decidido incluir este esquema en el que se pretende explicar una idea general de cómo está construida la aplicación y cómo va a funcionar.

8.1.1 Gestión de datos

Lo primero que hay que hacer es recolectar los datos. Cada dato consistirá en una foto de una especie distinta de seta con una etiqueta que la clasifica en una clase.

Inicialmente empezaremos recolectando imágenes de un subconjunto de 10 clases de seta de diversos sitios de internet. Una vez hemos recolectado un conjunto suficientemente grande (por ahora especificaremos que sean un mínimo de 100 por clase, lo que nos deja a 1000 imágenes en total), aplicaremos técnicas de aumentación para incrementar el tamaño del dataset (rotaciones aleatorias y vueltas de imagen).

La idea es que al principio partimos de un dataset relativamente pequeño, pero luego confiamos en que los usuarios harán sus propias inclusiones lo que hará que se vaya volviendo más grande.

8.1.2 Modelado machine learning

Una vez tengamos un set de datos suficientemente grande, nos dedicaremos al modelado machine learning. Para ello usaremos Python en el entorno de desarrollo proporcionado por Anaconda, en los *notebook* de Jupyter.

Las librerías de tensorFlow y Keras nos proporcionarán las funcionalidades necesarias para desarrollar el modelo. Con tal de poder desarrollar un modelo eficiente rápidamente aplicaremos técnicas de Transfer Learning para utilizar el modelo ya existente MobileNetsv2 para adaptarlo a nuestros objetivos.

8.1.3 Desarrollo del servidor y de la aplicación

Una vez que tengamos el modelo creado, podremos empezar a trabajar en el servidor y en la aplicación.

El servidor lo desplegaremos en DigitalOcean utilizando Flask. El servidor contendrá el modelo, el dataset de imágenes y los algoritmos correspondientes para realizar los entrenamientos automáticamente periódicamente. Además, incluirá una interfaz API Rest a partir de la cual el usuario podrá conectarse para acceder a sus servicios.

La aplicación será desarrollada en Android Studio y será a partir de esta por la que el usuario podrá acceder a nuestros servicios. Así, desde la aplicación el usuario se podrá descargar el modelo (en formato .tflite, un formato diseñado específicamente para hacer que los modelos sean ligeros y eficaces ejecutados en móvil) y colaborar con sus propias contribuciones al servidor.

En resumen, la relación entre aplicación y servidor será la siguiente: el usuario se descarga del servidor el modelo, el usuario realiza la identificación de una seta con el modelo descargado, el usuario puede decidir corregir al servidor en caso de error y así añadir una nueva foto con una etiqueta de especie al dataset global, el servidor se encarga de reentrenar el modelo con todas las nuevas inclusiones de los usuarios y, por último y de vuelta, el usuario podrá descargarse el nuevo modelo ya actualizado.

8.1.4 Operación principal de machine learning

Como ya hemos dicho antes, la operación principal de machine learning consiste en identificar la especie de seta a partir de una imagen. El usuario buscará en su galería una imagen de una seta y la presentará a la aplicación, que retornará por pantalla las tres clases a las que esa seta tiene más probabilidades de pertenecer. Desde aquí, si la identificación no ha sido correcta, el usuario podrá indicárselo a la aplicación y su contribución con la nueva clasificación será incluida en el servidor para el siguiente reentreno.

8.1.5 Investigación de los outputs

El último paso será investigar los outputs. En particular nuestra intención será que, una vez hayamos implementado todas las funcionalidades será hacer una enumeración de los desafíos que hemos encontrado y la eficiencia de nuestra solución.

8.2. ARQUITECTURA LÓGICA

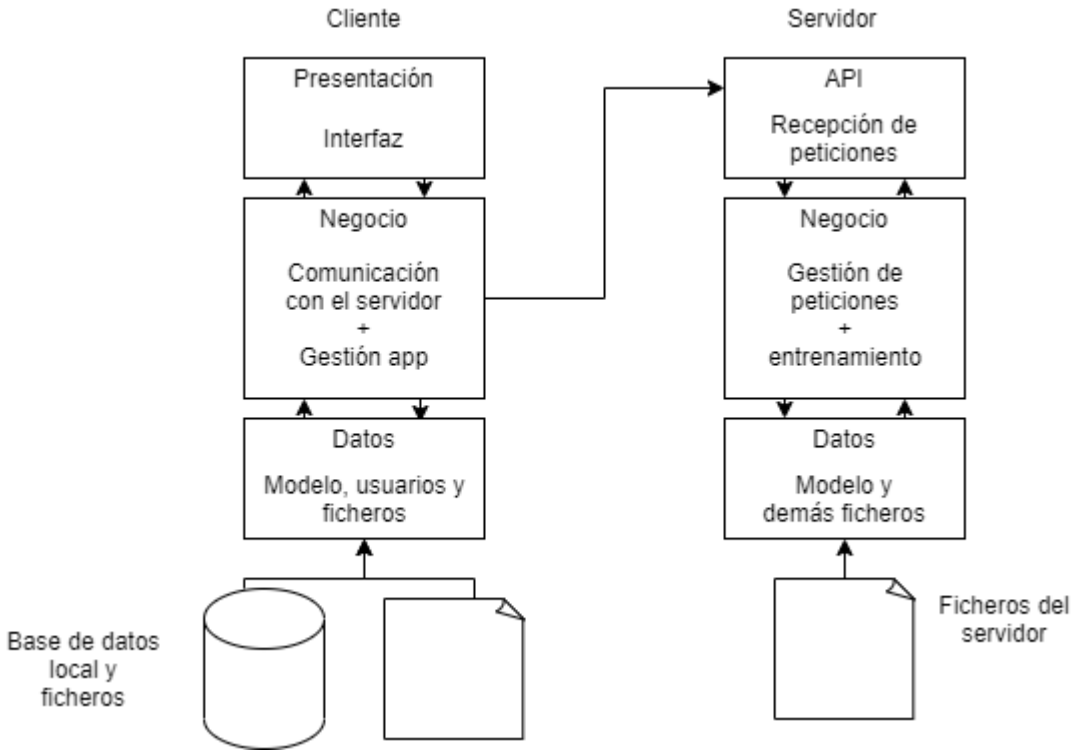

Ilustración 7: Diagrama de arquitectura lógica

El sistema que se desarrolla tiene, como ya hemos dicho, dos componentes principales. El primero es la aplicación móvil del cliente, el segundo es el del servidor. Aunque los componentes concretos serán distintos, ambos se pueden entender según el modelo de arquitectura en tres capas.

En la aplicación, la capa de presentación se corresponde a la interfaz que servirá para servirle y pedirle información al usuario: todas las actividades, menús y gráficos. La capa de negocio será la que se encargue de realizar todo el procesamiento de los datos del dominio del problema: hará la gestión general de la app (la gestión de usuarios, enciclopedia, logros, identificación machine learning, etc.) y la comunicación con el servidor remoto. Por último, la capa de datos será la encargada de gestionar los contenidos: accediendo a la base de datos y los ficheros internos y externos de la aplicación.

En el servidor, la API es encargada de recibir las peticiones realizadas desde las diversas aplicaciones móviles. Estas peticiones son enviadas a la capa de negocio donde son gestionadas: enviando la respuesta de vuelta o introduciendo sus contenidos en el servidor. Por último, la capa de datos del servidor consiste en los ficheros necesarios para realizar, concretamente, para las tareas de mantenimiento y entrenamiento del modelo.

8.3. ARQUITECTURA FÍSICA

Una vez que hemos visto la arquitectura lógica, ahora en este apartado presentaremos cómo se distribuyen cada una de estas capas en nuestros componentes físicos.

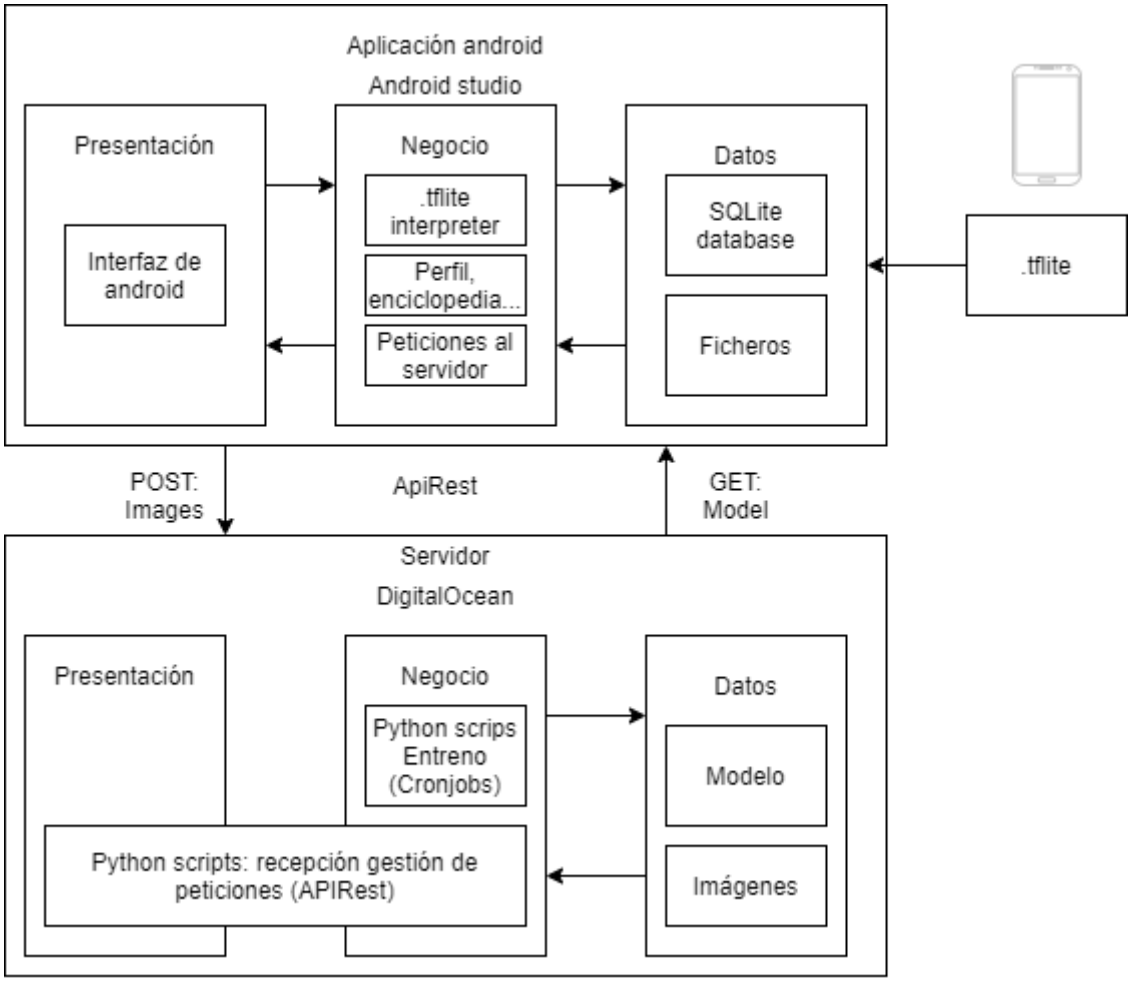

Ilustración 8: Arquitectura física

8.3.1 Aplicación

Como podemos observar, la capa de presentación será gestionada con la interfaz de Android: a partir de ésta se presentará y se pedirá datos al usuario.

En la capa de negocio se realizará todo el procesamiento de los datos. El modelo (que está implementado en tensorflow y en formato .tflIte) será interpretado desde aquí con el tensorflow lite interpreter. Las peticiones al servidor se harán con Okhttp3 y retrofit2, librerías que sirven para gestionar peticiones http. El resto de funcionalidades se realizarán utilizando en su mayoría mediante las librerías de Java, aunque la gestión de los usuarios tendrá un componente SQLite para contactar con la base de datos correspondiente.

Por último, la capa de datos estará distribuida entre archivos internos y externos a la aplicación. Los archivos internos incluyen la información necesaria para las entradas de la enciclopedia de las setas (la información y la imagen que las representa) y de gestión general de la aplicación (por ejemplo, el icono de la aplicación, la imagen de la cubierta, etc.). El modelo es el único archivo que se guarda en documentos externos porque es el único modo posible de hacerlo intercambiable en tiempo de ejecución.

8.3.2 Servidor

La capa de presentación se refiere a la interfaz Restful API que ofrecerá para atender a las peticiones de los clientes cliente. Se construirá mediante un script de Python usando Flask.

La capa de negocio está compuesta por dos scripts. El primero es el mismo que se encuentra en la capa de presentación, solo que aquí se encarga de gestionar las peticiones. El segundo es el que se encarga de entrenar y desplegar el modelo y dejarlo listo para descarga de nuevo desde la aplicación. Por último, hay un componente Cronjobs que se encarga de ejecutar este segundo script cada último día de la semana para realizar los reentrenos.

La capa de datos consiste básicamente en el modelo .tflite listo para descarga y el paquete de imágenes a partir del cual se realizan los entrenos del modelo, el mismo paquete de imágenes a los que los usuarios desde su aplicación podrán hacer contribuciones.

8.4. OPERACIONES DE LOS COMPONENTES ARQUITECTURA: APLICACIÓN

En este apartado ahora hablaremos en detalle de las operaciones de cada capa. Explicaremos cada una de ellas en qué consiste y las operaciones concretas que se realizan.

8.4.1 Capa de presentación

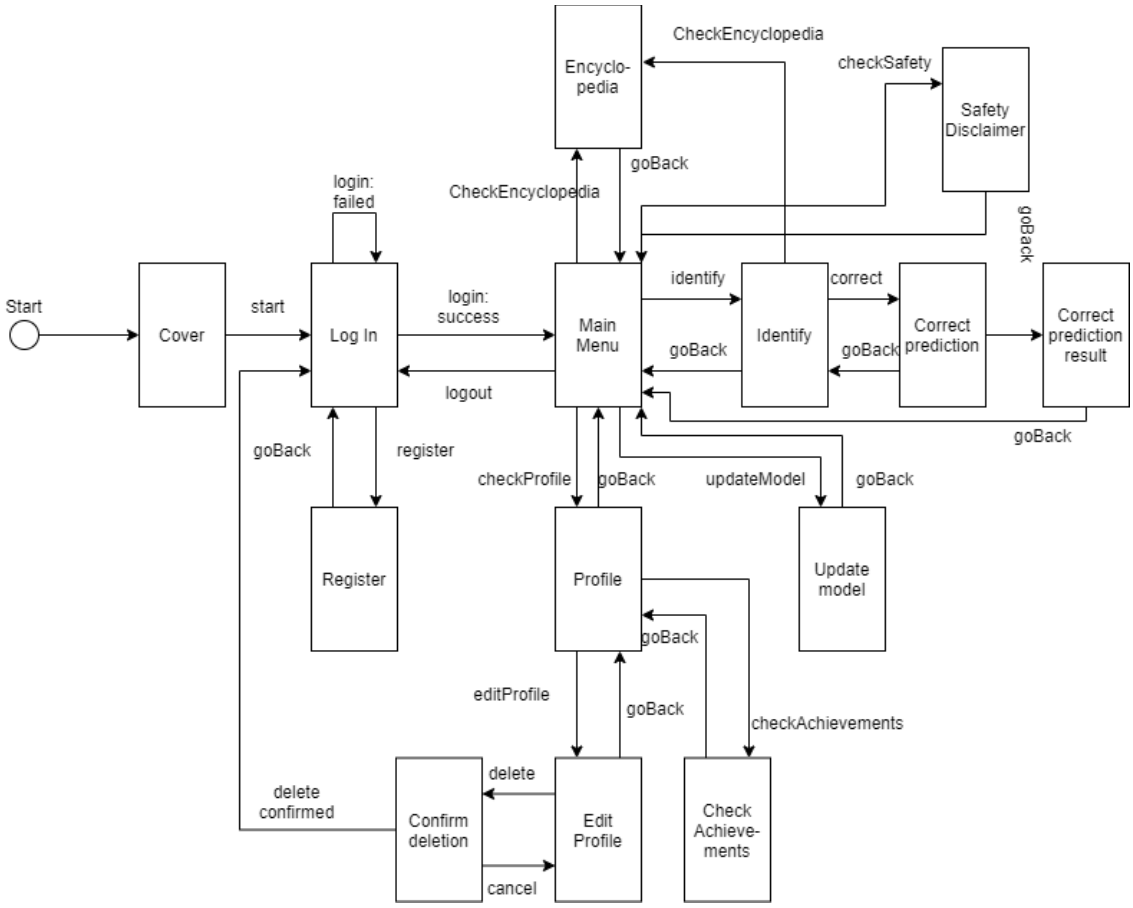

Ilustración 9: Diagrama de flujo de ventanas

En la ilustración 11 podemos ver el flujo general de ventanas. El usuario empezará en la cubierta desde donde podrá acceder a la ventana de login. Ahí podrá autenticarse o registrarse. Una vez autenticado, accederá al menú principal desde donde podrá acceder a las funcionalidades principales de la aplicación: identificar setas, ver su perfil, actualizar el modelo ML, consultar la enciclopedia y consultar las normativas de seguridad. Desde la identificación podrá acceder a la corrección de setas y desde la página de perfil podrá consultar sus logros, editar o modificar el usuario

Ahora entraremos más en detalle en cada una de las ventanas de la aplicación.

8.4.1.1 Cover activity

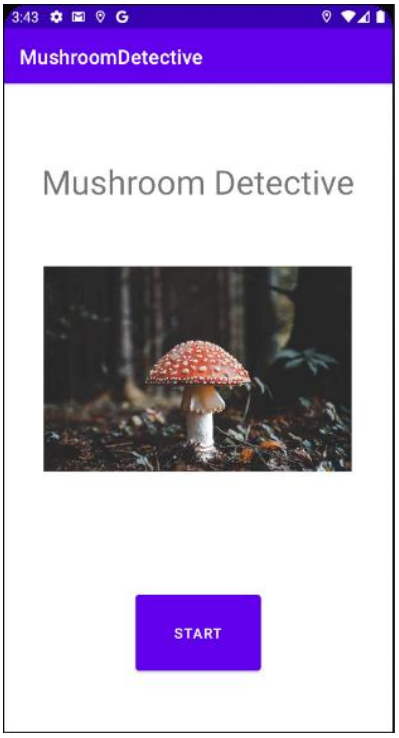

Ilustración 10: Cubierta

Esta será la primera actividad que verá el usuario. La única funcionalidad que tiene es presentar al usuario la aplicación. Para proceder el usuario deberá de presionar el botón *Start* y será enviado a la pantalla de Log in.

8.4.1.2 Log in activity

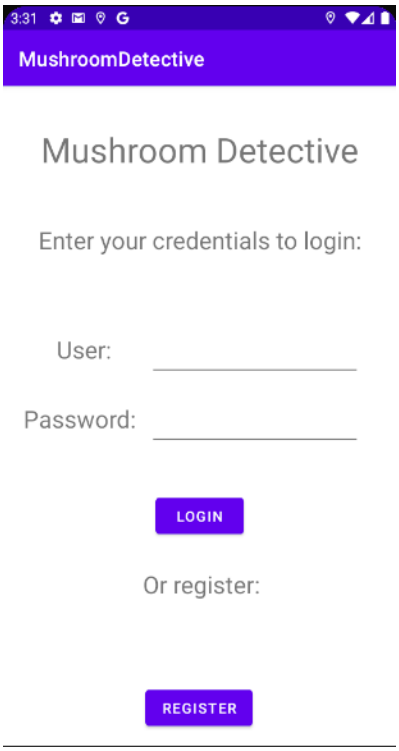

Ilustración 11: Actividad de log in

La siguiente actividad que verá el usuario es la de Log in. Aquí tendrá que introducir su nombre de usuario y su contraseña para acceder a las funcionalidades de la aplicación. Si no tiene credenciales, puede registrarse haciendo click en el botón *Register*. Una vez haya introducido las credenciales y una vez sean válidas, hacer click en el botón *Login* le llevará al menú principal.

8.4.1.3 Register activity

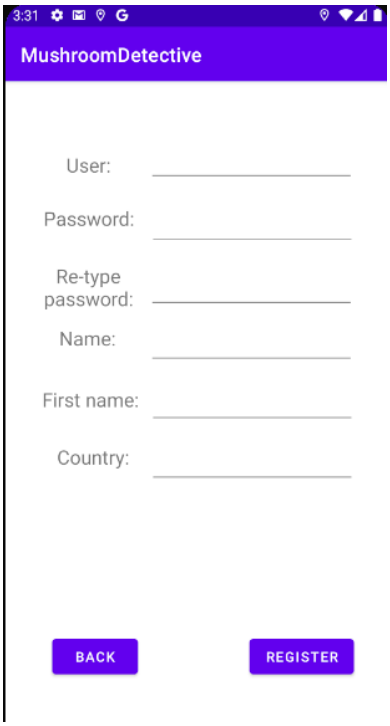

Ilustración 12: Actividad de registro

Esta es la actividad de registro, accesible desde la pantalla de Login. El usuario deberá de introducir sus datos personales y la contraseña dos veces. Si alguno de los campos está vacío, la contraseña no se corresponde con la contraseña repetida o el nombre de usuario ya está escogido por otro usuario registrado en el sistema, saldrá un error por pantalla informándole sobre lo sucedido. Una vez el usuario haya insertado unos datos válidos, será registrado en la aplicación y podrá usar esas credenciales para hacer login desde la actividad anterior.

8.4.1.4 Main menu activity

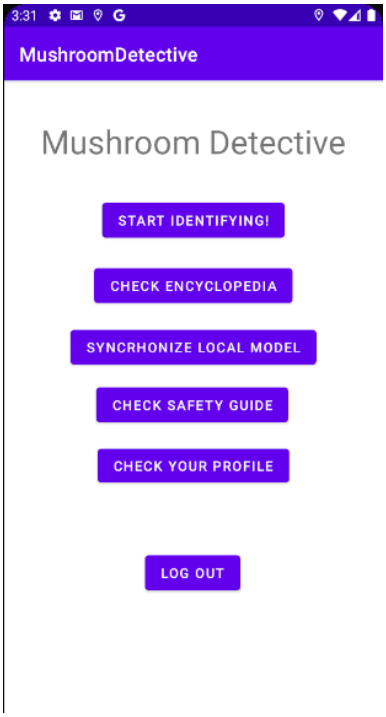

Ilustración 13: Actividad de menú

Una vez el usuario haya hecho *login*, podrá acceder al menú desde donde tendrá acceso a las funcionalidades de la aplicación. *Start identifying* le llevará a la interfaz de identificación de imágenes. *Check encyclopedia* le llevará a la actividad de la enciclopedia. *Synchronize local model* a la actividad para gestionar versiones del modelo salvado en local. *Check safety guide* le llevará a una actividad donde se describen prácticas de seguridad a tener en cuenta a la hora de usar esta aplicación. *Check your profile* llevará a la actividad desde donde podrá consultar, editar y borrar los datos de su perfil. Finalmente, *logout* le permitirá salir del menú y volver al menú de *login*.

8.4.1.5 Profile activity

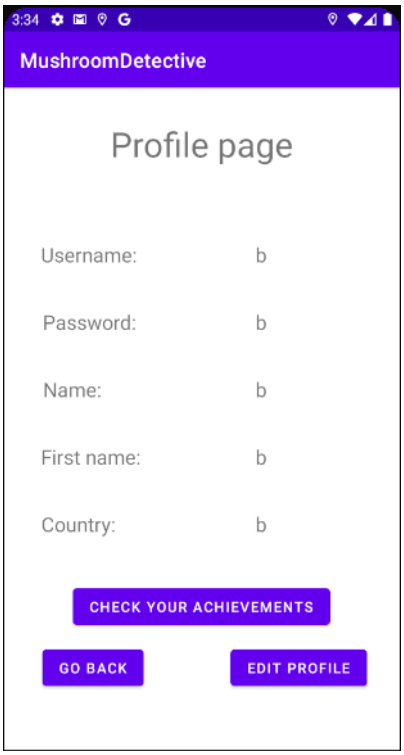

Ilustración 14: Actividad de consulta de perfil

Esta es la actividad de consulta: aquí el usuario podrá consultar y gestionar todo lo referido a su perfil. Desde aquí también podrá dirigirse al menú de edición de perfil, consultar el progreso en los logros o volver al menú principal.

8.4.1.6 Edit profile activity

Ilustración 15: Actividad de edición de perfil

Desde esta actividad el usuario podrá editar todo lo referente a su perfil. Podrá cambiar sus datos, pero siguiendo las mismas restricciones que la ventana de login (la password debe de ser reintroducida dos veces, ningún campo puede estar vacío y el nombre de usuario no debe estar repetido con el nombre de ningún otro usuario registrado en la aplicación). Desde aquí podrá dirigirse a la ventana de eliminación de perfil o retroceder al menú principal.

8.4.1.7 Confirm deletion

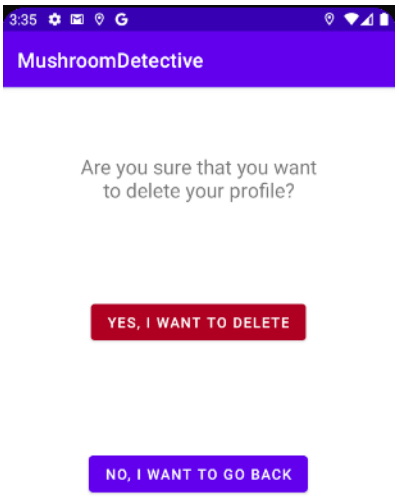

Ilustración 16: Actividad de confirmación de eliminación

En caso de que el usuario desee eliminar su perfil, se enfrentará a esta pantalla. Aquí se le pedirá que confirme la acción. Si se da marcha atrás volverá a la página de perfil. Si por otro lado confirma, el perfil será eliminado de la aplicación y se abrirá la pantalla de login para identificarse con otro usuario o registrarse de nuevo.

8.4.1.8 Check achievements activity

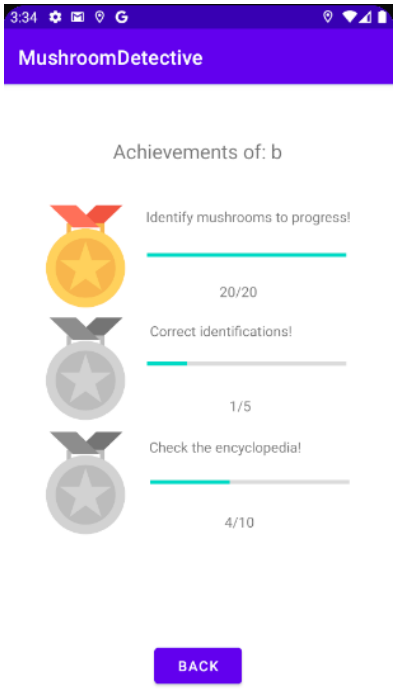

Ilustración 17: Actividad de consulta de logros

Desde la página de perfil el usuario podrá acceder a la actividad de consulta de logros. Aquí se muestran los logros disponibles, su progreso y sus objetivos. A medida que el usuario emplee la aplicación, se irá haciendo seguimiento de sus acciones y éstas quedarán registradas aquí adecuadamente.

8.4.1.9 Update model activity

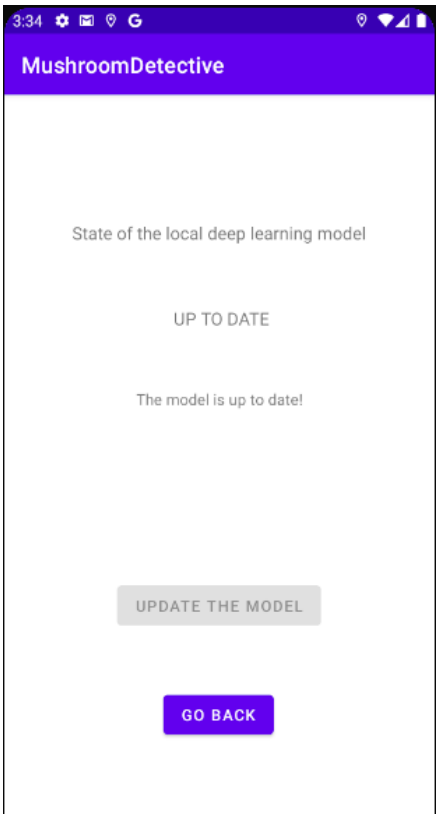

Ilustración 18: Actividad de actualización del modelo

Para poder usar la aplicación en su máxima efectividad, el usuario deberá de realizar actualizaciones del modelo periódicamente. Desde aquí se puede hacer seguimiento del estado actual. Hay tres estados posibles *UP TO DATE*, *NOT UP TO DATE* y *INEXISTENT*. En el primer caso el modelo está actualizado con la versión más reciente del servidor, por lo que no es necesario realizar ninguna actualización. En el segundo caso el modelo es un poco antiguo, por lo que conviene actualizarlo. En el tercer caso el modelo no se encuentra y, por lo tanto, convendrá pedirle al usuario que actualice para descargarse el modelo del servidor.

En los dos últimos casos, el botón de *Update model* se desbloqueará. El botón de *Go back* devolverá al usuario al menú principal.

8.4.1.10 Identify activity

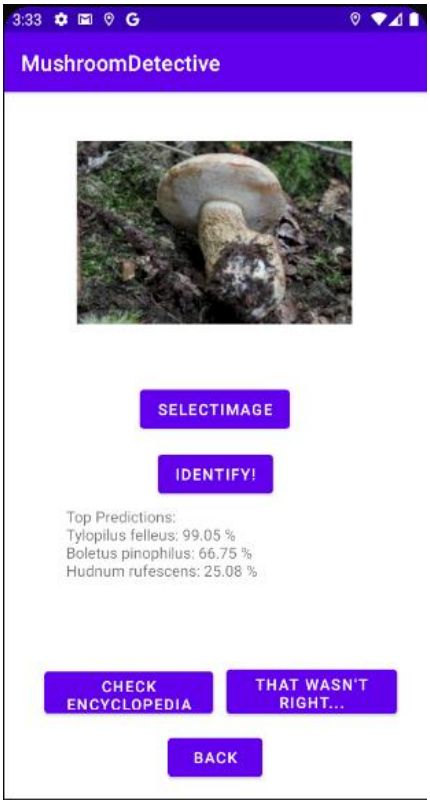

Ilustración 19: Actividad de identificación de setas

La funcionalidad principal de la aplicación accesible desde el menú principal. En cuanto el usuario acceda a esta ventana, *Check Encyclopedia* y *That wasn't right…* estarán bloqueados. Primero, el usuario deberá hacer click en *Select image*, esto le llevará a su galería desde donde podrá escoger la imagen a ser identificada. Luego hará click en *Identify!* que mostrará por pantalla el porcentaje de pertenencia de la imagen a cada una de las clases de la aplicación. En este momento, *Check Encyclopedia* y *That wasn't right…* se desbloquearán. Hacer click en el primer botón llevará al usuario a la entrada de la enciclopedia de la aplicación correspondiente a la imagen. Hacer click en el segundo botón llevará a la ventana de corrección de identificación. El botón *Back* devolverá al usuario al menú principal.

8.4.1.11 Correct prediction activity

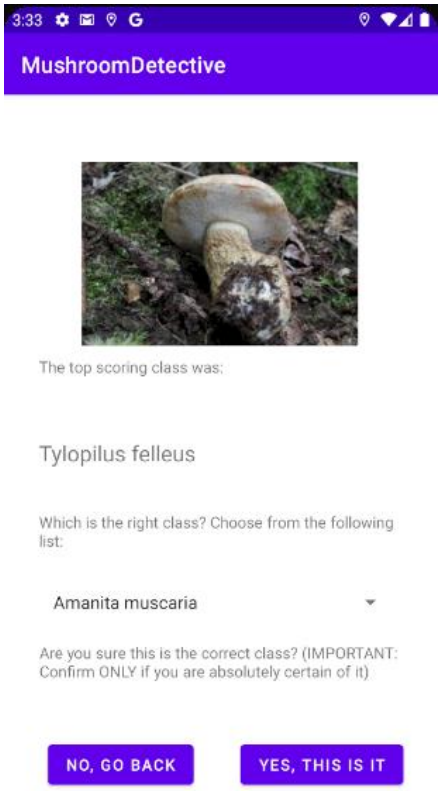

Ilustración 20: Actividad de corrección de imagen

Si al realizar una identificación el usuario ve que el modelo se ha equivocado, podrá corregirlo. Si ésa es su intención, se encontrará con esta ventana. El usuario a partir de aquí podrá escoger la clase verdadera a la que pertenece la foto y enviar la imagen con la nueva etiqueta al servidor remoto. Si quiere volver atrás, solo deberá pulsar *No, go back*.

8.4.1.12 Correct prediction result

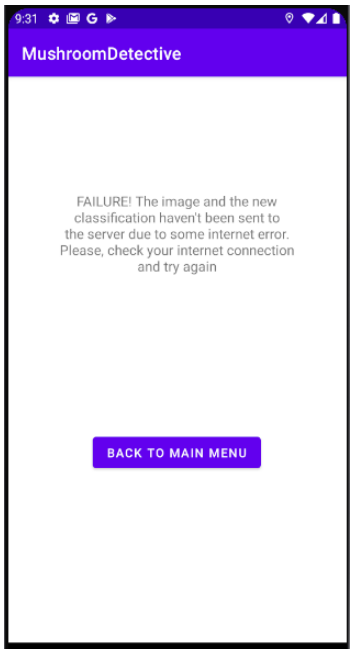

Ilustración 21: Actividad de resultado de la corrección

Una vez que el usuario haya realizado una corrección de etiqueta, será dirigido a esta pantalla en la que podrá leer el resultado de su corrección. Si se pudo conectar al servidor se le indicará que la operación ha sido un éxito. Si no ha sido posible establecer la conexión se le explicará al usuario lo que ha pasado y se le pedirá que lo intente de nuevo con conexión a internet. Por último, el botón le enviará de vuelta al menú principal.

8.4.1.13 Encyclopedia activity

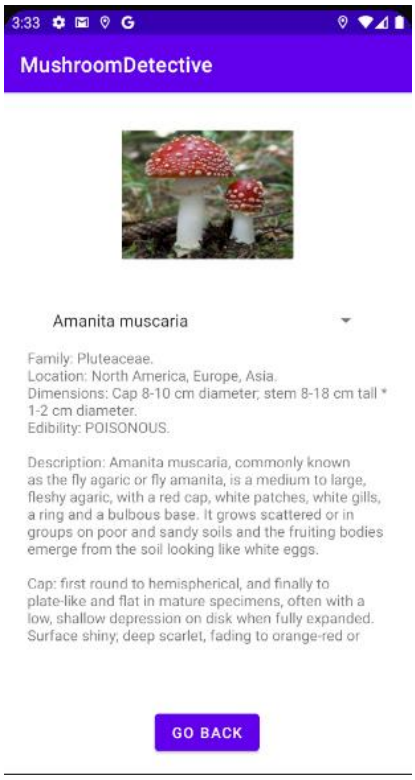

Ilustración 22: Actividad de enciclopedia

Tanto desde el menú principal como desde la identificación de setas el usuario podrá acceder a la enciclopedia para consultar toda la información relevante de una especie de setas. Para ello deberá de escoger la clase de seta desde el Spinner debajo de la imagen. Podrá ver una imagen de ejemplo de la seta y la descripción en la que se detalla, entre otras cosas, aspecto físico, familia, dimensiones y si es comestible o no.

8.4.1.14 Safety disclaimer

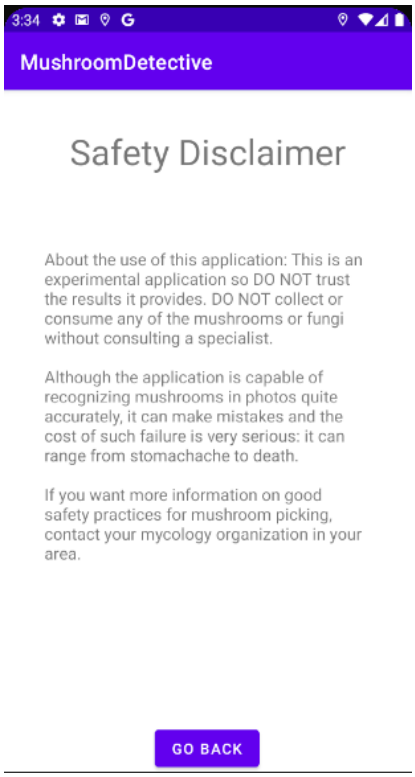

Ilustración 23: Actividad con el aviso de seguridad

La última actividad es el aviso de seguridad. Teniendo en cuenta los riesgos que entrañan la recogida de setas y la posibilidad de que la aplicación cometa algún error a la hora de identificar la especie, es importante informarle de los riesgos y de buenas prácticas de seguridad a llevar a cabo. Entre ellos se le pide que no se fíe de los resultados de la aplicación para llevar a cabo la decisión final y que tome como última palabra la de los expertos en este campo.

8.4.2 Capa de negocio

8.4.2.1 Gestión de usuarios:

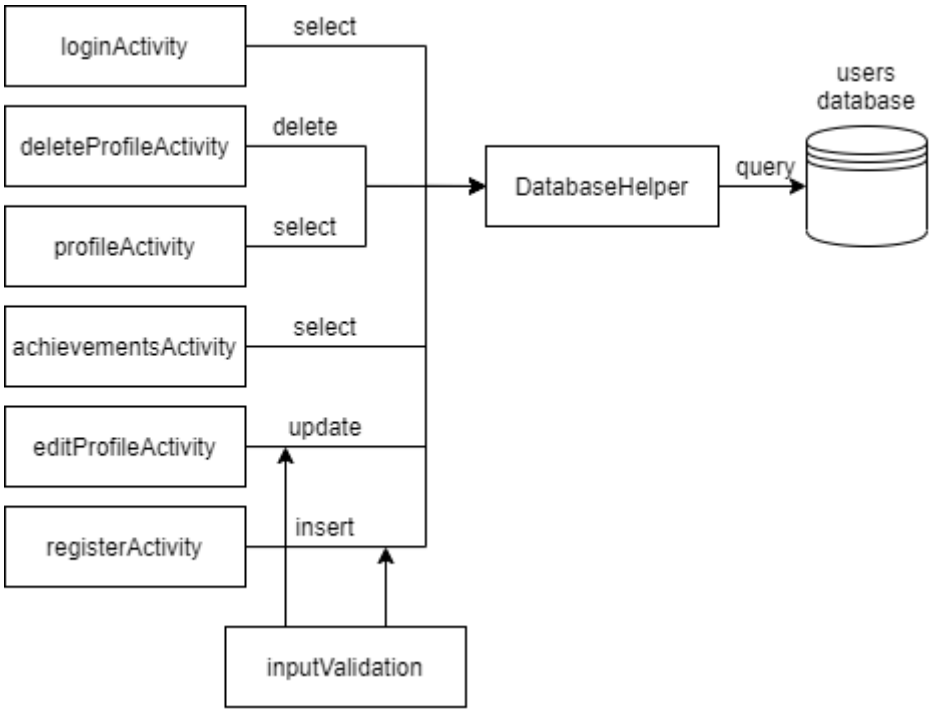

Ilustración 24: Secuencia de gestión de usuarios

La primera operación que se realiza es la gestión de usuarios. En esta operación están involucradas las clases User, ApplicationHelper, DatabaseHelper e InputValidation.

La clase User es la que contiene todos los datos del usuario: id, username, password, name, firstName, country, achiv1, achiv2 y achiv3. Estos datos son guardados en una base de datos SQLite interna del modelo a la que se accede mediante la clase DatabaseHelper. La clase ApplicationHelper mantiene la permanencia de ciertos datos a lo largo de la aplicación, entre ellos, el usuario que está actualmente usando la aplicación. InputValidation por otro lado es una clase auxiliar que se encarga de validar que los inputs a la hora de hacer registro o edición de perfil sean válidos.

Así, por ejemplo, si un usuario se registra y o modifica sus datos, rellenará los campos con sus credenciales que serán verificados por InputValidation. Si es un registro, esos campos se convertirán en un User que será incluido en la base de datos a través de DatabaseHelper, si es una modificación, de nuevo a través del DatabaseHelper se buscará el usuario y se modificarán sus datos.

ApplicationHelper se encargará de mantener de actividad en actividad qué usuario ha iniciado la sesión.

Cabe añadir que ApplicationHelper y DatabaseHelper se encuentran en todos los sitios de la aplicación, puesto que es necesario para poder hacer seguimiento continuo de las acciones del usuario para actualizar sus logros.

8.4.2.2 Consulta de enciclopedia:

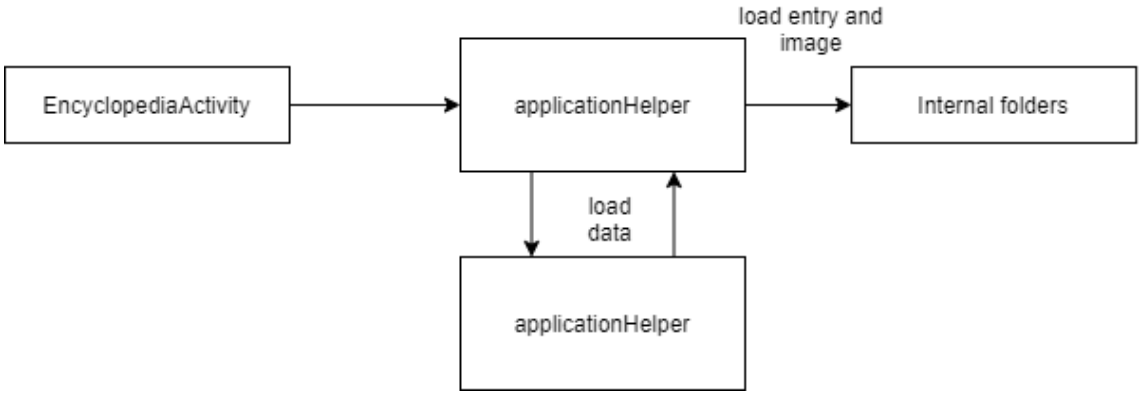

Ilustración 25: Secuencia de enciclopedia

La siguiente operación es la consulta de enciclopedias. En esta están involucradas la clase Mushroom, ApplicationHelper y archivos internos correspondientes (entrada de la enciclopedia e imagen de ejemplo).

Cada Mushroom tiene tres componentes, nombre, nombre del archivo con el contenido de la enciclopedia y la dirección de la imagen. ApplicationHelper se encarga de guardar un array de Mushroom con todas las clases clasificables de la aplicación. Cuando el usuario hace la consulta de una entrada de la enciclopedia para una seta, desde ApplicationHelper se carga el objeto Mushroom correspondiente, del cual a su vez se cargan la imagen y la entrada de la enciclopedia.

8.4.2.3 Identificación de setas:

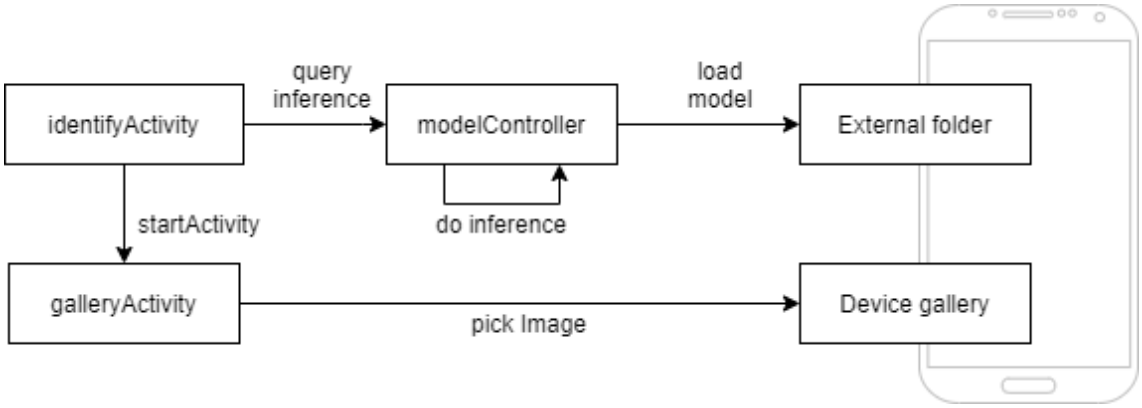

Ilustración 26: Secuencia de identificación de setas

La identificación de setas involucra al archivo .tflite que contiene el modelo ML y las herramientas de la biblioteca de tensorflow.

El archivo se encuentra en una carpeta del móvil externa a la aplicación. Cuando el usuario lanza una identificación, se llama al controlador para que busque el archivo en la dirección correspondiente. Una vez cargado el archivo, se toma la imagen que el usuario habrá importado desde su galería, se convierte a tensorImage y con ayuda del intérprete tensorflowlite se hace la inferencia sobre el modelo. Finalmente, se toman los resultados del modelo y se imprimen por pantalla.

8.4.2.4 Corrección de setas:

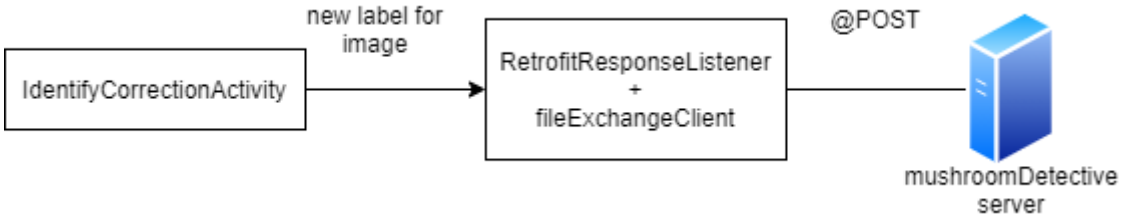

Ilustración 27: Secuencia de corrección de setas

La corrección de setas involucra a la clase RetrofitResponseListener y fileExchangeClient.

Cuando el usuario decide corregir una inferencia del modelo, introduce una nueva etiqueta a la imagen de la seta y hace click en “Aceptar”. Hacer eso lleva a que se haga una petición POST (interfaz que provee fileExchangeClient) al servidor en el que se pasan por parámetros la imagen como String codificado en Base64 y la nueva etiqueta.

RetrofitResponseListener actúa como una interfaz que nos ayuda a comprobar si la petición ha sido un éxito o un fracaso.

8.4.2.5 Actualización del modelo:

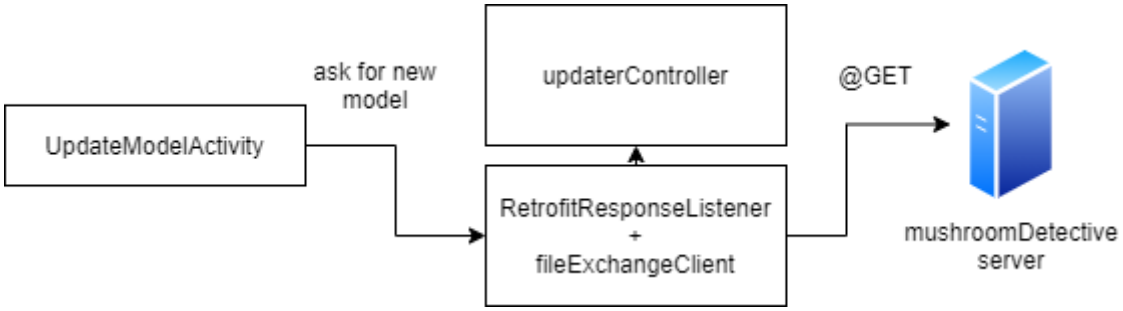

Ilustración 28: Secuencia de actualización del modelo

La actualización involucra a la clase updaterController, RetrofitResponseListener y fileExchangeClient.

Cuando el usuario quiere actualizar el modelo utilizará la interfaz RetrofitResponseListener para lanzar una petición GET al servidor. Si tiene conexión a internet, el nuevo modelo se descargará y reemplazará al viejo (o creará uno nuevo si no existía el modelo). De nuevo, fileExchangeClient se encarga de comprobar si ha sido un éxito o un fracaso. Updater controller se encargará de recibir el archivo y escribirlo en disco.

8.4.3 Capa de persistencia de datos

8.4.3.1 Usuarios

El primer componente de la capa de datos que vamos a ver es la base de datos de Usuarios. Se gestionan mediante SQLite. Consiste en una base de datos local que contiene una tabla llamada Users. Esta tabla dispone de una columna por cada atributo de usuario: id (declarada como clave primaria), username (que debe ser único), password, name, firstName, country, achiv1, achiv2 y achiv3 (el progreso correspondiente a cada uno de los logros). Ninguna de las columnas puede estar vacía.

Esta base de datos se guarda internamente y se puede acceder a ella mediante una clase llamada DatabaseHelper, que se encarga de gestionar las operaciones de consulta, modificación y borrado.

8.4.3.2 Información de setas

Debido a que para cada clase de seta es necesario guardar un documento de texto y una imagen, estimamos oportuno guardarlo a partir de archivos internos de la aplicación debidamente identificados y separados. Otro archivo interno nos indica cuales son las clases existentes en la aplicación y las direcciones internas a las que acceder para cargar cada recurso. Este mismo archivo se utiliza también para cargar las etiquetas de las setas para las otras funcionalidades de la aplicación.

8.4.3.3 Modelo Deep Learning

Debido a que el modelo debe de ser modular, esto es, cambiable en tiempo de ejecución, no puede guardarse como archivo interno de la aplicación, por lo que es necesario guardarlo como archivo externo. El modelo se guarda en la carpeta “Documentos” del móvil en el que habita la aplicación, dentro a su vez de una carpeta propia. Cuando se quiere realizar una consulta, se busca el archivo en la carpeta correspondiente.

Si por algún motivo esa carpeta no existiera o no se pudiera encontrar el modelo, se dirigirá la inferencia a una segunda versión del modelo no actualizable guardada en los archivos internos de la aplicación.

8.5. OPERACIONES DE LOS COMPONENTES ARQUITECTURA: SERVIDOR

8.5.1 Capa de API

La API del servidor consiste en los puntos finales de las peticiones GET y POST que se pueden llamar desde la aplicación. Cuando el servidor se pone en marcha se le pone a la “escucha” de dichas peticiones mediante un script programado con Python y Flask. La gestión de estas peticiones se realiza en el mismo script, pero pertenecen estas a la capa de negocio.

8.5.2 Capa de negocio

8.5.2.1 Gestión de las peticiones GET y POST

El mismo script explicado en el apartado anterior se encarga de gestión de las dos peticiones.

En caso de GET, se busca el modelo .tflite entre los archivos del servidor y ser envía a la dirección de origen.

En caso de POST, se recoge la petición, se extrae la nueva etiqueta y la imagen codificada, se descodifica la imagen, se le asigna un identificador único y, finalmente, se guarda en la carpeta correspondiente a su clase.

8.5.2.2 Entreno del modelo ML

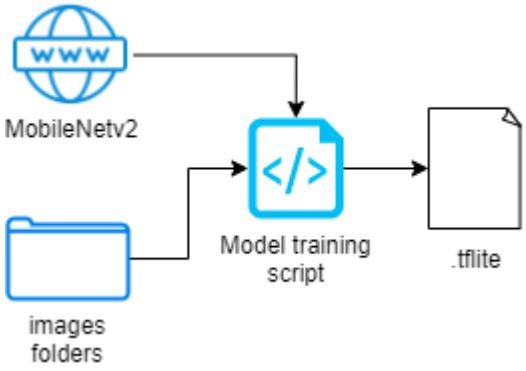

Ilustración 29: Entreno del modelo ML

Una de las operaciones que deberá de efectuar el servidor es el entreno del modelo. Se realiza mediante la ejecución de un script en Python que se ejecuta periódicamente cada semana. En el apartado de la implementación procederemos a explicar cómo se desarrolla el entreno del modelo en detalle, pero aquí nos limitaremos a explicar los aspectos básicos. Gracias a CronJobs, cada día final de la semana se ejecuta el Model training script, que automática recoge las imágenes presentes en las carpetas del servidor (separadas por clases), se descarga el modelo MobileNetsv2 y a partir de estos componentes reescribe el documento .tflite, que es el que los usuarios podrán descargarse.

8.5.3 Capa de datos

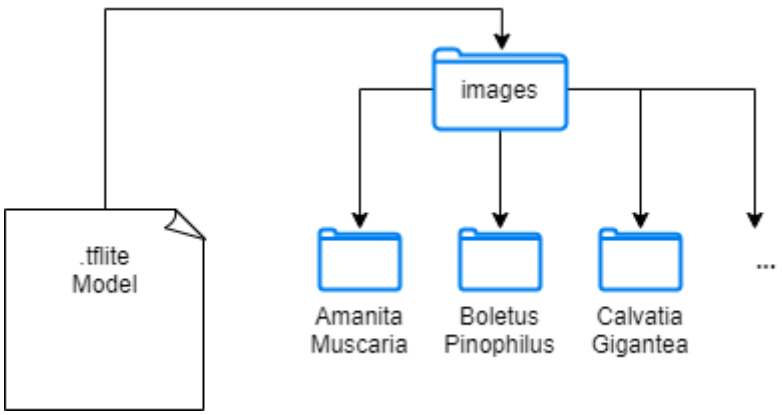

Ilustración 30: Capa de datos del servidor

La capa de datos del servidor consiste en el modelo .tflite descargable y en la carpeta de imágenes. La carpeta de imágenes contiene a su vez subcarpetas que contienen las imágenes de setas que usaremos para entrenar el modelo. Estas subcarpetas están nombradas con el nombre de la case de seta de la que contienen imágenes y es donde van a parar las imágenes de setas subidas por los usuarios.

8.6. INTERACCIÓN ENTRE CAPAS

Con tal de facilitar la comprensión de esta arquitectura, aquí demostraremos cómo interactúan los diversos componentes de las diversas capas a través de la consecución de diversas historias de usuario.

Supongamos que María utiliza la aplicación: ha encontrado una seta muy rara de la cual conoce su especie, pero tiene curiosidad por saber qué es lo que dice la aplicación.

1) María pone en marcha la aplicación y hace click en *Start* para dirigirse a la pantalla de login.

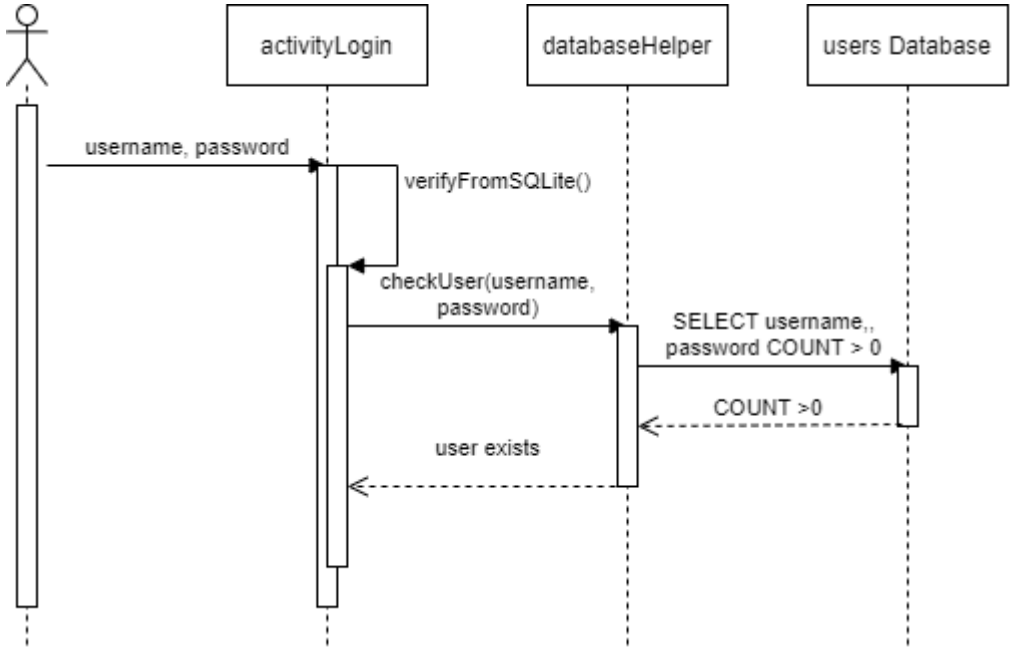

Ilustración 31: Diagrama de secuencia de login

2) En la pantalla del login introduce sus credenciales, esto es, su nombre de usuario y su contraseña. Al hacer esto descendemos a la capa de negocio desde donde se toman los inputs y, mediante el DatabaseHelper, se lanza una petición a la capa de datos para comprobar si existe un usuario con ese nombre de usuario y esa contraseña, en concreto,

se lanza una petición SQL a la tabla de usuarios de la base de datos. Si el usuario existe y las credenciales son correctas, se le deja pasar.

3) En la pantalla de menú principal, María hace click en la opción *Start identifying*. Eso le llevará a la ventana de identificación.

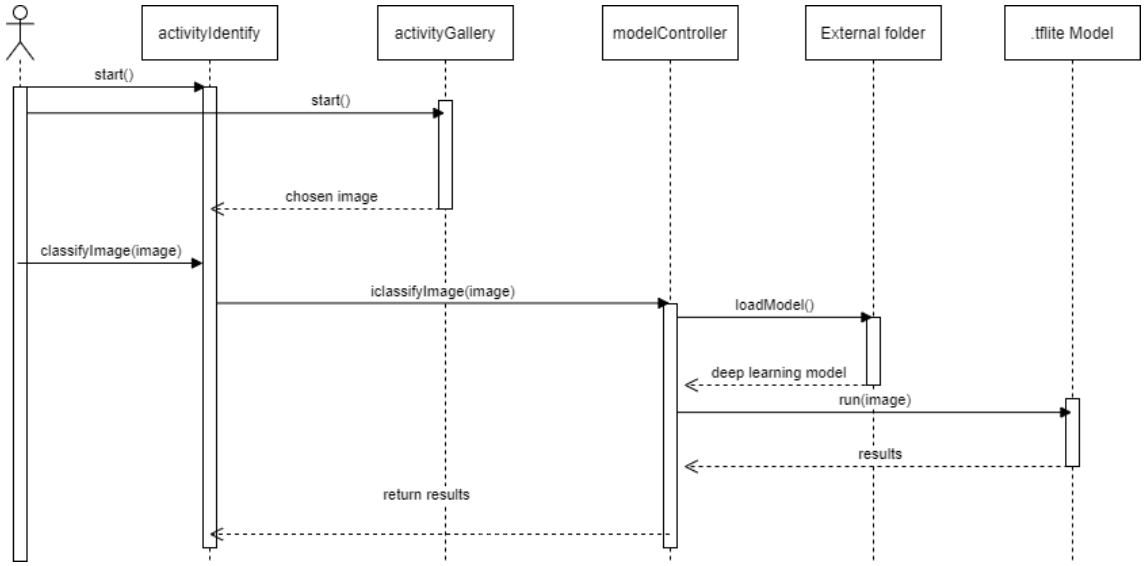

Ilustración 32: Diagrama de secuencia de selección de imagen, carga de modelo e inferencia

4) Una vez ahí, María selecciona una foto de la galería haciendo click en *Select Image*. Una vez seleccionada, hace click en *Identify!* En este momento la capa de negocio toma esa petición y pide cargar el modelo ML a la capa de base de datos. Una vez lo tiene, usando un intérprete, le pasa al modelo la imagen introducida por el usuario en el formato correspondiente y mira los resultados. Luego toma esos resultados y los muestra a la capa de presentación.

6) Por desgracia, María sabe que la seta que ha introducido no es de la clase que ha indicado el modelo, por lo que decide corregirle. Hace click en el botón *That wasn't right…* y se dirige a la pantalla de corrección de setas, desde donde indica qué clase de seta es y confirma la selección.

7) Al confirmar la selección, la capa de presentación pasa la imagen y la nueva etiqueta a la capa de negocio. Aquí se codifica la imagen en base64 y se envía junto a la etiqueta indicada por el usuario al servidor usando una petición http tipo POST con la ayuda de las interfaces FileExchangeClient y RetrofitResponseListener.

8) El servidor está a la escucha. Desde la capa de presentación se recoge la petición post y se envía a la capa de negocio. Ahí se descodifica la imagen en base64, se le asigna un identificador único mediante un generador de identificadores únicos y, finalmente, envía a la capa de datos, esto es, se guarda la imagen en la carpeta correspondiente a su etiqueta. Mientras tanto, RetrofitResponseListener recibe la confirmación de si la operación ha sido un éxito o un fracaso y le comunica el resultado al usuario.

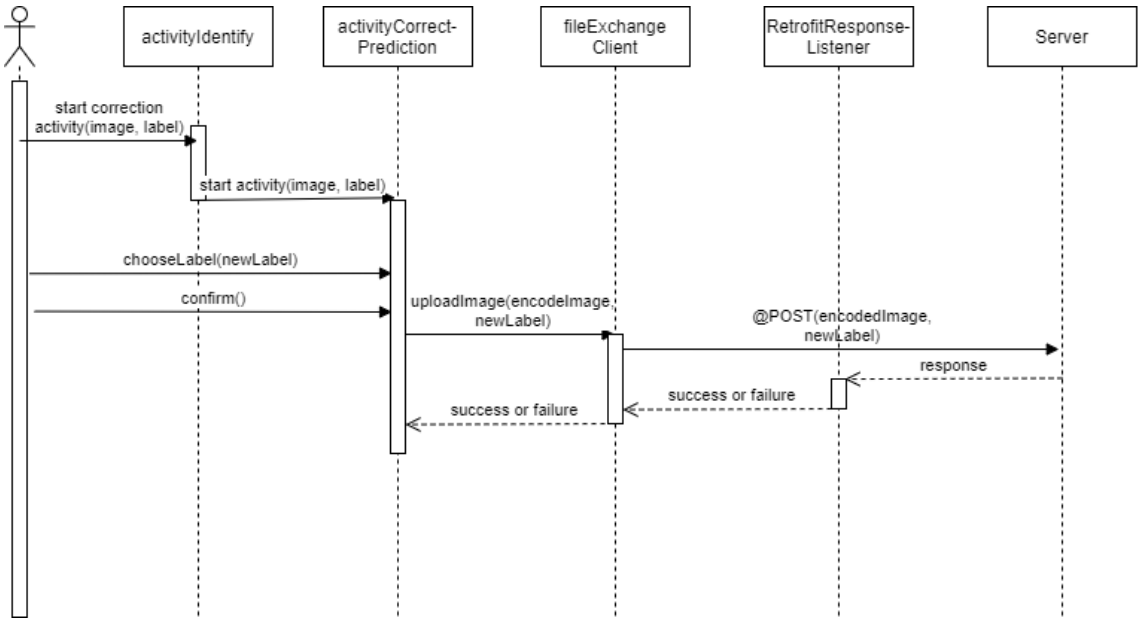

Ilustración 33: Diagrama de secuencia del envío de la imagen corregida con la nueva etiqueta

9. IMPLEMENTACIÓN

Como hemos ido viendo, la implementación de todo este sistema tiene tres partes, la parte de modelo ML, la parte de la aplicación y la parte del servidor. En este apartado explicaremos el "cómo", esto es, los aspectos técnicos del proceso de implementación: tecnologías, metodologías y enfoques utilizados. Además de esto dedicaremos un apartado para explicar en detalle cómo hemos desarrollado el componente ML.

9.1. CONCEPTOS GENERALES

Como ya hemos explicado anteriormente en el capítulo de metodologías, se sigue la metodología *Agile Scrum*. Esto significa que tomamos las funcionalidades que queremos implementar y las distribuimos a lo largo del tiempo disponible de desarrollo del proyecto en pequeños períodos de tres o cuatro semanas llamados *sprints*. Esta metodología permite al desarrollador tener una visión completa a corto y a largo plazo: por un lado le permite concentrarse en los objetivos inmediatos y, por el otro, le permite hacer un seguimiento general del progreso del proyecto.

De nuevo, también se usa gitflow, esto es, una metodología en la que, usando un repositorio (en nuestro caso, usamos Github con la facilidad Github Desktop), se desarrollan las funcionalidades en ramas apartadas y a medida que se van completando se van incorporando al proyecto.

9.2. IMPLEMENTACIÓN DEL MODELO

Como partimos sin conocimiento alguno acerca de machine learning, antes de iniciar la implementación llevamos a cabo las tareas de investigación necesarias para saber de qué va y cómo implementar el modelo adecuadamente. Entre las cosas que investigamos están los conceptos básicos de machine learning y las tecnologías necesarias. En este apartado mostramos los resultados de nuestra investigación y cómo han repercutido en el modo de construir el modelo.

9.2.1 Tecnologías

Para implementar el modelo se utilizan notebooks de Python, mediante el entorno GUI Anaconda Studio, en el fragmento Jupyter Notebooks. Para ello hemos creado un entorno virtual en el que incluimos las librerías de Keras y Tensorflow.

TensorFlow es una biblioteca de software gratuita y de código abierto que proporciona todas las herramientas necesarias para entrenar y validar modelos de machine learning.

Keras también es otra librería gratuita pero que actúa como interfaz de la librería de TensorFlow.

Aparte de todo esto utilizamos *pycodestyle* para garantizar la calidad del código y tensorflowLite para poder exportar el modelo a .tflite.

9.2.2 Desarrollo del modelo

Ahora en este apartado entraremos en detalle en el proceso de desarrollo del modelo.

9.2.2.1 Recogida de datos

Inicialmente el modelo está entrenado para poder identificar 10 clases de setas y, para cada una hay aproximadamente 100 imágenes. Estas 10 especies son:

1) Amanita Muscaria.

2) Boletus Pinophilus.

3) Calvatia Gigantea.

4) Galerina Marginata.

5) Hydnum Rufescens.

6) Lactarius Rufus.

7) Macrolepiota Procera.

8) Pholiota Squarrosa.

9) Strobilomyces Strobilaceus.

10) Tylopilus Felleus.

Aunque el número sea un tanto reducido al principio, a medida que se vaya desarrollando la aplicación se irán añadiendo más clases y más imágenes al set de datos. Todas estas imágenes son extraídas de una carpeta que separa las imágenes en subcarpetas nombradas con los nombres de las clases.

9.2.2.2 Separación entre set de validación y entrenamiento

Los datos se separan en un set de validación y en un set de entrenamiento. El set de entrenamiento se llevará el 80% de las imágenes, mientras que el set de validación se llevará el 20% de éstas. Las imágenes que caigan en cada uno de los sets son escogidas aleatoriamente y de forma equilibrada en cada una de las clases para evitar problemas de desequilibrio o parcialidad.

Por último también identificaremos las etiquetas en las que queremos clasificar las imágenes (de nuevo, por ahora son diez).

9.2.2.3 Capa de normalización

El siguiente paso consiste en aplicar una capa de normalización en la que reducimos la escala de los píxeles, inicialmente de 1 a 255, de 0 a 1, lo que facilita mucho los cálculos y añade eficiencia a la operación de entrenamiento.

9.2.2.4 Data augmentation

Como disponemos de pocos datos para entrenar un modelo, aplicamos una técnica en la que hinchamos artificialmente los datos del set de entrenamiento. La idea es que de cada imagen hacemos unos duplicados, les aplicamos algunas alteraciones y los reintroducimos como nuevas muestras dentro de nuestro dataset de entrenamiento. En nuestro caso las modificaciones que aplicamos consisten en rotaciones y vueltas aleatorias.

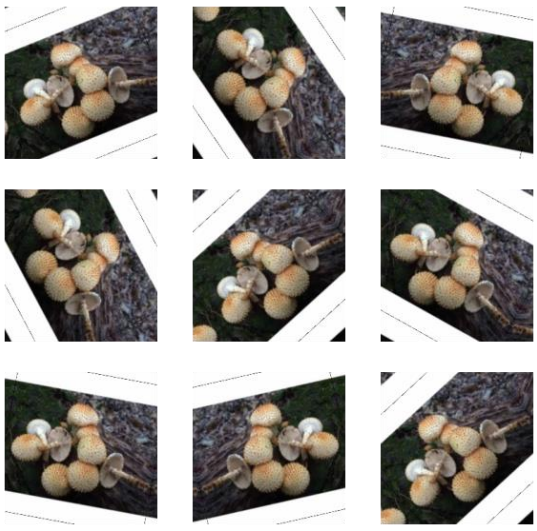

Ilustración 34: Resultado de data Augmentation: de cada imagen sacamos varios duplicados y aleatoriamente les aplicamos rotaciones y giros.

9.2.2.5 Definición de la arquitectura: Transfer learning

A la hora de desarrollar el modelo hemos de definir la arquitectura que va a seguir. Definir una arquitectura funcional es sencillo (de hecho, para realizar el proceso de aprendizaje realizamos unos cuantos modelos de prueba), pero definir la arquitectura óptima implica ser capaz de decidir la estructura exacta del modelo, esto es, el número de capas, las conexiones entre capas, el número de neuronas en cada una de ellas, etc. Esto requiere un proceso de experimentación demasiado largo y complejo para el tiempo del que disponemos, por lo que aplicaremos Transfer Learning.

Transfer Learning consiste en tomar el conocimiento de un modelo ya entrenado para resolver un problema y aplicarlo a otro problema. Así, por ejemplo, un modelo entrenado para ser capaz de reconocer camiones podría ser aplicado para reconocer coches.

En nuestro caso, teniendo en cuenta que nuestro objetivo es acabar desplegando este modelo en una aplicación Android, decidimos buscar una arquitectura rápida, ligera y eficiente para dispositivos móviles.

MobileNetsv2 **[17]** es una arquitectura CNN (*Convolutional Neural Network*) que cumple con estos requisitos.

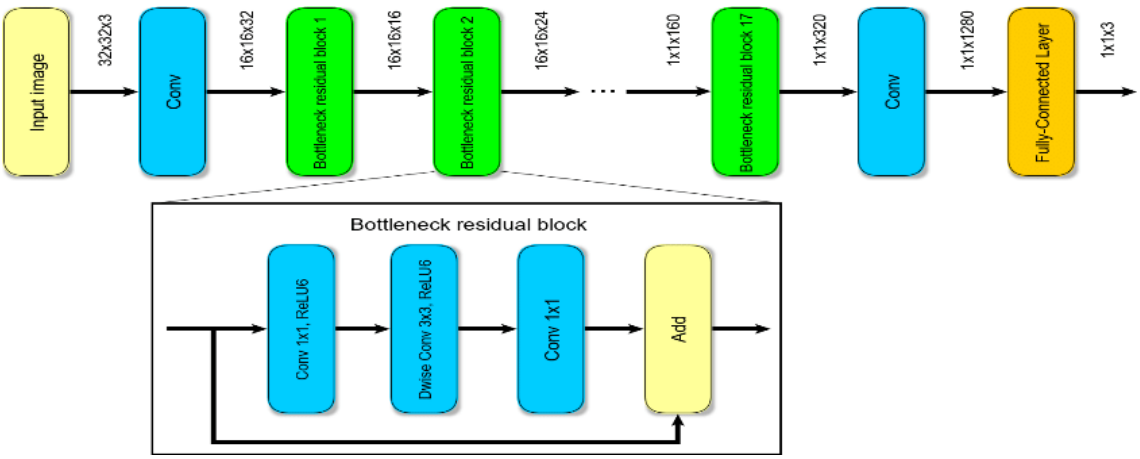

Ilustración 35: Estructura de MobileNetsv2

Como vemos en la ilustración se basa en *Bottleneck residual blocks* y cada uno de estos bloques está compuestos por tres capas. La primera es una capa 1x1 de convolución tipo Relu6, la segunda una *depthwise convolution* y la última otra 1x1 sin linealidad. La idea principal es que en cada primera capa se amplían las dimensiones de los canales de la

imagen, en la segunda se realiza la convolución independientemente por cada canal y en la tercera se vuelve a decrementar el tamaño de los canales: de este modo se consigue unos bloques de procesamiento menos intensos computacionalmente (gracias a que la convolución se realiza independientemente en cada canal) sin perder efectividad (gracias a los procesos de incremento y decremento de tamaño de canales).

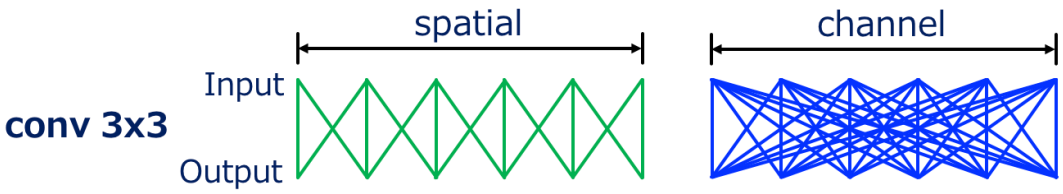

Ilustración 36: Ejemplo de convolución estándar. Fíjese como todos los canales son dependientes entre ellos, por lo que los costes computacionales para procesarlos son altos.

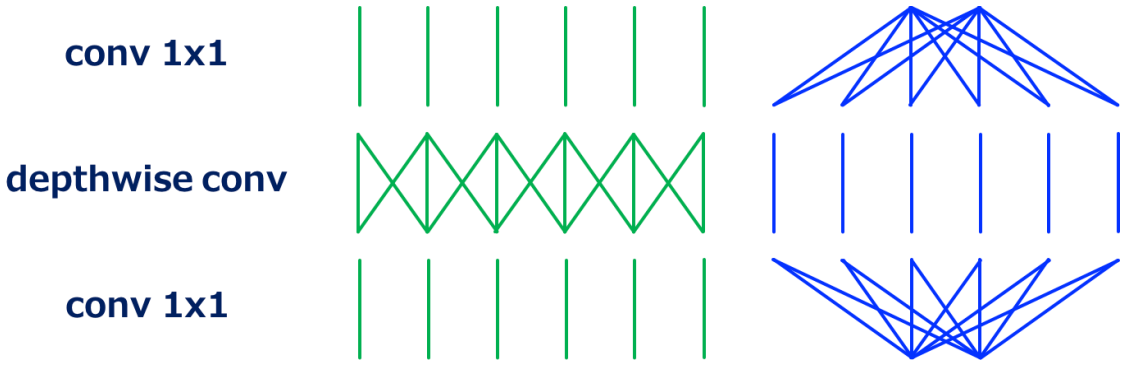

Ilustración 37: Ejemplo de convolución depthwise. Se realiza la operación de convolución independientemente en cada canal una vez han sido incrementados en tamaño. Luego se les decrementará de tamaño para continuar el proceso.

Para usar esta arquitectura nos la descargaremos de tensoflowHub (tfhub) y lo adaptamos para que las entradas acepten imágenes de tamaño 224x224 y con 3 canales. Luego configuraremos las salidas para que sean las correspondientes a las 10 clases.

9.2.2.6 Entrenamiento del modelo

Una vez el modelo está construido, toca entrenarlo. Para ello hay que decidir las métricas que nos interesa observar y evaluar. En nuestro caso nos decidimos por loss y *accuracy* en los grupos de entrenamiento y de validación.

Nos interesará observar los resultados tanto del entrenamiento como de la validación porque comparando ambos confiamos en que seremos más certeros a la hora de hacernos una idea sobre la precisión del modelo.

Por otro lado nos interesa observar además el *loss* porque sé que la *accuracy* puede ser, irónicamente, poco precisa a la hora de validar el modelo.

El motivo de por qué es así es como sigue: la *accuracy* se calcula de este modo dividiendo el número de predicciones correctas entre el número de predicciones realizadas en total.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Por otro lado, Loss se calcula de este modo: cuantificando la precisión del clasificador penalizando las clasificaciones erróneas.

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

Como podemos observar, aunque la *accuracy* me permita ver los aciertos sobre los fallos, es insuficiente para medir la gravedad de los fallos. Así, decir que un modelo diseñado para detectar pacientes con cáncer terminal con *accuracy* 90% es un buen modelo no es del todo correcto ya que el más mínimo error en este contexto es gravísimo y debería de tener más peso a la hora de juzgar un modelo como válido o no. Loss por otro lado permite estimar este "castigo por fallo".

También habrá que definir el número de *epochs* en los que se entrenará. Habrá que asegurarse que sea el número mínimo necesario para conseguir sacar los mejores resultados posibles. En nuestro caso en particular descubrimos que 20 *epochs* era un buen número ya que nos permitía alcanzar el 85% de *accuracy* y el <0.6 de *loss* que buscamos.

9.2.2.7 Exportar y guardar el modelo

Lo último será exportar el modelo. Existen muchos formatos pero nosotros usaremos tensorflow lite. Esta es una extensión para modelos que está diseñada de tal manera que optimiza los modelos para ser ejecutados y entrenados en dispositivos móviles, haciendo que sean modelos rápidos, ligeros, pequeños y eficientes en el consumo de la energía.

Este modelo será guardado en el servidor y será accesible por la aplicación móvil a través de las interfaces RESTFUL API ya descritas.

9.3. IMPLEMENTACIÓN DE LA APLICACIÓN

Android Studio es la herramienta principal sobre la que se desarrolla y se despliega la aplicación. Para la parte gráfica se utilizan actividades construidas en .xml y para la parte de cálculo se utiliza el lenguaje Java. Toda la aplicación se construye con Gradle.

Aparte de esto, también utiliza Okhttp3 y retrofit2, que son librerías para gestionar las peticiones http al servidor. También incluye el intérprete tensorflow lite, que se encargará de gestionar las peticiones al modelo tflite y mostrar sus resultados. El último componente es SQLite, que es un sistema de gestión de bases de datos interna a la aplicación que nos permitirá mantener permanencia de los datos de los usuarios.

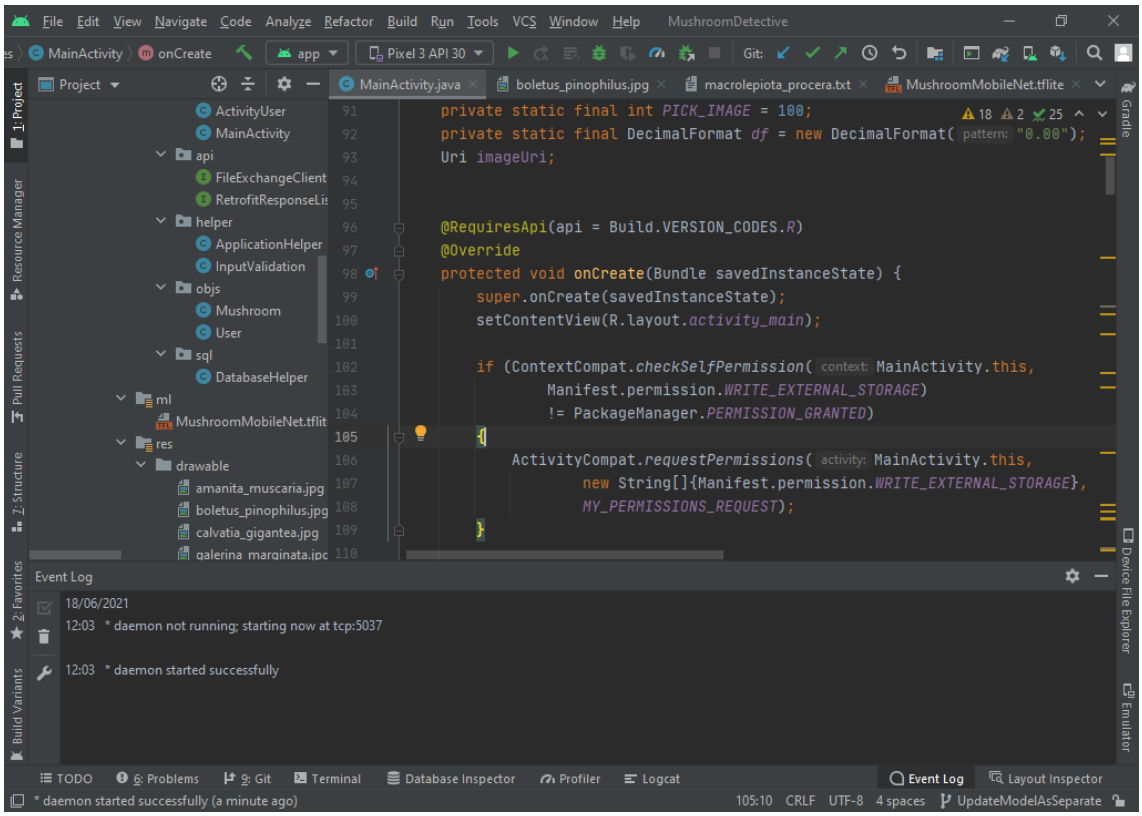

Ilustración 38: Entorno de Android Studio

9.4. IMPLEMENTACIÓN DEL SERVIDOR

Como el servidor también se encarga de entrenar, la mayoría de las tecnologías utilizadas para el modelo también se encuentran en la parte del servidor. En concreto se utilizarán Python Notebooks con las librerías Keras, TensorFlow y TensorFlowLite.

Para crear y gestionar el servidor utilizamos los servicios de DigitalOcean. Éstos nos permiten crear un *droplet*, que es un servidor que se puede utilizar en solitario o como

parte de una estructura más grande. Con tal de gestionar la lógica del servidor utilizaremos un script de Python que incluirá la librería Flask para escuchar y acoger las llamadas http.

Por último, para los reentrenamientos automáticos al final de semana se utiliza CronJobs para ejecutar el script de Python cada domingo.

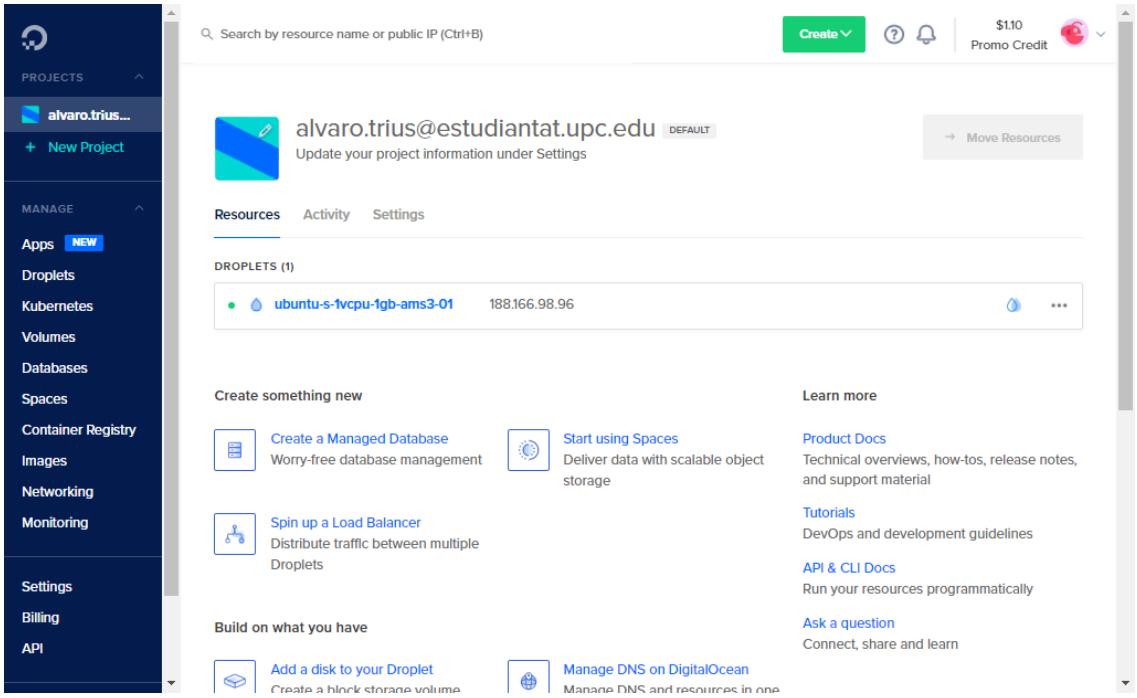

Ilustración 39: Interfaz de DigitalOcean

10. EJECUCIÓN FINAL DE LA PLANIFICACIÓN

Después del desarrollo del proyecto, ha llegado el momento de ver cuál ha sido la planificación final resultante: qué objetivos se han podido lograr y cuáles no y los costes esperados y los costes finales.

Para los objetivos haremos un informe detallado en el que sprint a sprint iremos detallando los objetivos iniciales propuestos y los objetivos completados. Para los costes mostraremos primero los costes estimados y luego los costes finales, justificando imprevistos.

10.1. DESARROLLO DE LOS SPRINTS

10.1.1 Sprint 1

Objetivos iniciales propuestos:

-Adquisición de conocimientos básicos de Machine Learning.

-Realización de una prueba de concepto.

-Especificación de los requisitos.

-Preparación del entorno.

-Diseño de la arquitectura.

Objetivos logrados:

-Adquisición de conocimientos básicos de Machine Learning: El objetivo fue alcanzado satisfactoriamente: pude investigar en diversas fuentes y adquirí los conocimientos necesarios para iniciarme en la implementación de una aplicación basada en machine learning.

-Realización de una prueba de concepto: El objetivo fue alcanzado satisfactoriamente. No solo se realizaron pruebas de concepto para solidificar los conocimientos adquiridos, sino que además se adelantó en el diseño e implementación de un web Server que era capaz de realizar predicciones basadas en imágenes de setas.

-Especificación de los requisitos: El objetivo fue alcanzado satisfactoriamente: se lograron identificar los requisitos de este proyecto.

-Preparación del entorno: El objetivo fue alcanzado con cierto éxito: aunque no fue posible prever todos y cada uno de los detalles del proyecto que fueran a necesitar de ciertas características del entorno, al menos se logró prepararlo lo suficiente como para empezar.

-Diseño de la arquitectura: El objetivo fue alcanzado con cierto éxito: se logró diseñar un primer esbozo de la arquitectura.

Cambios para la siguiente iteración:

La adquisición de nuevos conocimientos me hizo darme cuenta que algunos objetivos de la iteración 2 estaban mal concebidos: por eso se decidió que la iteración 2 sería dedicada a la elaboración de un MVP (*Minimal Viable Product*), esto es, una aplicación móvil que recogiera la funcionalidad básica de saber identificar setas. Por estos motivos, el objetivo de "Implementar una base de datos" fue sustituido por "Implementar un modelo en ML" y la implementación de la IA por "Integrar el modelo ML en una aplicación móvil". Esto a su vez implicaría que el objetivo de la iteración 3 "Inserción y reconocimiento de imágenes" sería llevado a la iteración 2, ya que descubrí que al integrar el modelo ML prácticamente ya estaba cumpliendo este objetivo.

10.1.2 Sprint 2

Objetivos iniciales propuestos:

-Implementar un modelo en ML.

-Integrar el modelo ML en una aplicación móvil.

Objetivos logrados:

-Implementar un modelo en ML: El objetivo fue alcanzado con éxito. Al principio tuve que diseñar una colección de imágenes suficientemente grande a partir de la cual entrenar el modelo y, después, fue necesario diseñar el modelo. El diseño del modelo tardó más de lo esperado porque, después de investigar, descubrí que sería más interesante diseñar el modelo utilizando Transfer Learning, una técnica que aumentó mucho la precisión del modelo.

-Integrar el modelo ML en una aplicación móvil: El objetivo fue alcanzado con cierto éxito. Se logró integrar el modelo en una aplicación de móvil básica, pero por desgracia, un bug impedía que diera los resultados correctos.

Cambios para la siguiente iteración:

La finalización de la integración del modelo ML en una aplicación móvil fue trasladada al siguiente sprint debido al bug encontrado.

10.1.3 Sprint 3

Objetivos iniciales propuestos:

-Arreglar el bug del modelo mal integrado en la aplicación.

-Corrección de imagen.

-Diseño de la interfaz.

Objetivos logrados:

-Arreglar el bug del modelo mal integrado en la aplicación: se consiguió resolver sin mayor dificultad. Por lo visto, al trasladar la el modelo al móvil no se hacía correctamente la normalización de la imagen, por lo que los resultados eran absolutamente erróneos.

-Corrección de imagen: resultó ser mucho más complicado de lo esperado. Este objetivo acabó dividiéndose en tres más: “obtener modelo actualizado del servidor”, “programar el reentrenamiento automático en el servidor” y “enviar corrección de imagen al servidor”. Debido a que era época de exámenes, sólo se logró conseguir la obtención del modelo actualizado del servidor.

-Diseño de la interfaz: se logró sin mayor dificultad.

Cambios para la siguiente iteración:

De modo análogo al sprint anterior, nos vimos obligados a trasladar algunos de los objetivos del sprint anterior al siguiente, en concreto, “programar el reentrenamiento automático en el servidor” y “enviar corrección de imagen al servidor”.

Además de esto se descubrió que, debido a especificaciones no funcionales, se hacía necesario implementar una nueva funcionalidad: el “aviso para la seguridad del usuario”.

Esta nueva funcionalidad consiste en que el usuario debería tener a mano algún informe sobre advertencias y buenas prácticas de seguridad para la recogida de setas y el uso de la aplicación.

10.1.4 Sprint 4

Objetivos iniciales propuestos:

-“Programar el reentrenamiento automático en el servidor”

-“Enviar corrección de imagen al servidor”.

-“Diseño e implementación de la interfaz”.

-“Gestión de usuario”

-“Gestión de enciclopedia”.

-“Aviso de seguridad”.

-“Logros”

Objetivos logrados:

-“Programar el reentrenamiento automático en el servidor”: Mediante el uso de CronJobs, se logró completar este objetivo.

-“Enviar corrección de imagen al servidor”: Se programó el envío del POST al servidor y su recepción con éxito.

-“Diseño e implementación de la interfaz”: Se logró finalmente decidir e implementar la interfaz de la aplicación.

-“Gestión de usuario”: Se logró implementar una base de datos SQLite con todas las funcionalidades e interfaces necesarios.

-“Gestión de enciclopedia”: Se logró implementar sin dificultad.

-“Aviso de seguridad”: Se logró implementar sin dificultad.

10.2. COSTES FINALES

En el informe de costes del GEP decidimos realizar la planificación de los costes asumiendo que la aplicación iba a ser realizada por un equipo de profesionales pero, de cara este informe final, estimamos necesario tratar este apartado desde el punto de vista real.

10.2.1 Costes previstos

Si recordamos, los costes de desarrollo de esta aplicación iban a ser muy pequeños, en concreto, sólo el coste de amortización y el energético de usar el portátil durante los meses de desarrollo durante cinco horas al día:

Modelo	Precio (€)	Dedicación diaria (horas)	Días laborales (/año)	Duración proyecto (horas)	Vida útil (años)	Amortización (€)
ASUSPRO Essential	1050	5	220	540	4	128,86

Consumo por hora (Kwh)	Precio (€/Kwh)	Precio total de consumo (€)
1,1	0,14	83,16

Tabla 14: Costes estimados reales

10.2.2 Costes reales

No ha habido variaciones reales entre el coste real y el coste esperado. El único coste inesperado que hubo fue el del mantenimiento del servidor DigitalOcean, servidor que usamos como host para atender a nuestras peticiones on-line.

Coste al mes (€)	Coste total (3 meses)
4.09 (alquiler) + 1.00 (mantenimiento de los *backups*)	15,27

Tabla 15: Costes extra

Lo que pasa es que al crear un servidor en DigitalOcean se te regala un crédito de 100 dólares (81,84 € en el momento que se hizo conversión de moneda) para los primeros meses, por lo que este gasto, al menos de cara al desarrollo del proyecto, no se aplica.

11. VALIDACIÓN

En este apartado realizaremos la validación, esto es, los requisitos que hemos logrado alcanzar y los que no.

Con tal de realizar la validación, hemos empleado pruebas de integración incremental. Esto es, cada vez que desarrollábamos una nueva funcionalidad de la aplicación lo hacíamos en el conjunto de todas las funcionalidades desarrolladas hasta el momento. Una vez que hemos considerado que una funcionalidad estaba implementada y testeada, nos hemos asegurado de volver a testear el resto de funcionalidades. En total se han hecho cuatro: una para la identificación de setas, otra para la comunicación con el servidor, una tercera para la gestión de contenidos tipo enciclopedia y aviso de seguridad y una última para la gestión de usuarios y logros.

11.1. REQUISITOS FUNCIONALES

-Identificación de setas: La aplicación es capaz de tomar una imagen y estimar las posibilidades de que pertenezca a un conjunto de clases de seta conocido.

-Corrección de identificación de setas: La aplicación permite al usuario realizar una corrección a la identificación en caso de que sospeche que se ha equivocado. Dicha corrección es enviada al servidor.

-Mejora en la identificación de setas: El sistema de servidor es capaz de tomar las correcciones de los usuarios e incorporarlos a su dataset.

-Consulta de enciclopedia de setas: La aplicación ofrece al usuario una pequeña enciclopedia en la que puede consultar los datos referidos a una clase de setas.

-Consulta de datos de setas: La aplicación permite al usuario que, una vez haya identificado la seta, consultar los datos de la misma inmediatamente dirigiéndolo a la entrada de la enciclopedia correspondiente.

-Gestión de usuario: La aplicación permite al usuario crear un perfil, iniciar sesión con él, consultarlo, modificarlo y borrarlo.

-Gestión de logros: La aplicación es capaz de seguir la actividad del usuario y otorgarle recompensas virtuales.

11.2. REQUISITOS NO FUNCIONALES

11.2.1 Requisitos no funcionales de la aplicación software

Requisito	Resultado	Validación
Eficiencia	Después de 50 identificaciones, la media era de 1.5 segundos. Después de 10 reentrenos, la media era de 18 minutos.	Validado.
Usabilidad	Después de distribuir la aplicación a 5 usuarios, el resultado obtenido en la encuesta de media ha sido 3.8. Aunque han indicado que han echado de menos un apartado que les dé instrucciones claras sobre cómo usarla.	Validado.
Disponibilidad	La aplicación es del todo funcional sin conexión a internet salvo por la corrección de setas y la actualización de setas.	Validado.
Salud y seguridad	Hemos incluido un aviso en la aplicación (funcionalidad nueva del Sprint 4) sobre lo arriesgado que es confiar en los resultados de la aplicación.	Validado.
Apariencia	Se ha recibido una nota de 3.6. Entre los comentarios recibidos el más común ha sido que, aunque se agradece la simpleza del diseño, habría sido bueno añadir “algo más interesante” que observar.	Validado.
Aprendizaje	Se ha recibido una nota de 4,1. Similarmente a la usabilidad, el comentario común ha sido que se ha echado de menos la inclusión de un manual de instrucciones.	Validado.
Fiabilidad	El modelo se ha logrado entrenar con una media de *accuracy* de 85% y de *loss* menor a 0.6.	Validado.

Tabla 15: Justificación de requisitos no funcionales parte aplicación

11.2.2 Requisitos no funcionales específicos del modelo machine learning

Requisito	**Resultado**	**Validación**
Modelo apropiado	Se ha utilizado transfer learning para utilizar una arquitectura eficiente para móviles. Además se ha usado un modelo que emplea imágenes de los tamaños especificados.	Validado.
Goodnes of fit	En los 10 entrenos del modelo se ha mantenido una *accuracy* del 85% de media y con un *loss* inferior a 0.6.	Validado.
Datos de entrada completos y equilibrados	Los datos de entrada han consistido en 10 clases de setas con 100 imágenes cada uno. La clase con más imágenes tienen 102, las que tienen menos tienen 100.	Validado.
Relevancia	En cada uno de los 10 entrenos se escoge aleatoriamente las entradas de los sets de validación y entreno. La *accuracy* y el *loss* se han mantenido igual.	Validado.
Eficiencia del entrenamiento y de la ejecución	De modo análogo a la eficiencia buscada en la aplicación, el modelo de por sí se entrena en una media de 18 minutos y es capaz de identificar setas en menos de un segundo.	Validado.
Calidad de código	En el script de entreno del modelo se ha usado *pycodestyle* y se ha construido un código de calidad (a pesar de las contradicciones de la herramienta). Cada celda está comentada con una explicación de lo que se hace.	Validado.
Medición continua del rendimiento y eficiencia	Durante el desarrollo del proyecto ni la *accuracy* ni el *loss* han bajado de los valores esperados.	Validado.

Tabla 16: Justificación de requisitos no funcionales de parte de modelo ML

12. JUSTIFICACIÓN DE COMPETENCIAS

En este apartado realizaremos la justificación de competencias, esto es, cómo hemos desarrollado cada una de las competencias propuestas a la hora de proponer el proyecto al principio del cuatrimestre.

CES1.1: Desarrollar mantener y evaluar sistemas y servicios software complejos y/o críticos. [En profundidad]

Esta competencia ha sido desarrollada al tener que construir un sistema complejo de redes neuronales que incorporaba elementos de comunicación cliente-servidor. Ha sido necesario mantener y evaluar cada uno de estos componentes y el modo con el que se comunicaba.

CES1.5: Especificar, diseñar, implementar y evaluar bases de datos. [Bastante]

Esta competencia ha sido desarrollada al tener que implementar una base de datos para mantener los datos de los usuarios.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. [En profundidad]

Durante el proceso de desarrollo del proyecto, ha sido necesario establecer metodologías y aplicar herramientas para garantizar la calidad general del desarrollo del software. Para la construcción del modelo de redes neuronales ha sido necesario establecer pruebas y métricas para controlar la calidad del software. A pesar de todo, no se ha desarrollado con la profundidad deseada.

CES1.8: Especificar, diseñar e implementar sistemas de control y de tiempo real. [Bastante]

A la hora de diseñar y desarrollar el modelo ha sido necesario establecer sistemas para controlar la eficiencia del modelo. A la hora de gestionar la comunicación entre cliente y servidor ha sido necesario establecer sistemas de tiempo real para garantizar la calidad de la conexión. Por último, el componente del servidor ha sido automatizado para realizar tareas a lo largo del tiempo

CES2.1: Definir y gestionar los requisitos de un sistema software. [Bastante]

Con tal de poder empezar a desarrollar este proyecto ha sido indispensable definir los requisitos funcionales y no funcionales del sistema software.

CES2.2: Diseñar soluciones apropiadas en uno o más dominios de aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos. [Un poco]

Una de las principales inspiraciones para desarrollar el proyecto ha sido social y ética: crear una aplicación para ayudar a los aficionados a la recogida de setas a poder identificarlas correctamente y no correr riesgo de ser envenenado.

13. INTEGRACIÓN DE CONOCIMIENTOS

13.1. INTEGRACIÓN DE CONOCIMIENTOS DE GESTIÓN DE PROYECTOS DEL SOFTWARE

Con tal de poder planificar el proyecto adecuadamente ha sido necesario aplicar conocimientos de gestión de proyectos, adquiridos de las asignaturas de Empresa y Entorno Económico (EEE), Ingeniería de Requisitos (ER), Gestión de Proyectos de Software (GPS) y Proyecto de Ingeniería del Software (PES). En particular:

-Para identificar una necesidad en el mercado y analizar su viabilidad.

-Para analizar los riesgos y oportunidades del producto.

-Para determinar la metodología a seguir y las herramientas necesarias.

-Para identificar correctamente los *stakeholders*.

-Para estimar los costes de las tareas en función de dinero y de tiempo.

-Para llevar a cabo las prácticas necesarias para hacer seguimiento del proyecto.

13.2. INTEGRACIÓN DE DISEÑO DEL SOFTWARE

Con tal de poder diseñar el proyecto ha sido necesario aplicar conocimientos de diseño del software, adquiridos de las asignaturas Gestión de Proyectos del Software (GPS), Ingeniería de Requisitos (ER), Introducción a la Ingeniería del software (IES), Proyecto de Ingeniería del Software (GPS) y Arquitectura del Software (AS). En particular:

-Para identificar correctamente los requisitos funcionales y no funcionales.

-Para la identificación de los casos de uso y el diseño de la arquitectura física y lógica.

13.3. INTEGRACIÓN DE CONOCIMIENTOS DE IMPLEMENTACIÓN

Con tal de poder diseñar el proyecto ha sido necesario aplicar conocimientos de implementación de software, adquiridos de las asignaturas Programación (P1 y P2), Estructuras de Datos y Algoritmos (EDA), Algoritmia (A), Proyectos de Programación (PROP), Proyecto de Ingeniería del Software (GPS), Aplicaciones y Servicios Web (ASW), Minería de Datos (MD), Bases de Datos (BD) y Videojuegos (VJ). En particular:

-Se han aplicado conocimientos de redes y servidores para implementar un servidor capaz de atender a peticiones API-Rest de la aplicación.

-Se han aplicado conocimientos de implementación del software generales para poder implementar la aplicación.

-Se han aplicado conocimientos de prácticas de buena programación para guiarnos a la hora de realizar la implementación.

-Se han aplicado conocimientos de minería de datos para construir, entender y validar los modelos de Machine Learning usados durante el desarrollo de este proyecto.

-Se han aplicado conocimientos de Bases de Datos para gestionar un modelo SQL que contiene tablas con la información de usuarios.

13.4. INTEGRACIÓN DE OTROS CONOCIMIENTOS

Aparte de los conocimientos adquiridos de las materias de la carrera, también se han aplicado conocimientos desarrollados como parte del proceso de aprendizaje. En concreto para realizar al cabo tareas de investigación para aprender acerca de nuevos temas y aplicarlos al desarrollo del proyecto.

14. INFORME DE SOSTENIBILIDAD

En el siguiente apartado presentamos el informe de sostenibilidad, en el que realizamos el cálculo de la sostenibilidad del proyecto.

Por desgracia, nos damos cuenta de que este cálculo no es tan fácil como parece, a fin de cuentas, no hay indicadores numéricos tan claros como los que tiene el caso del cálculo de presupuestos. Es por eso que el método que haremos no será matemático, sino que nos limitaremos a plantear las posibles implicaciones ambientales, sociales y económicas de nuestro proyecto y veremos cómo podemos minimizar el impacto negativo que pueda tener.

Este informe está dividido en dos partes: en la primera nos planteamos las preguntas del hito inicial planteadas en el GEP y en la segunda las del hito final. Finalmente presentaremos la autoevaluación en la que haremos una revisión general de la evolución de nuestra perspectiva sobre la sostenibilidad durante nuestro paso en la UPC y el desarrollo del proyecto.

14.1. HITO INICIAL

14.1.1 Dimensión ambiental

¿Se ha estimado el impacto ambiental que tendrá la realización del proyecto?

Teniendo en cuenta que el proyecto es de software, el impacto ambiental que tendrá va a ser mínimo. A pesar de todo, es cierto que va a haber un continuo consumo de recursos para desarrollar la aplicación, principalmente el consumo eléctrico. También es cierto que el desarrollo del proyecto podría implicar realizar actividades de recogida y reconocimiento de setas pero, como serán llevadas a cabo por una sola persona, su impacto medioambiental va a ser mínimo.

¿Han sido planteados modos de minimizar el impacto, por ejemplo, reutilizando recursos?

Sí, por un lado, con tal de minimizar el consumo eléctrico, se procurará trabajar llevando prácticas de ahorro como ajustar el brillo del monitor, apagar el ordenador si se va a estar sin usar durante dos horas, utilizar el modo de ahorro de energía y procurar adquirir software de manos de empresas comprometidas con el medio ambiente.

Por el otro, para la recopilación de fotografías de setas y hongos, intentaremos limitarnos a las de bases de datos ya existentes. A pesar de todo, si el trabajo de campo fuera necesario, nos aseguraremos de seguir todas las indicaciones de los expertos para no dañar el medio ambiente durante la expedición.

¿Cómo se resuelve actualmente el problema que se quiere abordar desde la perspectiva ambiental?

Actualmente existen aplicaciones alternativas, pero la solución más común es la consulta con expertos en materia y la lectura de sus libros y manuales.

Desde la perspectiva ambiental, aunque cada caso es único, las aplicaciones pueden estar diseñadas para reducir el gasto de energía y los libros y manuales pueden ser elaborados con papel reciclado.

¿En qué mejorará ambientalmente la solución a las existentes?

No mejorará mucho respecto a las aplicaciones existentes, pero al menos comportará un ahorro en el consumo de papel así como los recursos para la impresión. La consulta con expertos, a pesar de todo, seguirá siendo necesaria (a fin de cuentas, ya partimos de que nuestra aplicación no será perfecta para identificar setas y animaremos a los usuarios a consultarles), por lo que no podremos recortar los costes ambientales de lo que esto implique.

14.1.2 Dimensión económica

¿Ha sido estimado el coste de la realización del proyecto (recursos humanos y materiales)?

Sí, en el apartado anterior se ha elaborado un presupuesto exhaustivo de todos los recursos.

¿Cómo se resuelve actualmente el problema que se quiere abordar desde la perspectiva económica?

Tanto el mantenimiento y desarrollo de aplicaciones como la escritura de libros y manuales genera puestos de trabajo e incentiva el comercio de setas y hongos, poniendo así su granito de arena en el engranaje de la economía. Por otro lado, también estas

mismas aplicaciones y manuales suelen ser gratis o asequibles económicamente, lo que permite su fácil comercialización.

¿En qué mejorará económicamente la solución a las existentes?

De partida, la aplicación será gratis lo que resulta ser una ventaja económica muy grande. Aunque algunas de las otras aplicaciones son gratis, otras no lo son. Por otro lado, los manuales de setas suelen tener cierto coste, por lo que nosotros colaboraremos a hacer esta afición algo más asequible.

14.1.3 Dimensión social

¿Qué va a aportar a nivel personal la realización del proyecto?

Principalmente, el autor adquirirá conocimientos muy valiosos para el desarrollo de aplicaciones de software para móvil, muy en boga en el mercado actual. También permitirá al autor aprender machine learning, una de las nuevas tecnologías más prometedoras. Por último, también adquirirá conocimientos entretenidos acerca del mundo de las setas y los hongos.

¿Cómo se resuelve actualmente el problema que se quiere abordar desde una perspectiva social?

Las aplicaciones y manuales hacen a sus usuarios más conscientes de su entorno y les educan en el correcto proceder para practicar esta afición, además de incentivar la cooperación entre todos ellos.

¿En qué mejorará socialmente la solución a las existentes?

Teniendo en cuenta que ya vimos que las aplicaciones existentes y manuales tienen sus inconvenientes, esta aplicación intentará ser una alternativa usable, eficaz y asequible. De este modo, mi aplicación alcanzará los objetivos de educar y de aumentar la conciencia respecto al entorno de forma más eficiente.

¿Existe una necesidad real del proyecto?

Teniendo en cuenta el número de casos de toxicidad en España y el hecho de que la actividad de recogida de setas es aún muy practicada, en opinión del autor sí que hay una necesidad real.

Por otro lado, también hay que admitir que es una necesidad muy centrada en un sector particular de la población, por lo que su impacto social va a ser menor.

14.2. HITO FINAL

14.2.1 Dimensión ambiental

¿Has cuantificado el impacto ambiental de la realización del proyecto? ¿Qué medida has tomado para reducir el impacto? ¿Has cuantificado esta reducción?

Sí, lo hemos estimado a partir de costes energéticos, en kWh por hora. Hemos estimado que el PC en el que trabajamos consume 0,88 kWh (por ser un portátil) y el servidor de DigitalOcean 1,76 kWh. Teniendo en cuenta que le dedicamos 540 horas al proyecto eso da un consumo total de 1425,6 kW.

Con tal de reducir el impacto nos hemos asegurado de activar el servidor únicamente en las horas que dedicábamos a desarrollar las tareas relacionadas con él, que al final han resultado ser 120 horas, por lo que el coste ha bajado de los 1425,6kW estimados a 686,4 kW.

Si hicieras un nuevo proyecto, ¿podrías realizarlo con menos recursos?

No. El proyecto ya de por sí requiere muy pocos recursos, por lo que no creo que fuera posible.

¿Qué recursos estimas que se usarán durante la vida útil del proyecto? ¿Cuál será el impacto ambiental de estos recursos?

Teniendo en cuenta que el proyecto es sobre todo de software, los recursos utilizados van a ser sobre todo energéticos, por lo que va a colaborar a que haya una mayor demanda de energética.

Por otro lado también es un proyecto para móviles, por lo que también colaborará en incentivar la producción de estos dispositivos, con lo que el gasto de CO2 y el coste de extracción de materiales preciosos comportan como el coltán.

¿El proyecto permitirá reducir el uso de otros recursos? ¿Globalmente, el uso del proyecto mejorará o empeorará la huella ecológica?

Por un lado, en cuanto al consumo de recursos, posiblemente el único recurso que permitirá reducir es el del consumo de papel y gastos de impresión (porque, de nuevo, recordemos que mi proyecto tenía intención de ser un reemplazo a los manuales de búsqueda de setas comunes) a cambio de un aumento de consumo energético y de materiales preciosos para la producción de móviles. Teniendo en cuenta el gasto continuo que lleva el mantenimiento de una aplicación, creemos que el impacto será mayor. Además, es cierto que nuestro proyecto incentivará la búsqueda de setas, por lo que podría tener un impacto pesado sobre el reino fungi.

Aún y así, creemos que estos gastos son tan pequeños que, globalmente, no creemos que haya variación significativa.

¿Podrían producirse escenarios que hiciesen aumentar la huella ecológica del proyecto?

Si el proyecto se vuelve muy popular, esto claramente incrementaría los gastos energéticos globales, por lo que aumentaría la huella ecológica. También podría darse el caso que demasiada gente se volviera aficionada a la búsqueda de setas y hongos, lo que sería destructivo para el reino fungi.

14.2.2 Dimensión económica

¿Has cuantificado el coste (recursos humanos y materiales) de la realización del proyecto? ¿Qué decisiones has tomado para reducir el coste? ¿Has cuantificado el ahorro?

Han sido cuantificados en el apartado de costes en las secciones anteriores. Teniendo en cuenta que el proyecto ha sido reducido por una persona, no ha sido posible reducir el coste humano. Con respecto a la reducción de coste económico, la única medida que se ha tomado ha sido asegurarse que no haya ningún coste económico (salvo los energéticos), por lo que nos hemos asegurado de que todos los recursos que hemos usado hayan sido gratis.

¿Se ha ajustado el coste previsto al coste final? ¿Has justificado las diferencias (lecciones aprendidas)?

Sí y no. Es cierto que no esperábamos el coste del servidor, pero por otro lado, como ya comentamos en apartados anteriores, el servidor al final nos ha salido gratis gracias al crédito de regalo, por lo que, en la práctica, no ha sido necesario gastar nada.

¿Qué coste estimas que tendrá el proyecto durante su vida útil? ¿Se podría reducir este coste para hacerlo más viable?

Si el proyecto se convierte en una aplicación comercial, será necesario adquirir un equipo que se encargue de mantenerla y actualizarla, implicando sueldos, costes de equipos y Works pace.

1- Costes de equipo (extraídos del apartado 5.1.1 Costes de actividad) en un mes.

Sueldo total de equipo/hora	Horas de trabajo mensuales	Coste total al mes (€)
117,58	171,4	20153,21

2- Costes de amortización de equipos informáticos:

$$\frac{Coste\ (euros)}{Vida\ útil\ (años) * Días\ laborables\ al\ año * Dedicación\ diaria\ (horas)} * Duración\ del\ proyecto\ (horas)$$

Modelo	Precio (€)	Dedicación diaria (horas)	Días laborales (/año)	Trabajo en un mes (horas)	Vida útil (años)	Amortización (€)
ASUSPRO Essential	1050	8	220	171,4	4	25,86

Número de equipos informáticos	Amortización en un mes total (€)
6	153,38

3- Woks pace en un mes (de nuevo, contratamos los servicios de Zamness **[15]**).

Precio por mes (€)	125

Por lo que el coste total de mantener la aplicación en un mes es de: 20431,59€

El modo más claro de reducir los costes del proyecto es, sin duda alguna, reducir el equipo. Es cierto que necesitaremos cada miembro para realizar su tarea a lo largo de la vida útil, pero no lo necesitaremos siempre. Así, por cada segmento en la vida del proyecto, se podría determinar qué miembro del equipo es necesario contratar y cuál no.

¿Se ha tenido en cuenta el coste de los ajustes, actualizaciones, reparaciones durante la vida útil del proyecto?

Sí. Resulta imposible predecir qué ajustes, actualizaciones o reparaciones serían necesarias durante la vida útil del proyecto exactamente, pero al menos, volviendo a la pregunta anterior, sabemos que estos reajustes implicarán el trabajo de alguna persona, por lo que podemos estimar estos costes imprevistos a partir del coste de contratar a un responsable.

Así, por ejemplo, una actualización en el diseño gráfico de la aplicación se puede estimar fácilmente a partir de la amortización de un equipo informático nuevo y el sueldo de un nuevo miembro al equipo: el diseñador gráfico.

¿Podrían producirse escenarios que perjudicasen la viabilidad del proyecto?

Sí sin duda alguna. Este proyecto sólo puede sobrevivir en la medida que sea interesante comercialmente por lo que si no hay suficiente interés en la población o la competencia se vuelve mucho más atractiva, el proyecto se puede volver inviable.

14.2.3 Dimensión social

¿La realización de este proyecto ha implicado reflexiones significativas a nivel personal, profesional o ético de las personas que han intervenido?

Sí. Siendo yo el único desarrollador del proyecto, me he planteado en ocasiones acerca de la moralidad de realizar una aplicación que ayude a la gente a identificar setas para separar las beneficiosas de las venenosas. Me he dado cuenta de que si mi aplicación no es lo suficientemente precisa para saber distinguir bien entre hongos, parte de la

responsabilidad ética de que suceda una desgracia caerá sobre mis hombros. Esta reflexión es la que me ha llevado que me decidiera añadir una funcionalidad extra al proyecto en la que informaba a los usuarios de los riesgos de usar mi aplicación.

¿Quién se beneficiará del uso del proyecto? ¿Hay algún colectivo que puede verse perjudicado por el proyecto? ¿En qué medida?

El colectivo de aficionados a la búsqueda de setas serán los principales beneficiados, pero también pueden verse perjudicados si mi aplicación no es clara a la hora de declarar los riesgos que tiene fiarse de ella a toda costa.

¿En qué medida soluciona el proyecto el problema planteado inicialmente?

En la medida que se ha convertido en una herramienta más para ayudar a los aficionados a la búsqueda de setas a ser capaces de identificarlas.

¿Podrían producirse escenarios que hiciesen que el proyecto fuese perjudicial para algún segmento particular de la población?

Sí. Teniendo en cuenta que el entreno del modelo Deep Learning se basa sobre todo en contribuciones de usuarios, podría resultar que el modelo se vuelva totalmente descompensado e inviable, lo que resultaría una desgracia. Por ello sería necesario que, durante la vida del proyecto, asegurarse que el modelo evoluciona adecuadamente.

¿Podría crear el proyecto algún tipo de dependencia que dejase a los usuarios en posición de debilidad?

Tal vez. Hemos incluido técnicas de gamificación para incentivar el uso de la aplicación. Estas técnicas son bastante inocuas e inofensivas por el momento, pero la aplicación podría desarrollarse demasiado en esta dirección y convertirse en algo que podría generar adicción.

14.3. AUTOEVALUACIÓN

Durante los años que he estado en la universidad, la temática de la sostenibilidad ha sido una constante presente en muchas de nuestras asignaturas, por lo que he podido aprender mucho acerca de ella. También ha sido muy presente en las diversas actividades extracurriculares organizadas y/o permitidas por la UPC que, al cubrir todas las dimensiones, me han ayudado a tomar conciencia con los aspectos sociales, económicos y ambientales al desarrollar un proyecto profesional.

La dimensión que he visto aparecer con más frecuencia ha sido la medioambiental. En una asignatura se daban contenidos sobre la sostenibilidad ambiental que tenían tanta importancia como el resto de contenidos, llegando hasta tal punto de ser materia importante de examen. En otra asignatura tuvimos que desarrollar un proyecto de software partiendo del hecho de que debía ser sostenible medioambientalmente, obligándonos a tomar una perspectiva completamente distinta. Por último, a modo de actividad extra, para otra asignatura asistimos a la proyección de un documental sobre la extracción de materia prima para componentes electrónicas y sus repercusiones sociales y medioambientales, en la que tuvimos la ocasión de hablar con el realizador.

La segunda dimensión de la que he podido aprender mucho es la económica. Por un lado, una asignatura estaba plenamente centrada en el cálculo de costes y en la gestión empresarial y me ha servido como base para poder entender mejor esta dimensión. Por otro lado en otra asignatura nos vimos obligados a aplicar esos conocimientos para evaluar los costes de construir una empresa ficticia.

La dimensión social también ha sido bien desarrollada. Así, era una pregunta frecuenta durante el desarrollo de proyectos plantearse las repercusiones sociales y éticas de nuestro proyecto. Por otro lado, por mi parte también he ido realizando actividades de ayuda sociales, desde dirigir actividades para niños en esplais hasta ofrecer mi ayuda en comedores para gente en necesidad.

En conclusión, creo que gracias a la universidad he adquirido mucha experiencia en el ámbito de la sostenibilidad. Cierto es que aún tengo mucho que aprender, pero al menos me han hecho concienciarme de la importancia que tiene y de cómo hay que tener las tres en cuenta para que un proyecto tenga éxito.

15. CONCLUSIONES Y TRABAJO FUTURO

Encontrándonos ya al final de proyecto, es el momento de mirar atrás y de examinar qué hemos aprendido de éste. Primero haremos una vista general de las conclusiones que podemos sacar, luego un pequeño apartado en el que hablaremos de posibles mejoras que nos quedamos con las ganas de aplicar y, por último, incluiremos la valoración personal del proyecto

15.1. CONCLUSIONES DEL PROYECTO

Al final del proyecto hemos logrado implementar todas y cada una de las funcionalidades que nos habíamos propuesto. Se ha realizado siguiendo la metodología *Agile Scrum* con 4 Sprints. En el primer Sprint nos dedicamos a investigar todo lo necesario para empezar con el proyecto. En el segundo Sprint desarrollamos un modelo sencillo al que entrenamos para aprender a identificar entre clases de setas y que luego evolucionó en complejidad al aplicarle Transfer Learning. Más tarde logramos incorporar el modelo en una sencilla app móvil y hacerla mínimamente funcional. En el tercer Sprint trabajamos en la comunicación con el servidor y, finalmente, en el último Sprint, con el resto de funcionalidades de la aplicación. Todas las funcionalidades funcionan dentro de lo previsto.

En resumen, las principales contribuciones de este TFG han sido:

1) Creación de un modelo de machine learning para reconocimiento de setas usando TensorFlow utilizando Transfer Learning para aprovechar una arquitectura ya preentrenada y optimizada para dispositivos móviles (MobileNetsv2).

2) Integración del componente ML en una aplicación móvil para el reconocimiento de setas, utilizando el formato eficiente y portable tensorflow lite (.tflite).

4) Desarrollo de las funcionalidades de una aplicación móvil para el reconocimiento de setas: búsqueda de imágenes en la galería, lectura de dichas imágenes, interpretación del modelo en el dispositivo, extracción de resultados y mostrado de resultados.

5) Despliegue de un servidor en DigitalOcean para permitir la acogida de nuevas imágenes enviadas desde la aplicación, el reentreno automático del componente ML con esas nuevas imágenes y la descarga de dicho modelo actualizado.

6) Desarrollo de las funcionalidades de red necesarias para el envío de nuevas imágenes de setas para reentrenar el modelo y la descarga del componente ML del servidor.

7) Desarrollo de otras funcionalidades para facilitar el uso de la aplicación y hacerla más atractiva a los usuarios: gestión de perfiles, consulta de entradas de una enciclopedia y técnicas de gamificación.

8) Uso y aplicación de buenas prácticas de ingeniería en aplicaciones de software con componentes ML.

15.2. REFUTACIONES Y MEJORAS DEL FUTURO

A lo largo del desarrollo del proyecto fueron surgiendo ideas que, aunque podrían haber hecho el proyecto mucho más completo, consideramos que estaban fuera de nuestro alcance. Es por eso que en este pequeño apartado queremos presentar algunas de estas ideas:

1- Integración de esta aplicación en una red social: Una de las primeras ideas que surgieron fue integrar este proyecto en una red social. La idea es que esta podría conectar a entusiastas de la recogida de setas de todo el mundo y podrían apoyarse mutuamente entre ellos. Esto aportaría una gran riqueza funcional a la aplicación permitiendo, por ejemplo, que los usuarios conectaran entre ellos para ayudarse mutuamente a identificar una seta.

2- Expansión del número de especies de setas incluidas en el modelo: Una mejora claramente necesaria sería aumentar el número de especies de setas incluidas en el modelo. La idea al principio sería gestionar una serie de actualizaciones que poco a poco fueran incluyendo estas nuevas especies de setas. También sería interesante permitir a los usuarios subir sugerencias con nuevas colecciones de setas, y que éstas fueran incluidas a la aplicación después de ser consideradas por el equipo detrás.

3- Ofrecer modelos adaptados a cada localidad: No todas las setas crecen en todos los lugares, por lo que tal vez a un usuario que vive en una región donde no florezcan Amanitas le interese un modelo más especializado sin la intrusión de esta clase. Por lo tanto, la idea sería ofrecer modelos especializados en cada región.

4- Contratación de un equipo que haga seguimiento del desarrollo de la aplicación: Una mejora claramente necesaria para que este proyecto pueda funcionar a largo plazo es

que tiene que existir la figura de un analista de datos que cada poco tiempo vaya revisando los resultados de los modelos. Además de eso, sería necesario contratar a un moderador que tome todas las muestras de setas y se asegure de corregir aquellas en los que los usuarios se hayan confundido.

15.3. VALORACIÓN PERSONAL DEL PROYECTO

Este proyecto me ha servido para solidificar mis conocimientos de gestión de proyectos y, sobre todo, para abrir mi mente al mundo de las redes neuronales. Machine Learning para mí siempre ha sido un misterio que tenía ganas de explorar pero del cual no tenía conocimiento alguno. Cuando me vino a la cabeza la posibilidad de poder desarrollar un proyecto relacionado con esta temática, no pude negarme.

El proceso de aprendizaje ha sido duro, pero ha valido la pena. Las primeras semanas fueron identificadas por una sensación continua de terror y de vértigo ante lo que se me avecinaba: quería hacer un proyecto de machine learning y no sabía ni por dónde empezar. Tampoco me pude quitar esa sensación de vértigo durante el resto del proyecto, puesto que a medida que iba implementando y desarrollando, me iba encontrando con más y más desafíos que no estaba seguro de ser capaz de superar.

Afortunadamente, mediante una planificación exhaustiva y un seguimiento continuo por parte de mis directores, he sido capaz de superar todos los desafíos uno a uno y, al hacerlo, he podido disfrutar de su correspondiente satisfacción.

A pesar de todo, soy del todo consciente de que en un solo proyecto he tocado muchos temas y sólo he podido mojar los pies ligeramente en las variadas temáticas: tal vez me hubiera gustado más escoger alguna temática más concreta y haberme centrado en ella

Por otro lado, considero que el haber tocado tantos temas me ha dado una visión general de la tecnología de las aplicaciones móviles, servidores y modelos machine learning extremadamente valiosos para mi futuro.

16. REFERENCIAS

[1] "Cada año se producen 400 casos graves de intoxicaciones por setas", *El Heraldo* (En línea) 26/2/2021 (Consultado el 26/02/2021). Disponible en: *https://www.heraldo.es/noticias/salud/2018/10/08/cada-ano-producen-400-casos-graves-intoxicaciones-por-setas-1270707-2261131.html#:~:text=Cada%20a%C3%B1o%20se%20producen%20en,un%20trasplante%20de%20h%C3%ADgado%20urgente*.

[2] Siebert, Julien; Joeckel, Lisa; Feidrich, Hens; Nakamichi, Koji; Ohashi, Kyoko; Namba, Isao; Yamamoto, Rieko; Aoyama, Mikio. "Towards Guidelines for Assessing Qualities of Machine Learning Systems". *Proceedings of the 13th International Conference on the Quality of Information and Communications Technology* (En línea). 2018 (Consultado 7/4/2021). Disponible en: https://arxiv.org/abs/2008.11007.

[3] González, Andrés. "¿Qué es el Machine Learning?". *Cleverdata* (En línea). 2018 (Consultado el 25/02/2021). Disponible en: *https://cleverdata.io/que-es-machine-learning-big-data.*

[4] Redacción APD. "¿Cuáles son los tipos de algoritmos del machine learning?". *APD* (En línea). 4/4/2019 (Consultado el 27/02/2021). Disponible en: *https://www.apd.es/algoritmos-del-machine-learning/.*

[5] Annapurnapp Technologies. Mushroom Identify. 9/7/2020 (Consultado el 26/02/2021) Disponible en: *https://play.google.com/store/apps/details?id=com.pingou.champignouf.*

[6] Foragers Friends. Shroomify. 11/11/2020 (Consultado el 26/02/2021). Disponible en: *https://play.google.com/store/apps/details?id=com.mushroom.shroomify.*

[7] Next Vision Limited. Picture mushroom. 10/11/2020 (Consultado el 27/02/2021). Disponible en: *https://play.google.com/store/apps/details?id=com.glority.picturemushroom.*

[8] Steinhauser, Dominik. Fungus - Identification of fungi. 30/4/2021 (Consultado el 27/02/2021) Disponible en: *https://play.google.com/store/apps/details?id=com.steinhauser.mushrooms2.*

[9] Equipo de redacción Drew. "Ventajas y desventajas de la metodología scrum". *Drew* (En línea). 3/12/2019 (Consultado el 22/3/2021). Disponible en: *https://blog.wearedrew.co/ventajas-y-desventajas-de-la-metodologia-scrum.*

[10] Atlassian. Trello. (Consultado el 28/02/2021) Disponible en: *https://trello.com/en.*

[11] Github. Github. (Consultado el 28/02/2021) Disponible en: *https://github.com/*.

[12] Dropbox team. Dropbox. (Consultado el 28/02/2021) Disponible en: *https://www.dropbox.com/?_hp=c*.

[13] Indeed**. (**Consultado el 13/3/2021**).**Disponible en: *https://es.indeed.com/*

[14] Zamness**. (**Consultado el 13/3/2021**).** Disponible en: *https://zamness.com/*.

[15] Se4ML team. "Engineering best practices for Machine Learning". *SE4ML* (En línea) 2020 (Consultado el 7/4/2021). Disponible en: *https://se-ml.github.io/practices/*.

[16] Sandler, Mark; Howard, Andrew; Zhu, Menglong; Zhmoginov, Andrey; Chen, Liang-Chieh. "MobilenetV2: Inverted Residuals and Linear Bottlenecks". *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018* (En línea). 2018 (Consultado el 17/4/2021). Disponible en: https://arxiv.org/abs/1801.04381

[17] Uchida, Yusuke. "Why MobileNet and its Variants (e.g.ShuffleNet) Are Fast" .*Medium* (En línea). 2020 (Consultado el 18/4/2021). Disponible en: *https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d.* 2018.