eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

LANDA

Alumno: Irastorza, Albo, Ibai

Directora: Atutxa, Salazar, Aitziber

Director: Villamañe, Gironés, Mikel

Curso: 2020-2021

Fecha: Bilbao, 2, noviembre, 2020

Preámbulo

Este proyecto surge de mi profunda admiración por la naturaleza y de la necesidad de aprender y descubrir el entorno que nos rodea. Se trata de una aplicación móvil para encontrar e identificar setas.

En las próximas páginas se detalla todo el proceso llevado a cabo en el desarrollo de este proyecto.

Índice general

Índice de figuras

1 Introducción

1.1 Descripción del proyecto

Landa es una aplicación móvil que permite encontrar e identificar setas.

La funcionalidad principal es un clasificador de hongos que se caracteriza por un proceso de dos partes:

- Se saca una foto a una seta, y se muestra una interfaz muy visual con un carrusel de las predicciones de la inteligencia artificial, ordenadas en base a la probabilidad. Por cada predicción, se mostrará unicamente tres fotos, el nombre y las características clave que le ayudarán al usuario a determinar si efectivamente, se trata del espécimen en cuestión.
- Una vez confirmado que se trata de la seta presentada, se le mostrará la ficha completa con toda la información, junto a un carrusel con las posibles confusiones y un botón para guardar el hallazgo y completar el proceso.

Por otra parte, se ha diseñado un mapa de zonas naturales, que se basa en, utilizando los mapas de Google, mostrar al usuario marcadas las zonas verdes cercanas, categorizadas por su ecología y distribución forestal. Además, por cada zona marcada, se le mostrará al usuario las setas que ahí pueden aparecer, teniendo en cuenta, claro está, factores ambientales tales como la época y las características meteorológicas de los últimos días.

1.2 Motivación detrás del proyecto

Este proyecto surge de mi profunda admiración por la naturaleza, y de la necesidad de aprender y descubrir nuevas formas de disfrutarla.
Y es que, como habitantes de grandes ciudades y consumidores de una sociedad frenética, nos hemos ido alejando del medio natural, olvidando la riqueza que tanto bienestar nos ha proporcionado.

El acercamiento al mundo de la micología es la propuesta de este proyecto. Se trata de una actividad divertida, emocionante, y muy gratificante. Añadiéndole el incentivo de poder disfrutar del manjar que supone cocinar algunas de estas especies.

1.3 Definiciones, acrónimos y abreviaturas

Para evitar posibles confusiones en la lectura de este documento, se van a definir algunos conceptos que pueden no estar al alcance de todo el mundo.

- **Machine learning**: Disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente.
- **Micología**: Ciencia que se dedica al estudio de los hongos. Es una de las áreas de la ciencia más extensas y diversificadas, que aporta avances significativos a la investigación científica y al desarrollo tecnológico.
- **Endpoint**: URL de un servicio que utiliza un sitio web para cargar o consumir información.
- **Scraping**: Técnica que sirve para extraer información de páginas web de forma automatizada, para conseguir cantidades industriales de información (Big data) sin teclear una sola palabra. A través de los algoritmos de búsqueda podemos rastrear centenares de webs para extraer sólo aquella información que necesitamos.
- **API**: *Application Programming Interface* es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.
- **Dataset**: Conjunto de datos. En este caso, todas las imágenes de las setas de las que se dispone, así como toda la información de cada una.
- **Mockup**: Prototípo o maqueta hecha de forma rápida para transmitir la idea del proyecto.
- **Apk**: Tipo de archivo ejecutable propio de las aplicaciones Android.
- **Epoch**: Hace referencia a cada ciclo de entrenamiento de una red neuronal.
- **SDK**: *Software Development Kit*, es el conjunto de herramientas de desarrollo de software.
- **RAM**: *Random Access Memory*, se utiliza como memoria de trabajo de computadoras y otros dispositivos.
- **CPU**: *Central Processing Unit*, es el hardware dentro de un ordenador u otros dispositivos programables, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.
- **GPU**: *Graphics Processing Unit*, es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante.
- **Bitmap**: literalmente, un mapa de bits, es decir, la representación binaria en la cual un bit o conjunto de bits corresponde a alguna parte de un objeto como una imagen o fuente.
- **DDOS**: *Denial of Service*, es un ataque que tiene como objetivo inhabilitar un servidor.

2 Planteamiento inicial

En este primer capítulo se expone el punto de partida desde el cual se ha construido este proyecto. Detallando los objetivos, el alcance, el diseño de la infraestructura, las tecnologías a explorar y las implicaciones logísticas que va a suponer.

2.1 Objetivos del proyecto

El objetivo principal de este proyecto es crear una aplicación que permita identificar setas, mediante una red neuronal, y encontrarlas gracias a un mapa con las zonas forestales marcadas. Más en concreto:

- Desarrollar una red neuronal capaz de identificar setas con una precisión de al menos el 90%.
- Construir un sistema de identificación de setas.
- Construir un catálogo de setas con imágenes e información.
- Desarrollar un sistema de exploración basado en un mapa de las zonas forestales de España.
- Conseguir la estabilidad y robustez en la aplicación.
- Diseñar una animación inicial basada en un paisaje.
- Establecer un modelo de negocio.
- Asegurar el cumplimiento de la legislación vigente.
- Configurar los aspectos comerciales de la plataforma de distribución y lanzar la aplicación al mercado.

2.2 Herramientas

Para llevar a cabo este proyecto es necesario utilizar varias herramientas y tecnologías. Entre ellas, segmentadas por su funcionalidad, se encuentran:

- **Gestión, organización y control**
  - **Office 365** (Word, Excel, OneDrive, ...): Toda la suite de *Microsoft Office* es necesaria para la realización de la documentación, la planificación y la sincronización de archivos entre dispositivos.
  - **Github**: Plataforma para alojar el código del proyecto.
  - **Trello**: Gestor de tareas online.
- **Diseño gráfico**
  - **Adobe Illustrator**: Editor gráfico vectorial para elaborar todos los diseños.
  - **After Effects**: Editor de video con el que diseñar las animaciones.
  - **Adobe Photoshop**: Editor de imágenes con el que retocar los gráficos.
  - **Adobe XD**: Herramienta para elaborar la maqueta.
- **Desarrollo**
  - **Android Studio**: Herramienta para desarrollo de aplicaciones *Android*.
  - **Visual Studio Code**: Editor de código que se va a usar para el desarrollo en python de los scripts de *machine learning*.
  - **Tensorflow**, **keras**, **numpy**, **pandas**, y varias librerías más de *Machine learning* y *data science*.
  - **Firebase**: Plataforma en la nube de Google para el desarrollo de aplicaciones web y móvil. Se va a utilizar por sus múltiples servicios y especialmente el alojamiento de las bases de datos y la autenticación. Además dispone de razonables límites de uso gratuitos a los que se intentará acoger.
  - **Google Play Console**: Plataforma de distribución mediante la cual se va a lanzar la aplicación al mercado.
  - **SQLite**: Sistema de gestión de bases de datos relacional elegido para este proyecto.
  - **Telegram**: Plataforma de comunicaciones con varias herramientas para desarrolladores.
- **Documentación**
  - **LaTEX Enviroment**: Extensión instalada en *Visual Studio Code* para elaborar la documentación.
  - **GanttProject**: Herramienta para la elaboración de diagramas *Gantt*.
  - **Adobe Illustrator**: Editor gráfico vectorial utilizado para la elaboración de todos los gráficos, casos de uso, diagramas de secuencia, ...

2.3 Arquitectura

La arquitectura de esta aplicación es en mayor parte local, sin embargo, ciertas funcionalidades utilizan la descarga y respaldo de datos en el servidor. A continuación se expone en detalle la arquitectura diseñada:

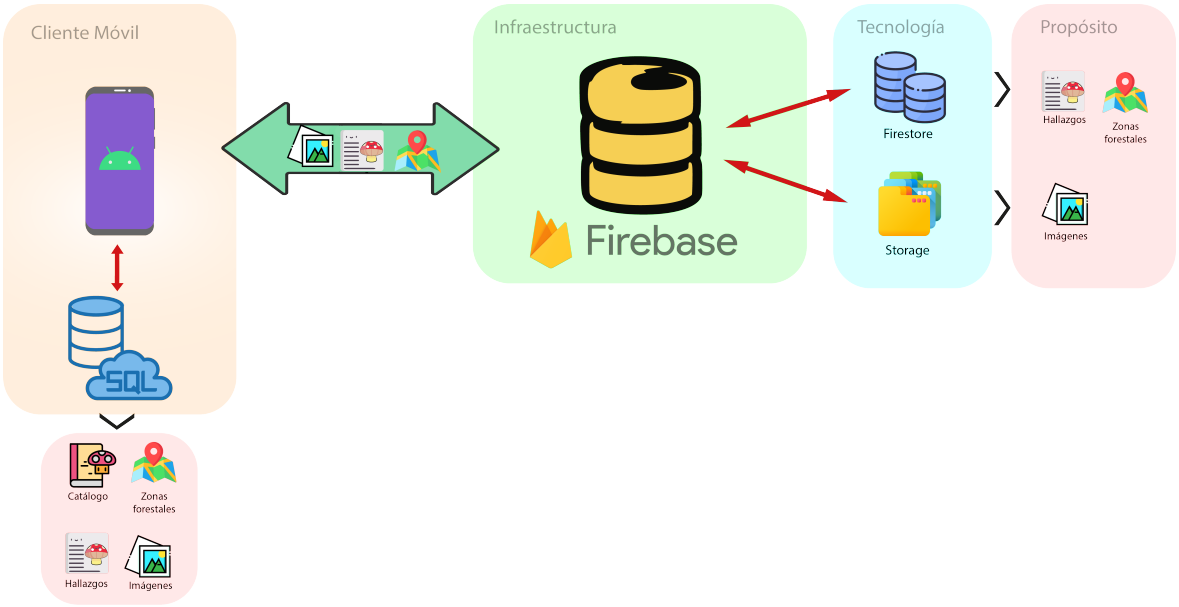


Figura 2.1: Diagrama de la arquitectura

La aplicación debe ser capaz de funcionar completamente *offline*, por lo tanto, la base de datos interna *SQLite* albergará todos los datos necesarios.

- El catálogo de setas con sus imágenes e información.
- Los hallazgos realizados por el usuario con la correspondiente imagen, fecha y nombre de la seta.
- Las zonas forestales.

Estos datos irán pre-empaquetados con la aplicación, y se dispondrá de ellos desde la instalación. Sin embargo, la colosal cantidad de zonas forestales existentes hace impracticable alojarla en su totalidad de forma local. Por lo tanto, en la primera ejecución de la aplicación, se descargarán únicamente las zonas en un radio de diez kilómetros de su posición.
Por otra parte, si el usuario lo desea, los hallazgos que realize se enviarán a la plataforma *Firebase* para almacenarlos y poder recuperarlos en caso de instalaciones en otros dispositivos.

También comentar que por ahora el desarrollo se va a centrar en el sistema operativo Android. Disponer de una versión para el sistema operativo iOS queda fuera del alcance del proyecto, porque supondría exceder el tiempo que se espera de un trabajo fin de grado.

De la plataforma *Firebase* también se utilizarán los siguientes servicios:

- **Authentication:** Para gestionar el registro y login de usuarios con su cuenta de Google.
- **Firestore:** Sistema de base de datos principal en la que se pretende alojar la información de los hallazgos de los usuarios y las zonas forestales.
- **Storage:** Sistema de almacenamiento principal en el que se pretenden alojar las imágenes de los hallazgos.
- **Crashlytics:** Esta imprescindible funcionalidad permite monitorizar, analizar y controlar los *bugs* y la estabilidad del sistema, una vez la aplicación se haya lanzado a producción.
- **Analytics:** Otro servicio imprescindible para monitorizar la acogida del mercado.

2.4 Alcance del proyecto

En este apartado se detalla el trabajo que es necesario realizar, en que orden y durante cuanto tiempo.

2.4.1 Ciclo de vida

El ciclo de vida de este proyecto es incremental. Es decir, se plantea el desarrollo de las funcionalidades de forma progresiva e iterativa, y se testea cada prototipo para poder enfocar el producto a las necesidades de los usuarios.
Se ha elegido esta metodología por ser la más orgánica y que más se aproxima a los paradigmas de las metodologías ágiles.

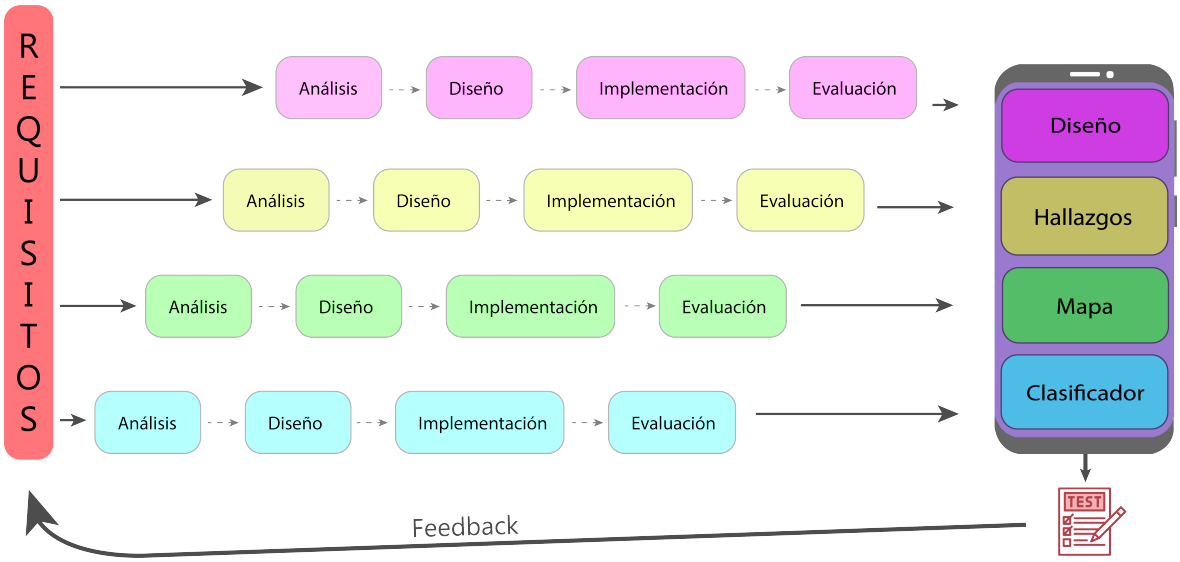


Figura 2.2: Ciclo de vida

Se realizarán cuatro versiones de la aplicación de forma incremental. Es decir, primero se desarrollará el sistema de identificación, luego se le añadirá el sistema de exploración, a continuación se le añadirá el sistema de gestión de hallazgos y finalmente el aspecto gráfico.
Por cada iteración se realizará el análisis, el diseño, la implementación y la evaluación de esa parte.

Segmentando el desarrollo en esos cuatro bloques, se pretende facilitar su implementación y garantizar que cada nueva funcionalidad se construya sobre una base sólida.

2.4.2 Estructura de descomposición del trabajo

Cada una de las tareas y fases que forman parte del proyecto se representan en la Estructura de Descomposición del Trabajo (EDT). Aquí se encuentran:

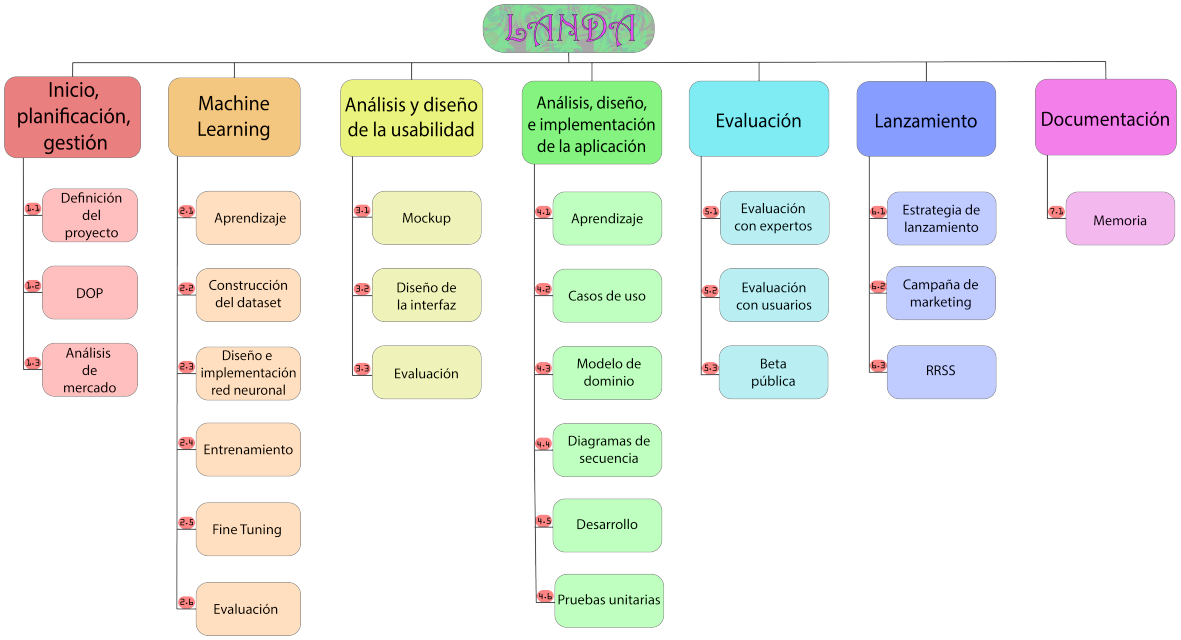


Figura 2.3: EDT

Como se puede apreciar, el EDT se compone de las siguientes fases:

- **Inicio, planificación y gestión:** Fase inicial del proyecto en la que se debe definir todos los aspectos relacionados con su puesta en marcha y su implementación.
- **Machine Learning:** Fase de varios pasos en los que se debe construir el dataset y la red neuronal.
- **Análisis y diseño de la usabilidad:** Fase en la que se va a diseñar e implementar toda la parte gráfica.
- **Análisis, diseño e implementación de la aplicación:** Fase en la que se va a implementar la aplicación móvil y su correspondiente infraestructura.
- **Evaluación:** Fase en la que se contrasta si todo el trabajo realizado tiene la calidad suficiente.
- **Lanzamiento:** Fase de distribución de la aplicación.
- **Documentación:** Elaboración de la documentación de todo el proyecto.

2.4.3 Paquetes de trabajo

A continuación se definirá de forma exhaustiva los detalles de cada tarea.

- **Inicio, planificación y gestión**

1.1	**Definición del proyecto**
Descripción	No se puede empezar un proyecto sin tener claro que se quiere hacer. Por eso, esta primera tarea consiste en definir y acotar la magnitud del proyecto.
Duración	20 horas
Entradas	Ninguna
Salidas	Objetivos
Recursos	Ninguno
Precedencias	Ninguna

1.2	**Análisis de mercado**
Descripción	Una vez definido que se quiere hacer, es imprescindible conocer la existencia de proyectos similares en el mercado.
Duración	30 horas
Entradas	Ninguna
Salidas	Análisis de mercado
Recursos	LaTEX Enviroment, dispositivo móvil
Precedencias	1.1-Definición del proyecto

1.3	**DOP**
Descripción	Tarea crucial en la que se debe planificar todos los aspectos del proyecto y plasmarlos en el Documento de Objetivos del Proyecto.
Duración	100 horas
Entradas	Análisis de mercado
Salidas	Documento de Objetivos del Proyecto
Recursos	LaTEX Enviroment, Adobe Illustrator, GanttProject
Precedencias	1.1-Definición del proyecto, 1.2-Análisis de mercado

- **Machine Learning**

2.1	**Aprendizaje**
Descripción	Es necesario, antes de iniciar esta fase, adquirir conocimientos sobre *Deep Learning*, *Image classification*, *Transfer Learning*, *Keras*, *Tensorflow*, procesamiento de imágenes, ...
Duración	150 horas
Entradas	Ninguna
Salidas	Ninguna
Recursos	Internet
Precedencias	Ninguna

2.2	**Construcción del dataset**
Descripción	Probablemente la tarea más importante de esta fase. Es necesario determinar cuantas especies de hongos se van a tratar, cuantas imágenes se van a almacenar por cada especie, y que información interesa disponer de cada especie. Además, se deberán desarrollar scripts para *scrapear* las imágenes de internet (con sus respectivas licencias), recolectar información, organizar y limpiar las imágenes.
Duración	300 horas
Entradas	Ninguna
Salidas	Un catálogo de imágenes e información
Recursos	Un servidor, python y un gestor de bases de datos
Precedencias	2.1-Aprendizaje

2.3	**Diseño e implementación de la red neuronal**
Descripción	Una vez adquiridos los conocimientos y sabiendo el problema que se quiere solventar, el siguiente paso es diseñar una red neuronal e implementar el correspondiente script.
Duración	200 horas
Entradas	Ninguna
Salidas	Script de entrenamiento
Recursos	Python y varias librerías (Keras, Tensorflow, numpy, pandas, ...)
Precedencias	2.1-Aprendizaje, 2.2-Construcción del dataset

2.4	**Entrenamiento**
Descripción	Teniendo el dataset y el código, es hora de ponerse a entrenar.
Duración	700 horas desatendidas.
Entradas	Dataset y scripts
Salidas	Modelo
Recursos	Un servidor
Precedencias	2.2-Construcción del dataset, 2.3-Diseño e implementación de la red neuronal

2.5	**Fine Tuning**
Descripción	Entre entrenamiento y entrenamiento hay que ir ajustando los hiperparámetros y comprobar si mejoran los resultados.
Duración	50 horas
Entradas	Script de entrenamiento
Salidas	Ninguna
Recursos	Ninguno
Precedencias	2.4-Entrenamiento

2.6	**Evaluación**
Descripción	Se debe evaluar el rendimiento de los modelos generados.
Duración	50 horas
Entradas	Modelos
Salidas	Métricas
Recursos	Ninguno
Precedencias	2.4-Entrenamiento

- **Análisis y diseño de la usabilidad**

3.1	**Mockup**
Descripción	Antes de lanzarse a implementar la aplicación, conviene esbozar un prototipo y exponerlo a las opiniones de usuarios finales, para detectar prematuramente fallos de concepto, una dirección desafortunada o incluso recolectar ideas de mejora.
Duración	50 horas
Entradas	Casos de uso
Salidas	Maqueta
Recursos	Adobe XD
Precedencias	1.3-DOP, 4.2-Casos de uso

3.2	**Diseño de la interfaz**
Descripción	En base a la información recogida del *mockup*, se procede a diseñar las interfaces gráficas de la aplicación, los iconos y animaciones.
Duración	150 horas
Entradas	Maqueta
Salidas	Layouts, iconos, animaciones
Recursos	Adobe Illustrator, Android Studio, Adobe After Effects
Precedencias	3.1-Mockup

3.3	**Evaluación**
Descripción	Una vez implementada la aplicación, se expondrá al criterio de usuarios para detectar errores, incongruencias, excesos o insuficiencias.
Duración	10 horas
Entradas	La aplicación
Salidas	Conclusiones
Recursos	Grupos de personas
Precedencias	4.5-Desarrollo

- **Análisis, diseño e implementación de la aplicación**

4.1	**Aprendizaje**
Descripción	La envergadura y la complejidad de este proyecto requiere adquirir conocimientos avanzados sobre todo lo relacionado con Android y el desarrollo multiplataforma.
Duración	150 horas.
Entradas	Ninguna
Salidas	Ninguna
Recursos	Internet
Precedencias	Ninguna

4.2	**Casos de uso**
Descripción	El primer paso en todo proceso de desarrollo es definir los casos de uso.
Duración	40 horas
Entradas	Ninguna
Salidas	Casos de uso
Recursos	Adobe Illustrator
Precedencias	1.3-DOP

4.3	**Modelo de dominio**
Descripción	El segundo paso en todo proceso de desarrollo es deducir el modelo de dominio a partir de los casos de uso.
Duración	40 horas
Entradas	Casos de uso
Salidas	Modelo de dominio
Recursos	Adobe Illustrator
Precedencias	4.2-Casos de uso

4.4	**Diagramas de secuencia**
Descripción	Hay que diseñar diagramas de secuencia por cada funcionalidad de la aplicación.
Duración	50 horas
Entradas	Casos de uso
Salidas	Diagramas de secuencia
Recursos	Adobe Illustrator
Precedencias	4.3-Modelo de dominio

4.5	**Desarrollo**
Descripción	Programar la aplicación y la infraestructura.
Duración	450 horas
Entradas	Casos de uso, diagramas de secuencia, modelo de dominio
Salidas	La aplicación
Recursos	Android Studio, internet
Precedencias	4.4-Diagramas de secuencia, 4.1-Aprendizaje

4.6	**Pruebas unitarias**
Descripción	Testear programáticamente la aplicación y la infraestructura.
Duración	50 horas
Entradas	La aplicación
Salidas	Ninguna
Recursos	Android Studio
Precedencias	4.5-Desarrollo

- **Evaluación**

5.1	Evaluación de expertos
Descripción	Es imprescindible exponer la aplicación al criterio de micólogos expertos para detectar errores críticos.
Duración	50 horas
Entradas	La aplicación
Salidas	Conclusiones
Recursos	Ninguno
Precedencias	4.5-Desarrollo

5.2	Evaluación de usuarios
Descripción	Se deben hacer estudios con grupos de personas de distintas características para comprobar la recepción de las distintas demografías.
Duración	50 horas
Entradas	La aplicación
Salidas	Conclusiones
Recursos	Grupos de personas
Precedencias	4.5-Desarrollo

5.3	Beta pública
Descripción	Por último, para comprobar la recepción del mercado y exponer el sistema a condiciones reales impredecibles, se lanzará una versión *Beta* de la aplicación.
Duración	50 horas
Entradas	La aplicación
Salidas	Ninguna
Recursos	Google Play
Precedencias	5.2-Evaluación de usuarios

- **Lanzamiento**

6.1	Estrategia de lanzamiento
Descripción	Es importante planear una estrategia para el lanzamiento de la aplicación.
Duración	10 horas
Entradas	Ninguna
Salidas	Estrategia de lanzamiento, página de Google Play
Recursos	Google Play
Precedencias	5.3-Beta pública

6.2	Campaña de marketing
Descripción	Como parte de la estrategia de lanzamiento, se diseñará y ejecutará una campaña de marketing pre y post lanzamiento.
Duración	20 horas
Entradas	Ninguna
Salidas	Campaña de marketing
Recursos	Ninguno
Precedencias	6.1-Estrategia de lanzamiento

6.3	Redes sociales
Descripción	Como parte de la campaña de marketing, y para generar tráfico orgánico y darse a conocer, se van a emplear distintas redes sociales.
Duración	50 horas
Entradas	Campaña de marketing
Salidas	Ninguna
Recursos	Redes sociales
Precedencias	6.2-Campaña de marketing

- **Documentación**

7.1	Memoria
Descripción	Documentarlo todo.
Duración	200 horas
Entradas	DOP
Salidas	Memoria
Recursos	LaTEX Enviroment, Adobe Illustrator
Precedencias	1.3-DOP

2.4.4 Planificación temporal

En esta tabla se muestra la totalidad del esfuerzo por cada funcionalidad y a nivel de proyecto.

Tareas	Duración
Inicio, planificación, gestión	**150 horas**
1.1 Definición del proyecto	20 horas
1.2 Análisis de mercado	30 horas
1.3 DOP	100 horas
Machine Learning	**750 horas**
2.1 Aprendizaje	150 horas
2.2 Construcción del dataset	300 horas
2.3 Diseño e implementación de la red neuronal	200 horas
2.4 Entrenamiento	(700) horas
2.5 Fine Tuning	50 horas
2.6 Evaluación	50 horas
Análisis y diseño de la usabilidad	**210 horas**
3.1 Mockup	50 horas
3.2 Diseño de la interfaz	150 horas
3.3 Evaluación	10 horas
Análisis, diseño e implementación de la aplicación	**780 horas**
4.1 Aprendizaje	150 horas
4.2 Casos de uso	40 horas
4.3 Modelo de dominio	40 horas
4.4 Diagramas de secuencia	50 horas
4.5 Desarrollo	450 horas
4.6 Pruebas unitarias	50 horas
Evaluación	**150 horas**
5.1 Evaluación de expertos	50 horas
5.2 Evaluación de usuarios	50 horas
5.3 Beta pública	50 horas
Lanzamiento	**80 horas**
6.1 Estrategia de lanzamiento	10 horas
6.2 Campaña de marketing	20 horas
6.3 Redes sociales	50 horas
Documentación	**200 horas**
7.1 Memoria	200 horas
Total	2.320 horas

Y su correspondiente planificación en formato Gantt:

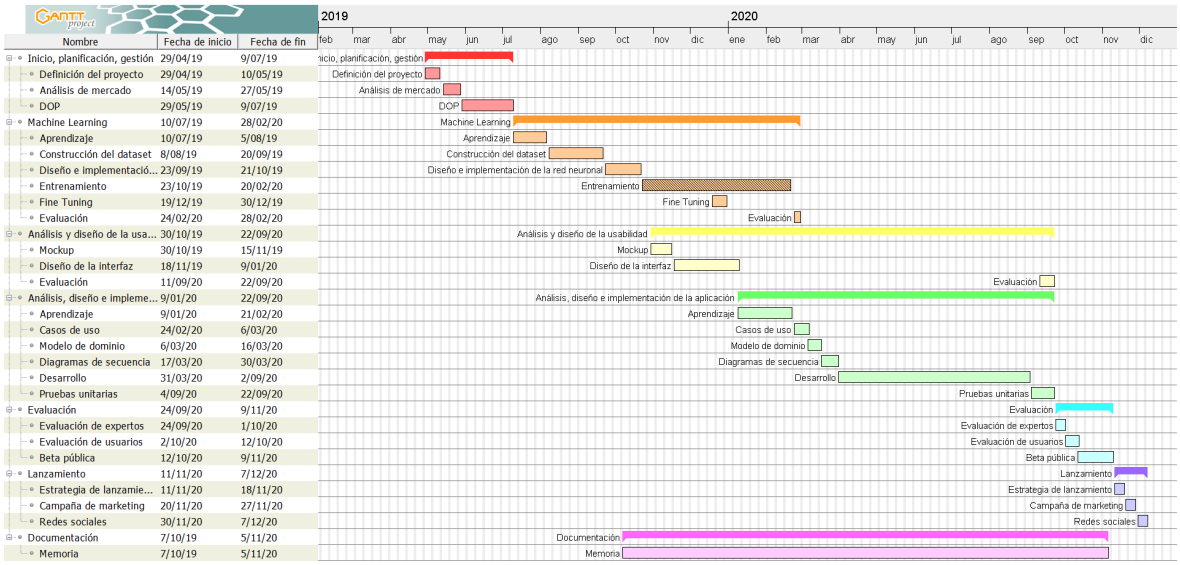


Figura 2.4: Diagrama Gantt

En el cual se puede observar el año y medio de duración del proyecto. Además de que la fase de lanzamiento queda para después de la defensa, y la fase de entrenamiento no se cuenta al tratarse de horas desatendidas.

2.5 Gestión de riesgos

A continuación se exponen las amenazas que acechan a este proyecto, divididas en categorías y coloreadas en base a su nivel de riesgo.

- **Riesgos de desarrollo**

Funcionalidad demasiado complicada	
Descripción	El desarrollo de alguna de las funcionalidades resulta excesivamente costoso y complicado.
Prevención	Un correcto análisis y diseño previo es imprescindible.
Plan de contingencia	Simplificar la funcionalidad, buscar una alternativa o prescindir de ella.
Probabilidad	Media
Impacto	Alto

Cambio drástico en los servicios principales	
Descripción	Ya sea que el SDK de Google Maps se vuelva de pago o Firebase deje de dar servicio, los proveedores sobre los que se basa este proyecto pueden fallar.
Prevención	Conocer alternativas y diseñar los sistemas para poder cambiar estos servicios sin demasiada complicación.
Plan de contingencia	Buscar una alternativa e implementarla.
Probabilidad	Baja
Impacto	Muy alto

Perdida de datos del proyecto	
Descripción	En algún momento los discos duros fallan o los servicios de copias de seguridad en la nube sufren algún contratiempo.
Prevención	Copias de seguridad periódicas en discos externos y tener la información almacenada en varios servicios en la nube.
Plan de contingencia	Recuperar la copia de seguridad más reciente y evaluar la información perdida.
Probabilidad	Baja
Impacto	Muy alto

- **Riesgos de producto**

Aplicación demasiado voluminosa	
Descripción	Una vez acabado el desarrollo, la aplicación exportada supera los 500MB.
Prevención	Seguir las directrices de desarrollo de Google y modularizar la carga de datos.
Plan de contingencia	Reducir la resolución de los gráficos y externalizar la carga de datos.
Probabilidad	Muy alta
Impacto	Medio

Sobrecarga de la CPU	
Descripción	La aplicación en ejecución, en algún momento de su ciclo de vida, requiere excesiva capacidad de procesamiento y ralentiza visiblemente el terminal.
Prevención	Eficacia, eficiencia y muchos threads en segundo plano.
Plan de contingencia	Ocultar cargas y procesos en animaciones y transiciones.
Probabilidad	Muy alta
Impacto	Muy alto

Sobrecarga de la ram	
Descripción	En algún momento de la ejecución, el uso de la RAM supera el límite permitido.
Prevención	Cargar únicamente lo necesario y ajustar la resolución de los Bitmaps.
Plan de contingencia	Limitar el número de imágenes, capas y procesos por vista.
Probabilidad	Muy alta
Impacto	Muy alto

Sobrecarga de la GPU	
Descripción	La cantidad de animaciones, imágenes o capas provoca la sobrecarga de la GPU.
Prevención	Analizar el uso de GPU.
Plan de contingencia	Reducir el número de gráficos por vista.
Probabilidad	Alta
Impacto	Medio

Uso excesivo de internet	
Descripción	La carga y descarga de datos e imágenes consume demasiados datos móviles.
Prevención	Insistir en realizar todas las descargar con wifi.
Plan de contingencia	Avisos y limitar las descargas a transmisiones wifi.
Probabilidad	Muy alta
Impacto	Medio

Tiempos de carga elevados	
Descripción	La complejidad de las operaciones provoca tiempos de espera excesivos en las funcionalidades.
Prevención	Precargar datos y utilizar el caché.
Plan de contingencia	Diseñar una animación de carga entretenida.
Probabilidad	Muy alta
Impacto	Alto

Clasificador impreciso	
Descripción	La red neuronal no es capaz de discernir entre especies con la suficiente confianza.
Prevención	Disponer de muchas imágenes y no escatimar en tiempo de entrenamiento.
Plan de contingencia	Aumentar el número de imágenes por cada clase y aumentar el número de *epoch*.
Probabilidad	Muy alta
Impacto	Medio

- Riesgos estructurales

Caída del sistema	
Descripción	En algún momento se interrumpe el acceso a los servicios cloud.
Prevención	Asegurar la funcionalidad sin conexión y posponer el tráfico a cuando se reconecte el sistema.
Plan de contingencia	Aviso y disculpa.
Probabilidad	Baja
Impacto	Alto

Sobrecarga del sistema	
Descripción	La cantidad de peticiones, lecturas, escrituras, cargas y descargas simultaneas incapacita momentaneamente la infraestructura.
Prevención	Asegurar la capacidad de escalabilidad de la infraestructura.
Plan de contingencia	Aviso y disculpa.
Probabilidad	Media
Impacto	Alto

Ataques al sistema	
Descripción	Sufrir ataques DDOS o de robo de bases de datos.
Prevención	Configurar correctamente las reglas de acceso a las bases de datos, externalizar la autenticación, seguir las recomendaciones de google y auditar la infraestructura.
Plan de contingencia	Determinar la gravedad e implementar una solución.
Probabilidad	Baja
Impacto	Muy alto

- Riesgos de mercado

No conseguir usuarios activos	
Descripción	Una vez lanzada la aplicación, nadie la usa, ya sea porque no es inmediatamente accesible mediante el buscador, o la aplicación no es lo suficientemente util.
Prevención	Tener una buena aplicación, buena campaña de marketing, buena exposición y buena comunicación.
Plan de contingencia	Identificar el problema y solventarlo.
Probabilidad	Muy alta
Impacto	Bajo

Recibir críticas y malas puntuaciones	
Descripción	Recibir críticas y malas puntuaciones en Google Play.
Prevención	Tener una buena aplicación.
Plan de contingencia	Arreglar los desperfectos.
Probabilidad	Muy alta
Impacto	Medio

Intoxicaciones a causa de la aplicación	
Descripción	Usuarios se intoxican por usar la aplicación.
Prevención	Avisos, buenas prácticas, dejar claro que la aplicación solo es una herramienta adicional a otras formas de identificación y redactar una clausula de exención de responsabilidad en los términos de uso y condiciones.
Plan de contingencia	Identificar que ha fallado en el proceso y solventarlo.
Probabilidad	Medio
Impacto	Muy alta

Funcionalidad inútil	
Descripción	Una vez lanzada la aplicación se descubre que alguna de las funcionalidades no se usa.
Prevención	Un buen análisis de necesidades, y una exhaustiva fase de evaluación.
Plan de contingencia	Pivotar o prescindir de la funcionalidad.
Probabilidad	Media
Impacto	Medio

- Riesgos legales

Violar derechos de autor	
Descripción	Ser denunciado por utilización indebida de algún material.
Prevención	Asegurarse de disponer de licencias para todo y atribuir adecuadamente a los autores.
Plan de contingencia	Eliminar el material en cuestión y pedir disculpas.
Probabilidad	Baja
Impacto	Alto

Violar clausulas de la LOPD	
Descripción	Hacer un tratamiento indebido de los datos de usuario.
Prevención	Seguir las directrices correspondientes.
Plan de contingencia	Solucionarlo y pedir disculpas.
Probabilidad	Media
Impacto	Alto

- Riesgos individuales

Perder la motivación	
Descripción	Perder la motivación o quemarse.
Prevención	No obsesionarse con el proyecto, establecer tiempos de descanso y huir de la perfección.
Plan de contingencia	Tomarse unas vacaciones.
Probabilidad	Media
Impacto	Muy alto

No ser capaz de desarrollar alguna parte	
Descripción	Ya sea por tiempo o esfuerzo, alguna fase del proyecto se puede complicar.
Prevención	Planear desarrollos dentro de mis capacidades.
Plan de contingencia	Pivotar el concepto o obviarlo.
Probabilidad	Baja
Impacto	Alto

Exceder el tiempo límite para realizar el TFG	
Descripción	La complejidad del proyecto retrasa los plazos hasta llegar a excederlos.
Prevención	Acotar el proyecto a objetivos asequibles.
Plan de contingencia	Recortar el proyecto o entregarlo tal cual.
Probabilidad	Baja
Impacto	Alto

- Riesgos económicos

Los gastos superan a los ingresos	
Descripción	Los costes de mantenimiento de la infraestructura superan a los ingresos.
Prevención	Limitar el uso de funciones cloud y exprimir las cuotas gratuitas.
Plan de contingencia	Limitar el uso de la aplicación a offline o detener el servicio.
Probabilidad	Alta
Impacto	Alto

2.6 Evaluación económica

Basándose en la planificación temporal, las herramientas y la arquitectura, se presenta el presupuesto del proyecto, junto al modelo de negocio y su correspondiente rentabilidad.

2.6.1 Presupuesto

Recursos humanos

Según la universidad europea, el salario medio de ingenieros informáticos recién graduados ronda los 1200€ netos mensuales. Teniendo en cuenta las 2.320 horas de trabajo requeridas para este proyecto, se deduce que la inversión en recursos humanos debería ser de aproximadamente 17.400€.

Descripción	Horas	Honorarios/Hora	Coste total
Mano de obra	2.320 horas	7.5€	17.400€

Maquina/herramienta

Para hacer esto posible, se dispone de un modesto ordenador de sobremesa valorado en unos 600€ con una vida util de 10 años. Su utilización en los 18 meses de desarrollo del proyecto, resulta en una amortización de unos 90€.

Descripción	Coste	Amortización	Coste de 18 meses
Ordenador de sobremesa	600€	10 años	90€

Software

Algunos de los programas que se van a utilizar tienen un precio que se debería abonar. Además del uso de licencias y tasas de servicio.

Descripción	Coste
Microsoft Office 365	126€
Suite de Adobe	900€
Licencia desarrollador Google Play	25€

Infraestructura

El sistema de APIs y servidores necesarios para implementar la arquitectura de la aplicación supone un gasto variable dependiente del número de usuarios de la aplicación. La mayoría de los servicios ofrecen cuotas gratuitas de uso mínimo a las que se intentará fervientemente acoger. Sin embargo, en el afortunado supuesto en el que la cantidad de usuarios y su correspondiente uso de la aplicación supere las cuotas gratuitas, conviene estudiar el coste que supondría:

Servicio	Descripción	Uso/Usuario	Límite gratuito
Firebase Database	Base de datos para almacenar la información de los hallazgos y las zonas forestales	1 MB	1GB
Firebase Storage	Contenedor para alojar las imágenes de los hallazgos	10MB	5GB

Usuarios gratuitos estimados	Precio	Coste/Usuario extra
1000 usuarios	0,15€ por GB al mes	0,00015€/mes
500 usuarios	0,0221€ por GB al mes	0,00221€/mes

Es necesario alojar la información relativa tanto a los hallazgos de los usuarios, como a las zonas forestales. Teniendo en cuenta que en la primera ejecución de la aplicación, se descargará toda la información relativa a los hallazgos y las zonas forestales cercanas al usuario.
Esas transferencias también suponen un coste que conviene analizar.

Servicio	Descripción	Espacio	Descargas estimadas/Usuario	Peticiones gratuitas	Usuarios estimados gratuitos	Coste extra
Firebase Database	Descargar la información de las zonas forestales	756 MB	10 secciones al día | 1MB al día	50.000 al día | 10GB al mes	5000 al día | 10000 al mes	0.05€ por cada 100.000 peticiones al mes
Firebase Storage	Transferir las imágenes de los hallazgos	N/A	5MB	1GB al día	200 al día	0,10€/GB
Open Weather Map	Api con la información meteorológica histórica y actual	1MB	1 al día	60 al minuto	60 al minuto	35€/mes

Entonces, agrupando todos los costes, el presupuesto queda de la siguiente manera:

Concepto	Coste
Mano de obra	17.400€
Amortización de herramientas	90€
Software	1.051€
Infraestructura	1€ al mes por cada 1000 usuarios activos
Base imponible IVA 21% IRPF 7%	18.541€ 3.654€ -1.218€
Total	20.977€

Tabla 2.1: Presupuesto

2.6.2 Modelo de negocio

Una parte importante de todo proyecto sostenible, es diseñar una forma efectiva e indolora de monetizar el valor ofrecido.
Existen varios tipos de modelos de negocio estandarizados en el desarrollo de aplicaciones móviles. Desde la más sencilla, que consiste en poner un precio a la aplicación, hasta complicados sistemas de publicidad con tiempos de espera y funciones premium.

Desde el principio se ha tenido claro que en ningún momento se iba a enturbiar la experiencia de usuario con molestos anuncios. Además del poco dinero que produce.
La solución que se considera que mejor puede funcionar en el tipo de aplicación que se pretende desarrollar, es un modelo freemium no restrictivo. Es decir, ofrecer la aplicación totalmente gratuita con todas las funcionalidades disponibles, y ofrecer ventajas extra a un precio simbólico de 0.99€, del cual Google Play se quedará una comisión del 30%

La funcionalidad extra a la que se tendrá acceso, será la posibilidad de hacer una copia de seguridad de los hallazgos en la nube y sincronizarlos con todos sus dispositivos.

2.6.3 Rentabilidad

Teniendo en cuenta que, de media, el porcentaje de los usuarios activos que realizan compras en aplicaciones móviles no supera el 5% (Lupis (2017)), se calcula que la cantidad de usuarios necesaria para recuperar la inversión inicial de 20.977€, teniendo como fuente de ingreso la adquisición de cuentas premium (0.693€ brutos), asciende a 605.396 usuarios (30.270 ventas).

Una vez recuperada la inversión, se calcula que cada 1000 usuarios generará un gasto aproximado de 1€ y un beneficio estimado de 34.65€.

3 Antecedentes

3.1 Análisis de mercado

En el mercado de Google Play, a día de hoy, hay varias aplicaciones que implementan funcionalidades similares a las que pretendo desarrollar.
Y como ejercicio de investigación, se han analizado los principales proyectos. En orden de relevancia:

Figura 3.1: Página de Google Play de *"Identificador de setas"*

La aplicación más importante a día de hoy es "Identificador de setas", de desarrolladores alemanes, ha sido recientemente actualizada y cuenta con más de 1M de descargas, 5764 valoraciones y una nota media de 3,9/5
Implementa las siguientes funcionalidades:

- Identificador de setas basado en un live feed o en una imagen de la galería.
- Un calendario de setas
- Un mapa en el que anotar tus hallazgos
- Quiz educativo
- Escueto catálogo basado en párrafos de la wikipedia
- Modelo de negocio freemium 4,99€/año
  - Sin anuncios
  - Copia de seguridad de las anotaciones del mapa
  - Añadir manualmente posiciones de setas
  - Últimos niveles del quiz desbloqueados

Fortalezas	Debilidades
-Una red neuronal sólida que aporta predicciones precisas y rápidas. -Una base de usuarios amplia -El puesto de mejor aplicación	-Una vez hecha la predicción no se aporta más información que un icono determinando su comestibilidad y posibles confusiones. -La funcionalidad del mapa resulta poco útil, al menos a mi criterio.

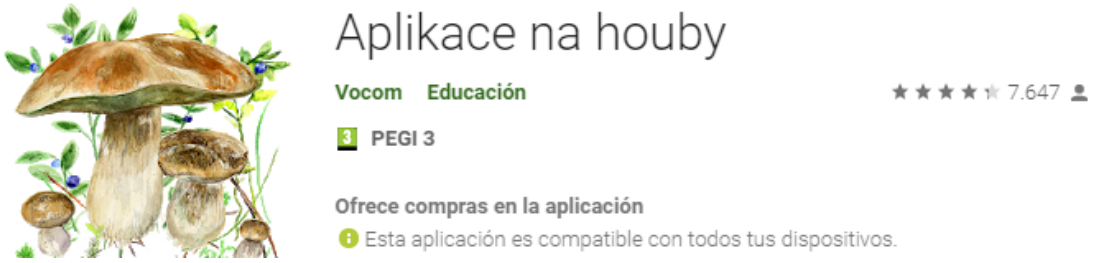


Figura 3.2: Página de Google Play de *"Aplikace na houby"*

Le sigue la aplicación Checa "Aplikace na houby", la cual dispone de 7647 reseñas, con una puntuación media de 4,4/5.

Implementa las siguientes funcionalidades:

- Identificador de setas
- Catálogo
- Sistema de filtrado del catálogo en base a características introducidas.

Fortalezas	Debilidades
-Una interfaz gráfica destacable.	-El modelo no es muy preciso. -La experiencia de usuario no es del todo satisfactoria.

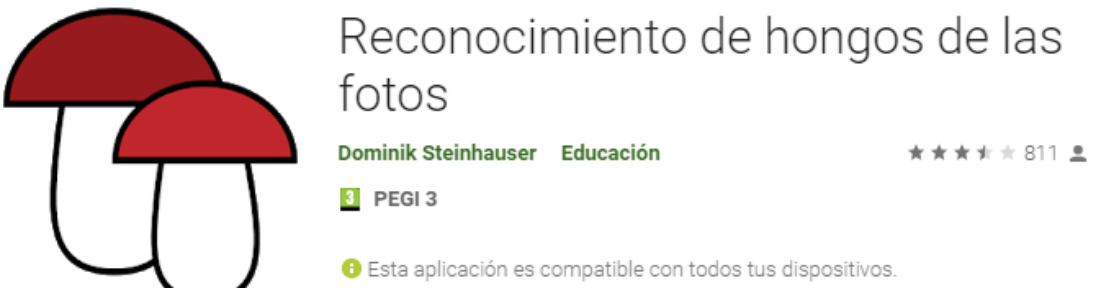


Figura 3.3: Página de Google Play de *"Reconocimiento de hongos de las fotos"*

La siguiente es "Reconocimiento de hongos de las fotos", la cual dispone de 811 reseñas, con una puntuación media de 3,6/5.
Únicamente implementa la funcionalidad de reconocimiento de setas.

Fortalezas	**Debilidades**
-Posibilidad de utilizar más de una foto para la inferencia del modelo.	-Modelo poco preciso. -Deficiente experiencia de usuario.

Figura 3.4: Página de Google Play de *"Snout"*

Finalmente, "Snout", una aplicación recién llegada al mercado con más de 100 descargas, pero sin reseñas.
Implementa las siguientes funcionalidades:

- Identificador de setas
- Catálogo

Fortalezas	**Debilidades**
-Una interfaz gráfica destacable. -Modelo preciso y preguntas adicionales (forma del sombrero, ...) para mejorar la precisión. -Experiencia de usuario agradable.	-No se da información adicional sobre la predicción.

3.2 Necesidades y oportunidades

Tras haber analizado las principales alternativas, se observa que la funcionalidad del clasificador de hongos está muy extendida en el mercado. Sin embargo, considero que hasta ahora ninguna de estas aplicaciones ofrece una experiencia de exploración y aprendizaje.
Es ahí donde veo una oportunidad y por lo que planteo siguientes posibles funcionalidades:

- Un clasificador de setas.
  - Predicción no solo basada en una foto, sino también en un *live feed* o en un cuestionario.
- Un mapa interactivo en el que se muestren capas de:
  - Los distintos tipos de bosques y zonas verdes.
  - Las setas que es probable que aparezcan en esa zona y su grado de probabilidad.
- Un catálogo centrado en el reconocimiento visual.
  - Mostrar claramente cuales son los rasgos característicos y diferenciadores de las posibles confusiones.
- Un sistema de interacción con el usuario.
  - Sistema de notificaciones que avise de altas probabilidades de aparición de setas en un radio de distancia.
- Recetas en base a la predicción o desde el catálogo.
- Un sistema para guardar los hallazgos, junto a un sistema de logros en base a la rareza de las setas.
- Una experiencia de usuario inmersiva y agradable.

4 Captura de requisitos

4.1 Introducción

En este capítulo se detallan los requisitos, los casos de uso y el modelo de dominio. Diseñados en base a la descripción del proyecto, las necesidades y los objetivos.

4.2 Requisitos funcionales

En este apartado, se expondrán los requisitos funcionales de la aplicación. Estos requisitos definen funciones del sistema o sus componentes. Estos requisitos son:

- La aplicación tiene que ser capaz de funcionar sin conexión a internet.
- La aplicación debe poder usarse sin registrarse, ni ningún otro requisito.
- El usuario deberá poder iniciar el proceso de identificación mediante una imagen de la cámara o de la galería.
- El usuario podrá elegir que zonas forestales mostrar en el mapa.
- El usuario podrá identificarse mediante su cuenta de Google.
- El usuario podrá ver los hallazgos que ha realizado.

4.3 Requisitos no funcionales

Los requisitos no funcionales definen requisitos que se deben cumplir siendo independientes de las funcionalidades que se van a implementar.

- El sistema debe ser estable.
- Se debe respetar las legislaciones europeas.
- Las interfaces y los procedimientos deberán ser comprensibles para todo el mundo.

4.4 Jerarquía de actores

La segmentación de tipos de usuarios es muy sencilla. ”Cualquiera” representa todas las funcionalidades que se pueden llevar a cabo. El ”usuario no identificado” es aquel que usa la aplicación sin estar identificado, pero tiene la posibilidad de registrarse. Una vez que decide identificarse, pasa a ser ”usuario identificado” o ”usuario premium”, en función de su condición. Los dos tienen la opción añadida de gestionar la cuenta. Y no hay ninguna diferencia entre los dos en cuanto a casos de uso se refiere. La única diferencia reside en la forma de llevar a cabo el subcaso de uso ”guardar hallazgo”.

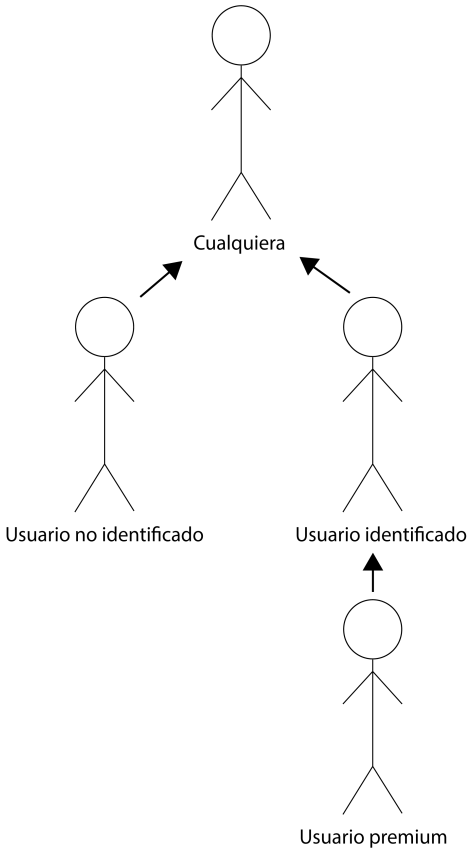


Figura 4.1: Jerarquía de actores

4.5 Casos de uso

La aplicación se puede usar sin necesidad de identificarse, por lo tanto, el actor "cualquiera" dispone de la mayor parte de los casos de uso:

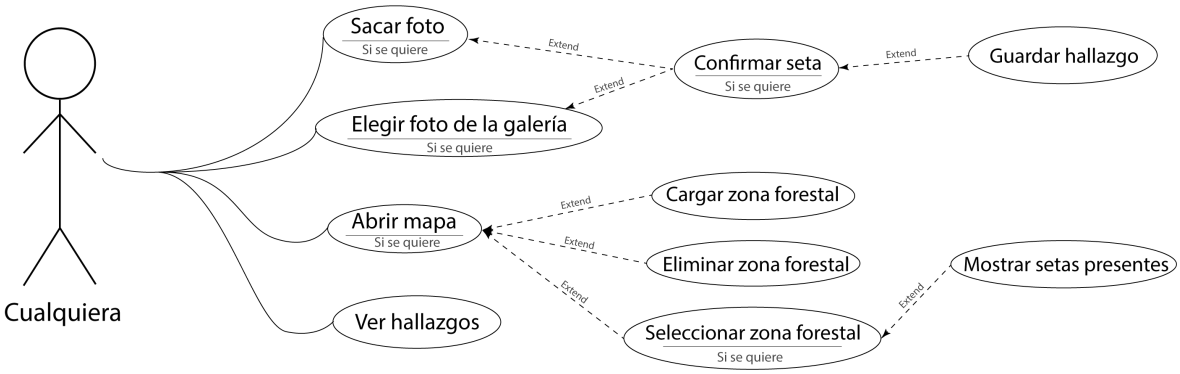


Figura 4.2: Casos de uso de cualquiera

Se observan las siguientes funcionalidades presentes a la pantalla principal:

- **Sacar foto:** Una de las dos maneras de comenzar el proceso de identificación de una seta.
- **Elegir foto de la galería:** La segunda manera de comenzar el proceso de identificación de una seta.
- **Abrir mapa:** Acción que inicia el despliegue del mapa interactivo con todas las diferentes zonas forestales.
- **Ver hallazgos:** Botón que desencadena la carga de la lista de las capturas realizadas por el usuario.

Por otra parte, se establece el único caso de uso de los usuarios no identificados.

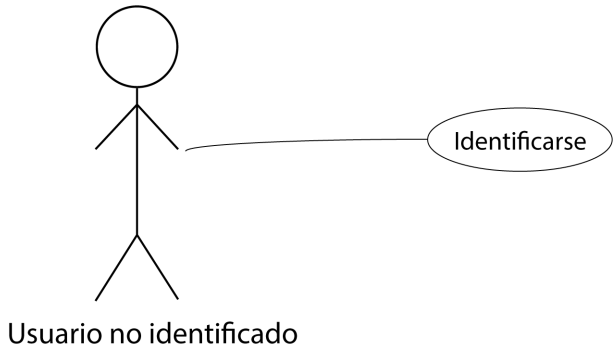


Figura 4.3: Casos de uso del usuario no identificado

El cual es, **identificarse** en el sistema, ya que hereda el resto de casos de uso de *cualquiera*. Una vez que el usuario se ha identificado, se cambia esa funcionalidad por la de gestionar la cuenta.

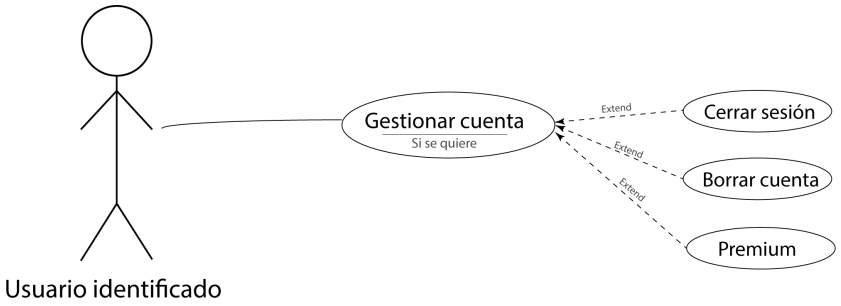
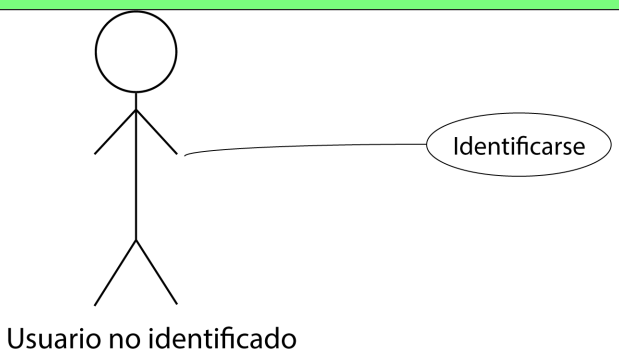
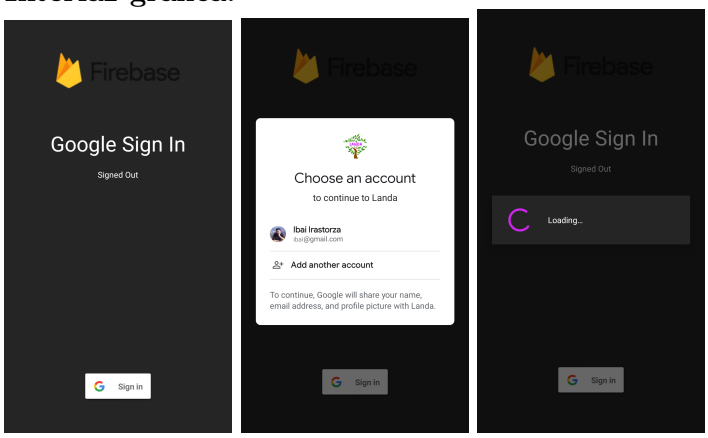


Figura 4.4: Casos de uso del usuario identificado

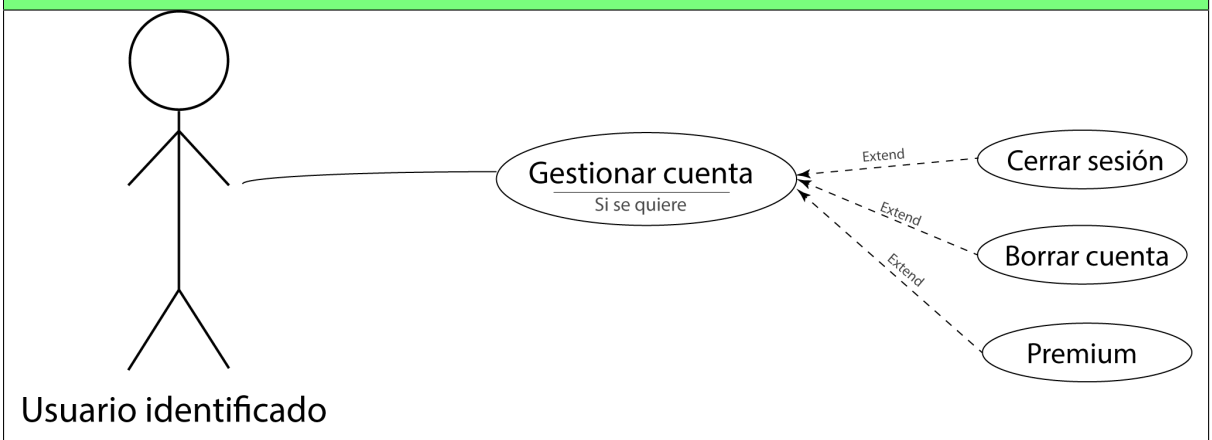
En el cual se puede cerrar sesión, borrar cuenta y adquirir las ventajas premium.
Por último, comentar que el actor premium, en cuestión de casos de uso, no se diferencia del identificado. Sus diferencias residen en procesos internos de la aplicación.

A continuación se detallan de manera extensa los casos de uso mencionados, organizados en función del actor que los interpreta.

4.5.1 Usuario no identificado

Identificarse

Descripción: El usuario se identifica en el sistema mediante su cuenta de Google
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario pulsa el botón ”identificarse” 2. El usuario selecciona la cuenta que desea utilizar del conjunto de cuentas registradas en el dispositivo 3. [Si el registro es correcto] 3a. Se accede al sistema [Si algo falla] 3b. Se muestra el correspondiente mensaje de error
Post condiciones: El usuario pasa a estar identificado, o no
Interfaz gráfica: 

4.5.2 Usuario identificado

Gestionar cuenta

Descripción: Pantalla en la que se muestra la información del usuario y se da opción a cerrar sesión, borrar la cuenta o adquirir la versión premium
Precondiciones: Ninguna
Requisitos no funcionales: Ninguno
Flujo de eventos: 1. El usuario pulsa el botón "Gestionar cuenta" situado en el lugar de "Identificarse" 2. Se le muestra la pantalla con toda la información 3. {Si el usuario pulsa "Cerrar sesión"} 3a. Se cierra la sesión y se devuelve a la pantalla principal 4. {Si el usuario pulsa "Borrar cuenta"} 4a. Se muestra una petición de confirmación 4b. [Si lo confirma] 4ba. Se procede a eliminar todos los registros del usuario en *Firebase* 4bb. [Si se ejecuta correctamente] 4bba. Se muestra un mensaje y se devuelve a la página principal 4bc. [Si algo falla] 4bca. Se muestra un mensaje de error detallando el motivo.

5. {Si el usuario pulsa ”Premium”}

5a. Se muestra la pasarela de pago ofrecida por el sistema de *Google Play billing*

5b. [Si el proceso se termina correctamente]

5ba. Se registra el cambio en *Firebase*

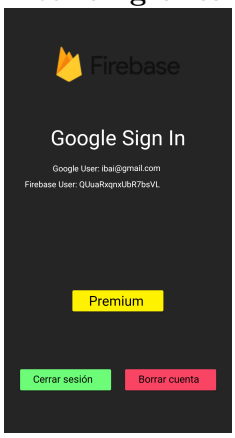
5bb. Se le muestra un mensaje de agradecimiento, pasa a ser ”Usuario premium” y se le devuelve a la página principal

5c. [Si el proceso falla]

5ca. Se le muestra un mensaje

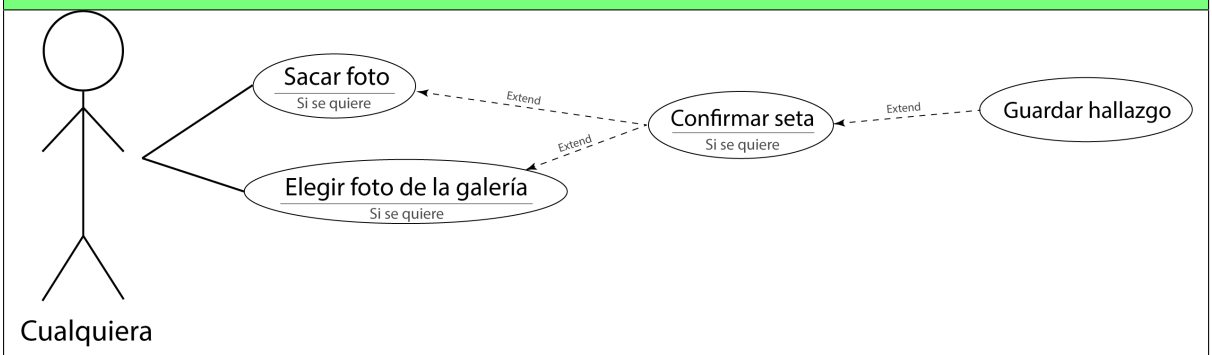
Post condiciones: No ocurre nada, o se ha cerrado la sesión y pasa a ser ”Usuario no identificado” o se ha borrado su cuenta en *Firebase* y pasa a ser ”Usuario no identificado”, o ha pasado a ser ”Usuario premium”.

Interfaz gráfica:



4.5.3 Cualquiera

Proceso de identificación de setas



Descripción: Proceso completo de identificación de una imagen, ya sea sacada de la cámara o de la galería.

Precondiciones: Ninguna

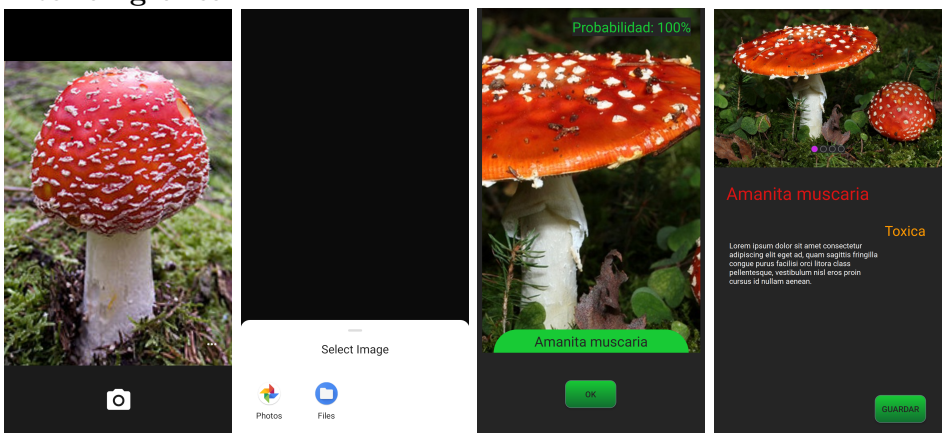
Requisitos no funcionales: Ninguno

Flujo de eventos:

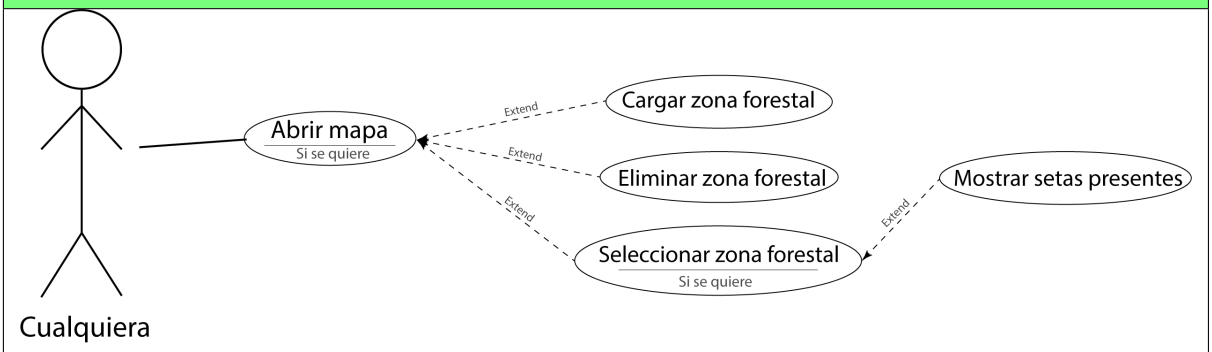
1. [El usuario inicia el proceso a través de la cámara]
   1a. El usuario abre la cámara y saca una foto a una seta
2. [El usuario inicia el proceso a través de una imagen de la galería]
   2a. El usuario abre la galería y selecciona una foto
3. Se analiza la imagen y se le muestra un carrousel con las posibilidades.
4. [El usuario confirma que alguna de las opciones se corresponde su foto]
   4a. Al usuario se le muestra la ficha completa de la seta.
   4b. {El usuario pulsa el botón ”guardar hallazgo”}
      4ba. Se guarda el hallazgo en la base de datos local
      4bb. [Si se trata de un usuario premium]
         4bba. Se guarda el hallazgo en *Firebase*
         4bbb. [Si ocurre algún error en la carga (No tiene conexión)]
            4bbba. Se le indica que se intentará subir más adelante
         4bbc. [Si se sube correctamente]
            4bbca. Se le manda al usuario a la página principal

Post condiciones: Se queda guardado el hallazgo, o no.

Interfaz gráfica:



Mapa



Descripción: Despliegue de un mapa con distintas capas e informaciones

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

Flujo de eventos:

1. El usuario pulsa el botón del mapa

2. Al usuario se le muestra un mapa con las capas que tenía seleccionadas cargadas, más los hallazgos

3. {El usuario despliega el menú y selecciona una zona forestal}

    3a. La zona forestal se carga

4. {El usuario despliega el menú y deselecciona una zona forestal}

   4a. La zona forestal se elimina

5. {El usuario selecciona una zona del mapa}

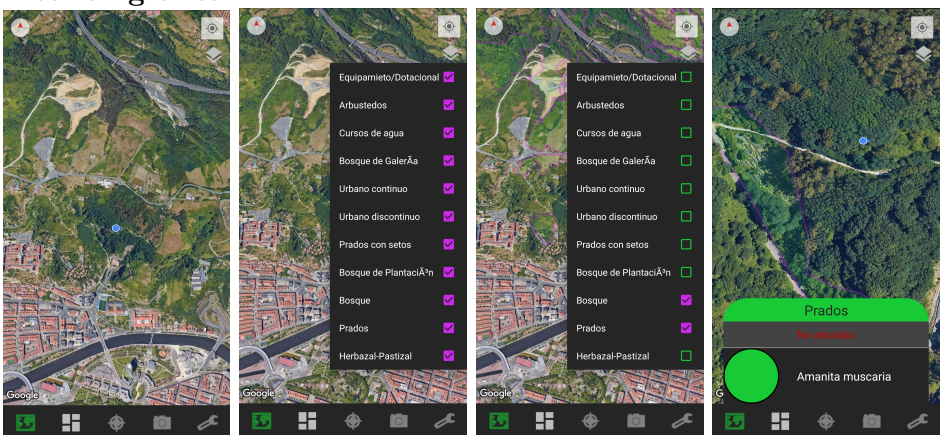
   5a. Se despliega un *Bottom sheet* con el tipo de zona forestal que es

   5b. {El usuario desliza el *bottom sheet* hacia arriba}

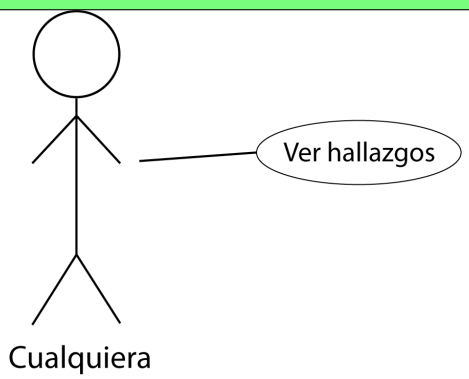
      5ba. Se le muestra la lista de setas que es probable que aparezcan ahí

Post condiciones: En caso de modificar la selección de las capas, se guarda como preferencias

Interfaz gráfica:



Ver hallazgos



Descripción: Mostrar la lista con los hallazgos guardados

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno

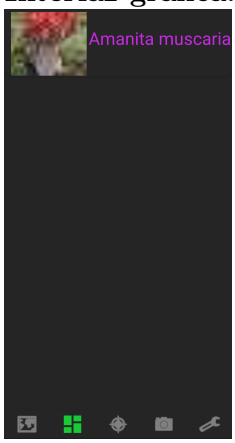
Flujo de eventos:

1. El usuario pulsa el botón de ”hallazgos”

2. Se le muestra una ventana con lista de tarjetas con las imágenes de cada hallazgo

Post condiciones: Ninguna

Interfaz gráfica:



4.6 Modelo de dominio

De los casos de uso diseñados se extrae el siguiente modelo de dominio.

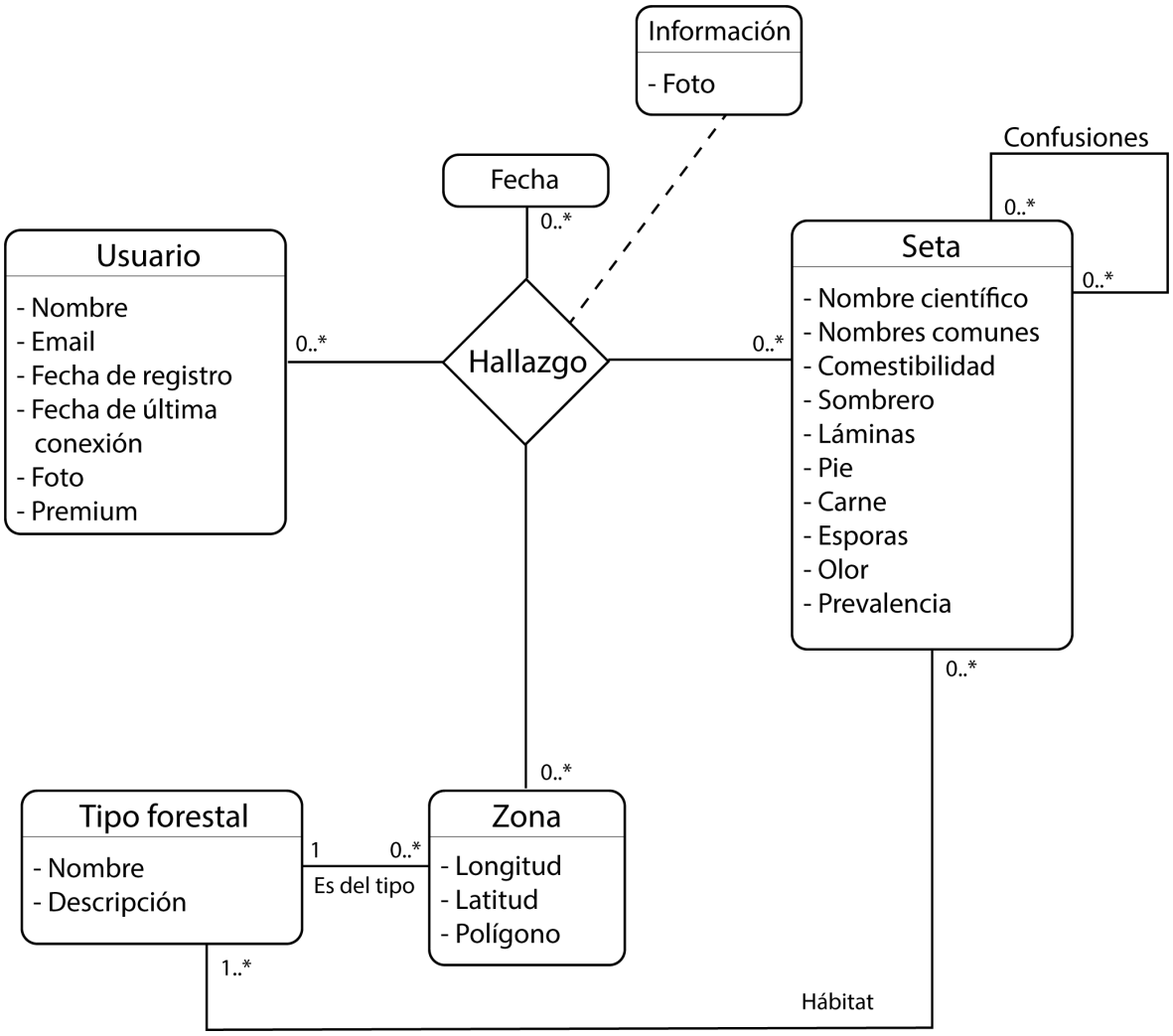


Figura 4.5: Modelo de dominio

Del cual se aprecian las siguientes entidades:

- **Usuario:** Recoge toda la información relativa a la gestión de usuarios del sistema. Tanto los datos de identificación, como un registro de las fechas de alta y última conexión. Además, se registrará en caso de haber apoyado económicamente el proyecto.

- **Seta:** Entidad responsable de almacenar toda la información correspondiente a las setas.
- **Zona:** Se corresponde con todos y cada uno de los polígonos del mapa. Caracterizados por las coordenadas geográficas y su tipo forestal.
- **Tipo forestal:** Hace referencia a las variedades de terreno y sus características. En caso de tratarse de un bosque, se indicará que arboles prevalecen.

Junto a las siguientes relaciones:

- **Hallazgo:** Relación múltiple que representa cada vez que un usuario realiza el proceso de identificación de una seta en una zona concreta y con una foto como muestra.
- **Confusiones:** Relación **M** a **N** entre setas que indica si alguna se confunde con otra.
- **Hábitat:** Relación **M** a **N** entre *seta* y *tipo forestal* que refleja las zonas en las que se desarrolla cada seta.
- **Es del tipo:** Relación que indica de que tipo forestal es cada zona.

El modelo de dominio es idéntico tanto para la aplicación local, como para la infraestructura de servidores. La única diferencia reside en que en local, los únicos datos de usuario que se van a almacenar, son los del propietario.

5 Análisis y diseño

5.1 Modelo de dominio a BD

A continuación se muestra el proceso de transformación del modelo de dominio diseñado, a una base de datos funcional. Para ello se declaran las entidades con sus atributos y se transforman las relaciones. El esquema de la base de datos resultante es la siguiente:

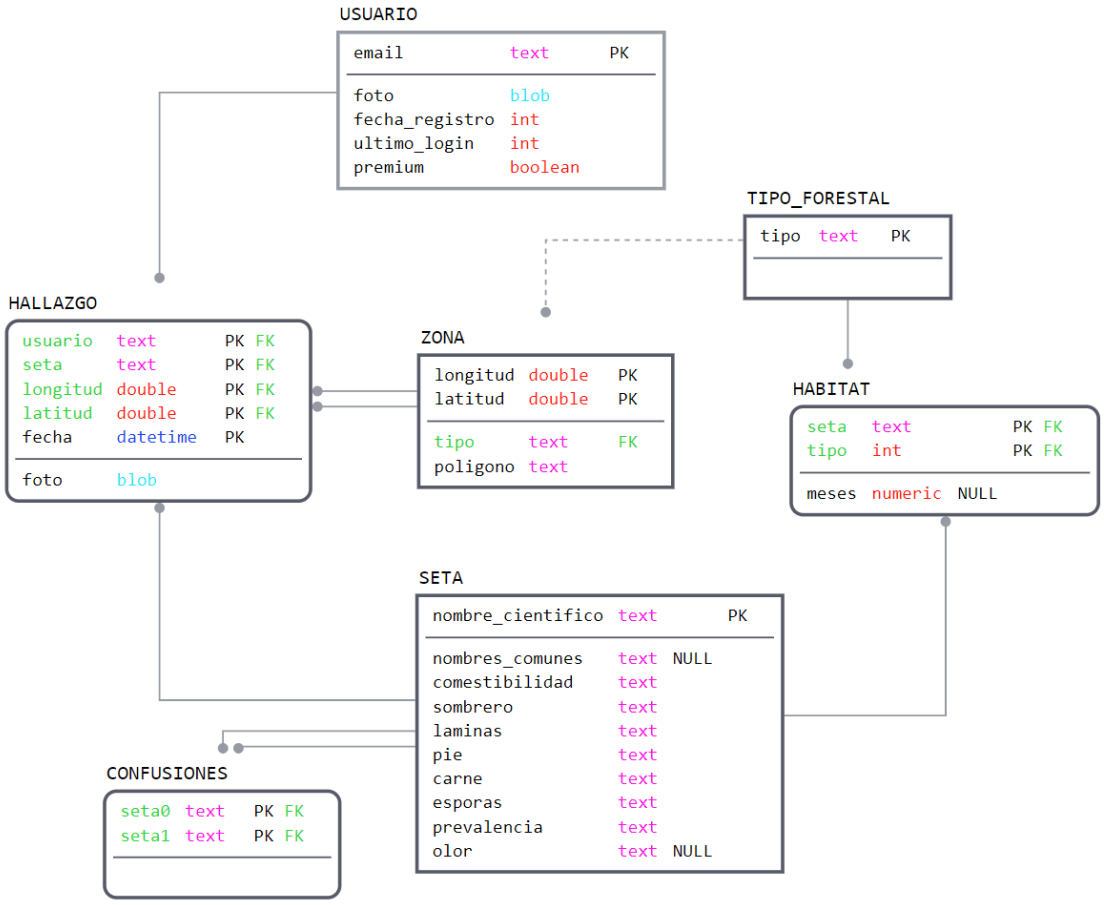


Figura 5.1: Modelo de base de datos

Tal y como se ha diseñado, se replicará la base de datos tanto en local como en el sistema de servidores. La única diferencia será que cada dispositivo únicamente almacenará la información del propietario, al contrario que en el sistema de servidores, donde se almacenará la información de todos los usuarios.

5.2 Diagrama de clases

De la misma manera que se ha definido la base de datos a partir del modelo de dominio, se ha diseñado el diagrama de clases. Teniendo también en cuenta los métodos utilizados en los diagramas de secuencia.

A continuación se muestra el diagrama de clases conceptual generado con las clases y los métodos más relevantes.

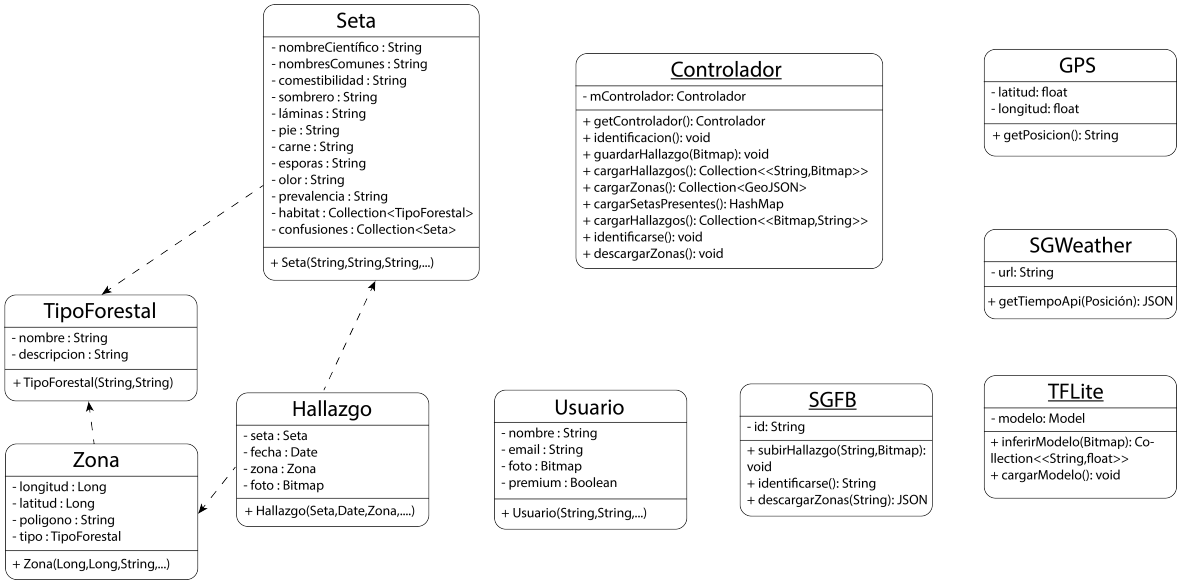


Figura 5.2: Diagrama de clases conceptual

5.3 Diagramas de secuencia

A continuación se muestran los diagramas de secuencia más relevantes de las distintas partes de la implementación.

5.3.1 Identificador de setas

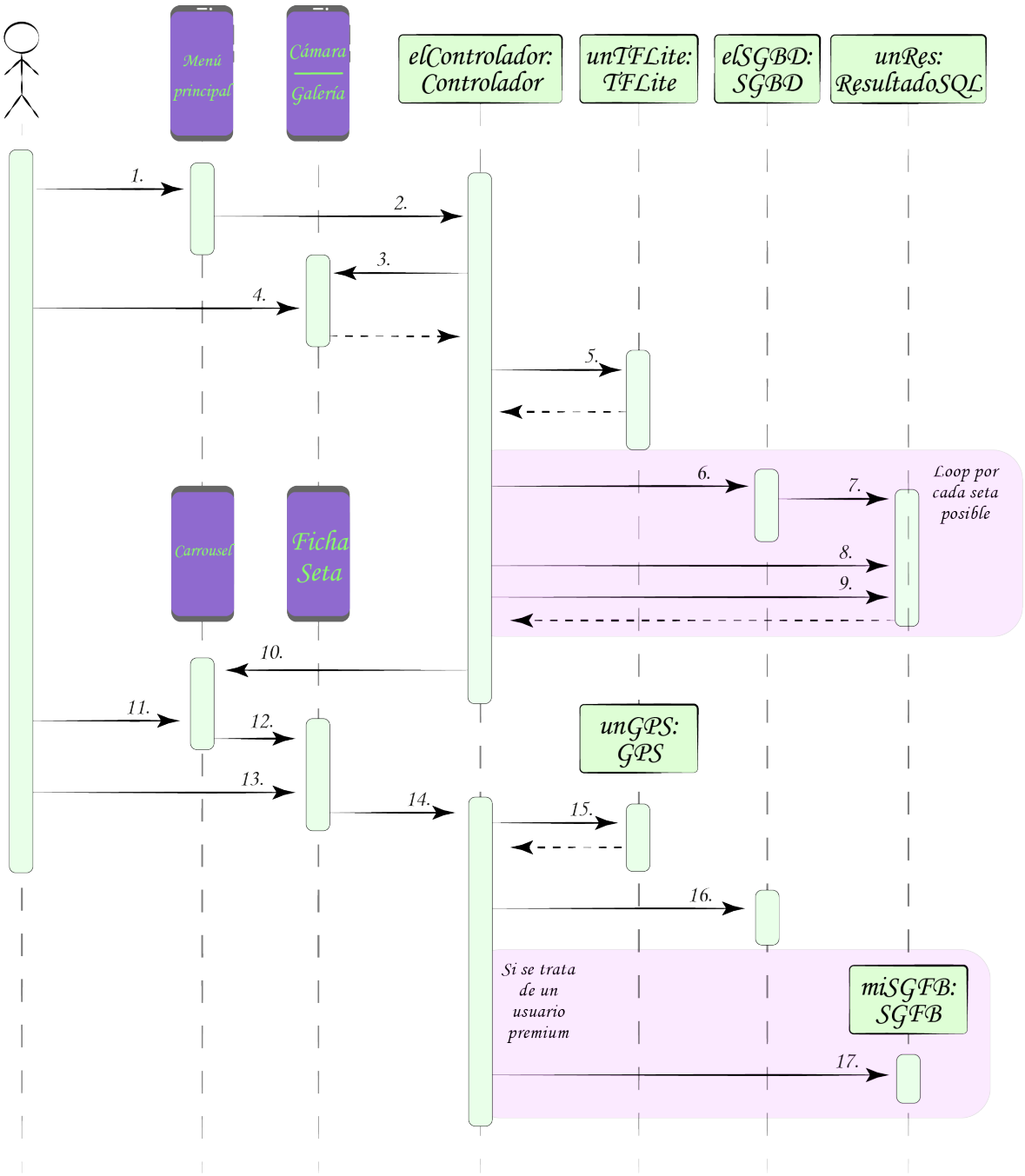


Figura 5.3: Diagrama de secuencia del proceso de identificación de setas

1. El usuario pulsa el botón de la cámara o de la galería
2. identificación(): *void*
3. newActivityForResult(): *Bitmap*
4. El usuario saca una foto o elige una foto de la galería
5. inferirModelo(*foto*): Collection< <String,float> >
6. *SELECT * FROM (Seta inner join Fotos on Seta.Nombre_cientifico=Fotos.seta) where seta=Nombre seta*
7. new ResultadoSql()
8. next()

[If true]

   9.1. getBitmap(”Foto”): Bitmap
   9.2. getString(”Comestibilidad”):String
   9.3. getString(”Sombrero”):String
   9.4. getString(”Pie”):String
   9.5. getString(”Carne”):String
   9.6. getString(”Olor”):String
   .....

10. new Carrousel(Collection< <String,float> >)
11. El usuario confirma de que seta se trata
12. new FichaSeta(Setas)
13. El usuario indica que desea guardar el hallazgo
14. guardarHallazgo(Foto): void
15. getPosicion(): String
16. *INSERT INTO Hallazgos Seta,Foto*
17. subirHallazgo(Seta,Foto)

5.3.2 Mapa

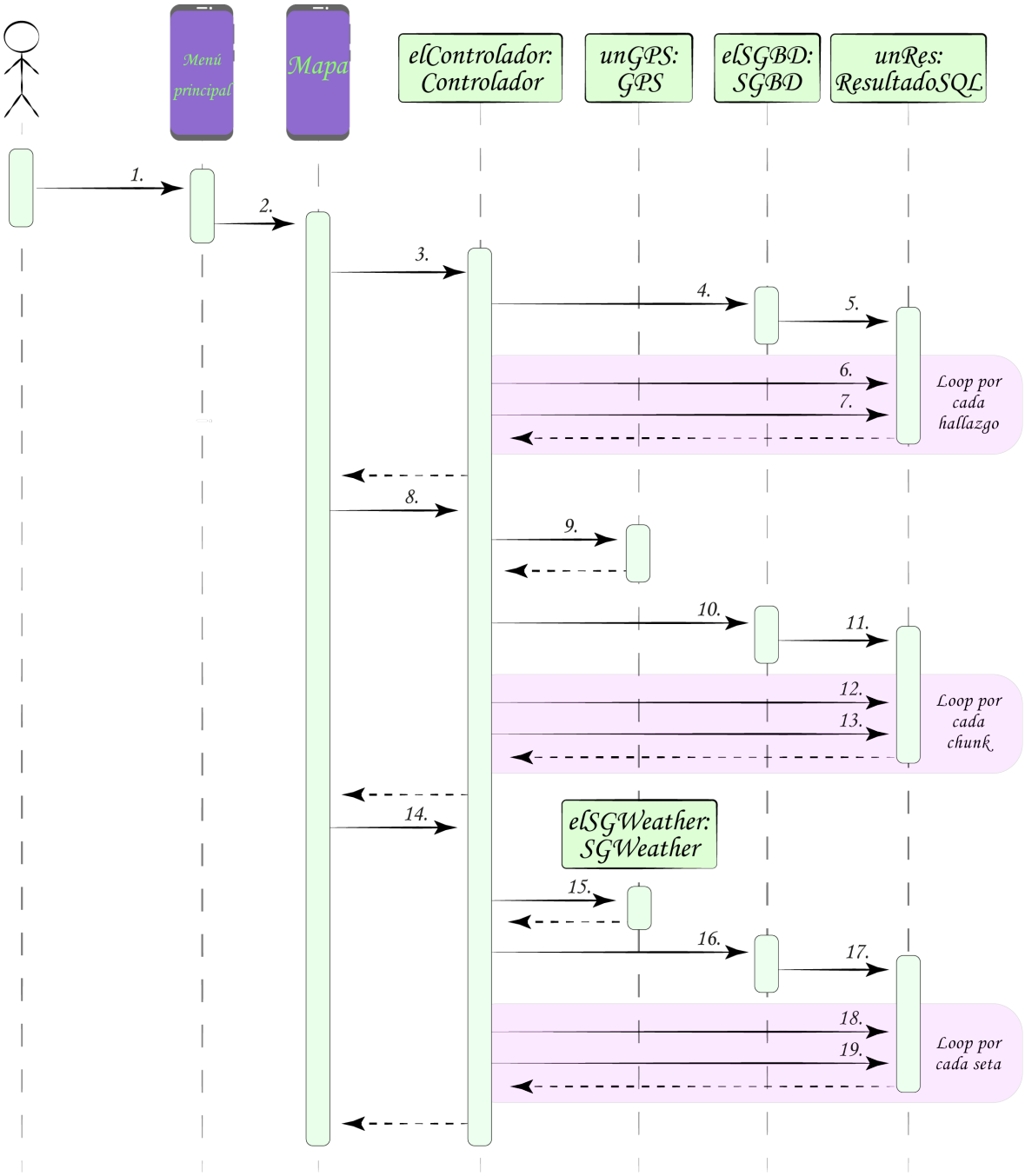


Figura 5.4: Diagrama de secuencia de la carga de los elementos del mapa

1. El usuario pulsa el botón del mapa
2. new Mapa()
3. cargarHallazgos(): Collection< <Seta,Foto> >
4. *SELECT * FROM Hallazgo*
5. new ResultadoSql()
6. next()

[If true]
   7.1. getString("Seta"): String
   7.2. getBitmap("Foto"): Bitmap

8. cargarZonas(): Collection<GeoJSON>
9. getPosicion(): String
10. *SELECT * FROM Zona where longitud<X and longitud>Y and latitud<Z and latitud>K*
11. new ResultadoSql()
12. next()

[If true]
   13. getString("Polígono"): String

14. cargarSetasPresentes(): HashMap
15. getTiempoApi(Posición): JSON
16. *SELECT Seta, Tipo FROM Habitat where meses include Este mes*
17. new ResultadoSql()
18. next()

[If true]
   19.1. getString("Seta"): String
   19.2. getString("Tipo"): String

5.3.3 Hallazgos

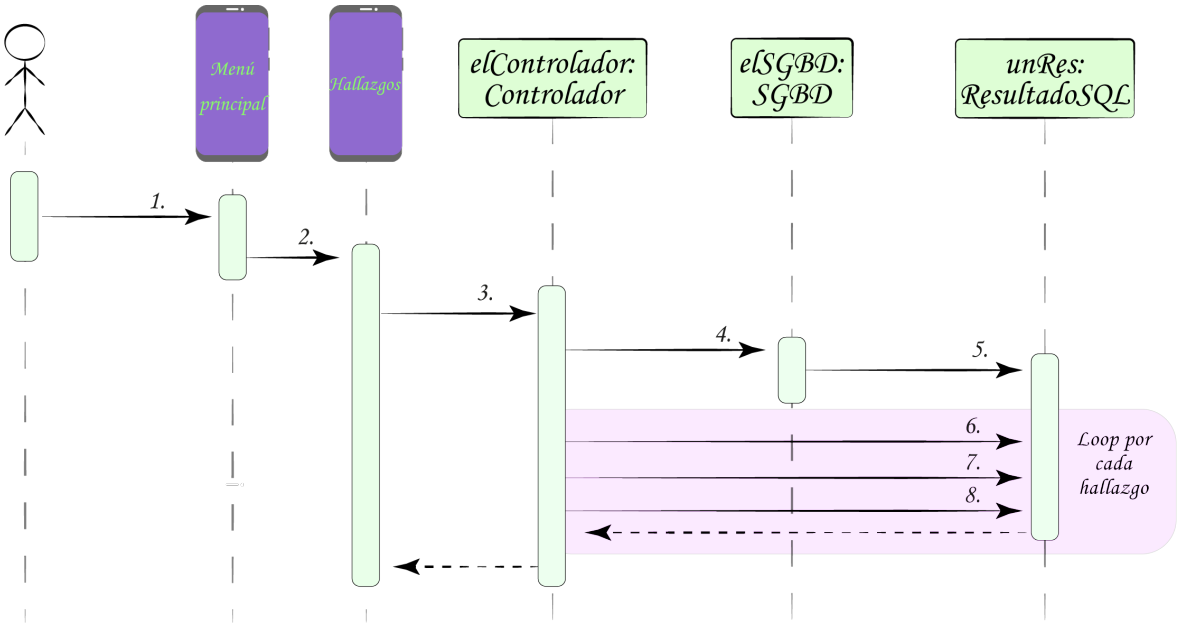


Figura 5.5: Diagrama de secuencia de la carga de la lista de hallazgos

1. El usuario pulsa el botón de la lista de hallazgos
2. new Hallazgos()
3. cargarHallazgos(): Collection< <Bitmap,String> >
4. *SELECT Nombre, Foto FROM Hallazgos*
5. new ResultadoSql()
6. next()

[If true]

   7. getBitmap(”Foto”): Bitmap
   8. getString(”Nombre”): String

5.3.4 Gestión de usuarios

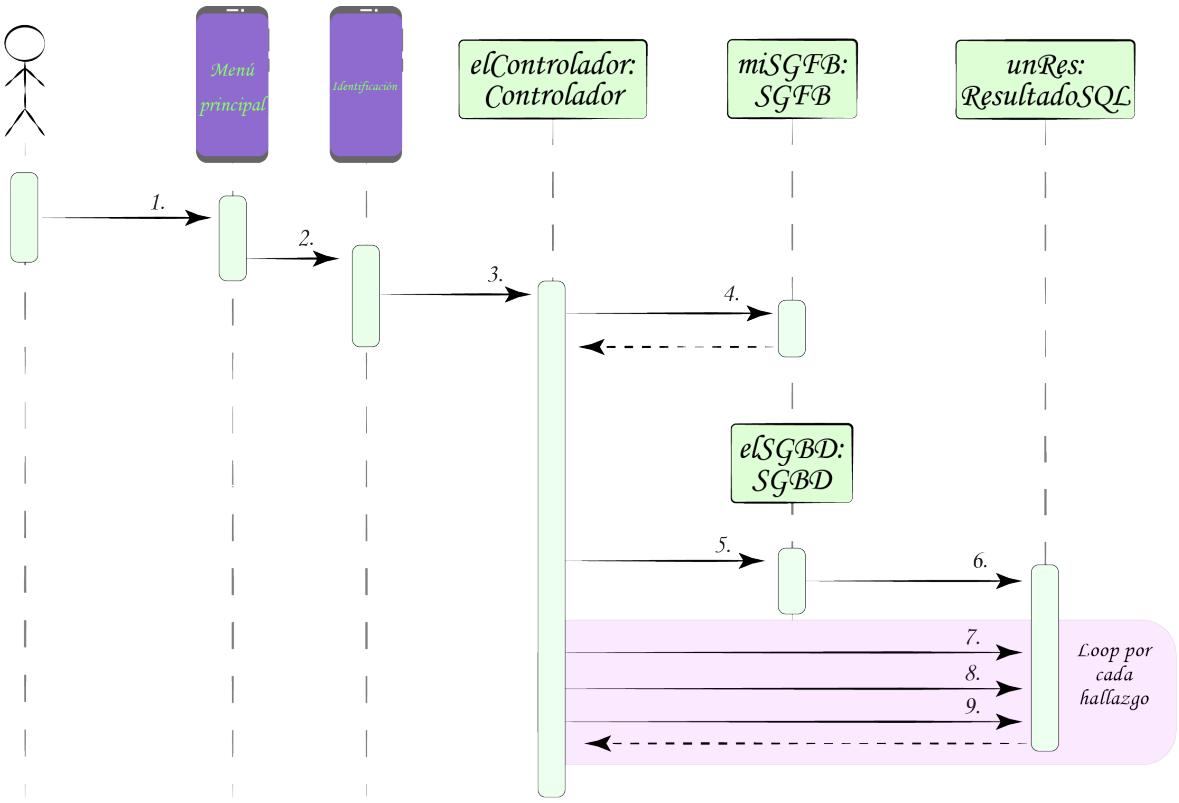


Figura 5.6: Diagrama de secuencia del proceso de identificación

1. El usuario pulsa el botón de identificarse
2. new Identificarse()
3. identificarse(): void
4. identificarse(): String

[If premium]

5. *SELECT Nombre, Foto FROM Hallazgos where usuario=usuario*
6. new ResultadoSql()
7. next()

[If true]

8. getBitmap(”Foto”): Bitmap
9. getString(”Nombre”): String

5.3.5 Gestión de datos

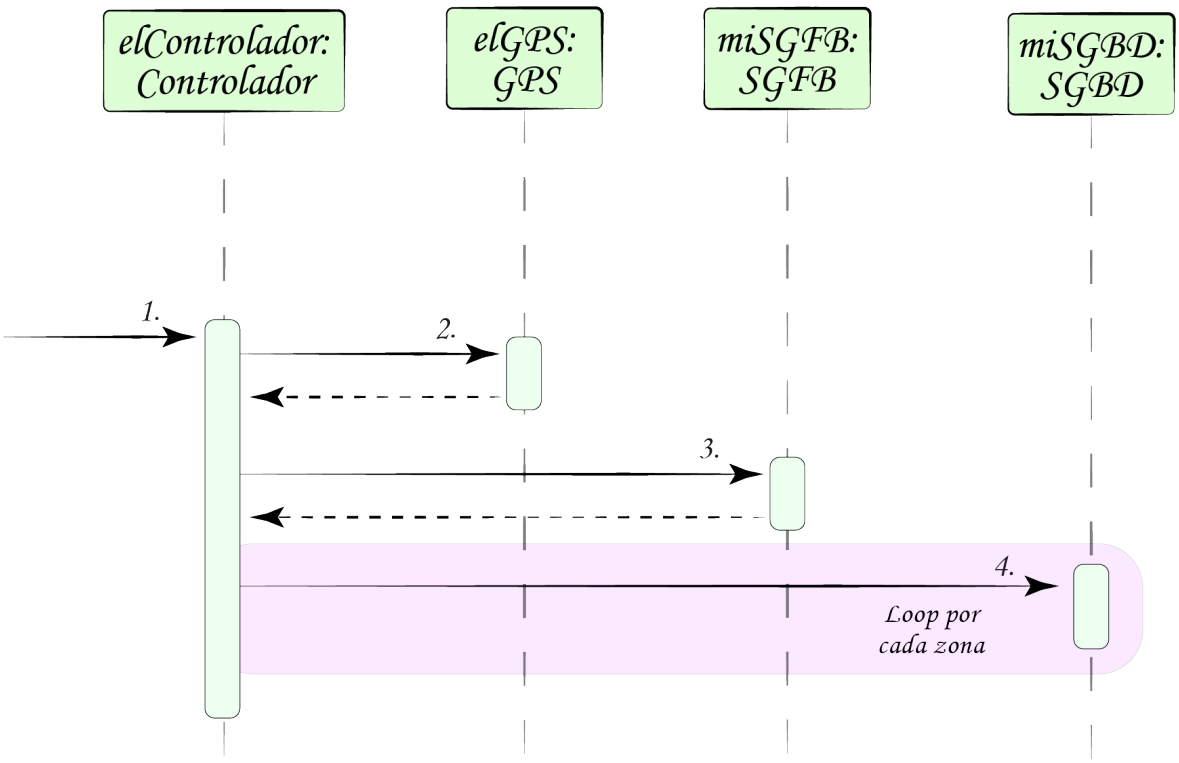


Figura 5.7: Diagrama de secuencia del proceso de descarga de datos

1. descargarZonas(): void
2. getPosicion: String
3. descargarZonas(Posición): *JSON_Zonas*
4. *INSERT INTO Zona (Longitud,Latitud,Tipo,Polígono)*

JSON_Zonas


```
[{
   "type": "FeatureCollection",
   "features": [
      {
      "type": "Feature",
      "geometry": {
         "type": "Polygon",
         "coordinates": [
         [
            [
            -2.50434,
            43.09487
            ],
            [
            -2.50436,
            43.09488
            ],
            [
            -2.50434,
            43.09487
            ]
         ]
         ]
      },
      "properties": {
         "PROV_MFE_1": "Bizkaia",
         "USOS_GENER": "Arbolado",
         "TIPESTR_DS": "Bosque de Plantación",
         "FORM_ARB_D": "Pinar de pino radiata",
         "TIPO_BOSQU": "Coníferas",
         "SP1_DS": "Pinus radiata",
         "O1": 100,
         "E1_DS": "Fustal",
         "SP2_DS": "sin datos",
         "O2": 0,
         "E2_DS": "Sin datos",
         "SP3_DS": "sin datos",
         "O3": 0,
         "E3_DS": "Sin datos",
         "FCCTOT": 100,
         "FCCARB": 0,
         "DISTRIB_DS": "Uniforme",
         "SUPERFICIE": 0.28165488
      }
      }
   ]
   }
,{...}
,{...}
]
```


Esta estructura de datos alberga la información de cada zona forestal de España

6 Desarrollo

La ambición por llenar esta aplicación de funcionalidades y ofrecer una experiencia de usuario memorable, ha provocado un aumento considerable en la complejidad de la implementación. Por lo tanto, el desarrollo consta de muchas y laboriosas fases. En orden cronológico:

- Construir y entrenar la red neuronal capaz de discernir entre el máximo número de setas con la mayor precisión posible.
- Construir la aplicación movil.
- Diseñar e implementar la usabilidad y experiencia de usuario.
- Hacer uso de plataformas de distribución para desplegar la aplicación.

6.1 Machine Learning

La funcionalidad principal de esta aplicación consiste en un clasificador de imágenes especializado en hongos capaz de reconocer la mayor cantidad de especies, lo más precisamente posible.

Para ello es necesario seguir unas pautas establecidas en todo proceso de desarrollo de inteligencia artificial.

1. Identificar el problema y elegir una aproximación adecuada.
2. Construir el dataset.
3. Elaborar el algoritmo.
4. Entrenar el modelo.
5. Evaluar el modelo.
6. Desplegar el modelo.

A continuación se detallan todos lo procesos realizados para hacer esto posible.

6.1.1 Aproximación

Se trata de un problema de clasificación de imágenes, y tras hacer una investigación inicial, quedó claro que a día de hoy, no hay mejor tecnología que las redes neuronales convolucionales para la tarea en cuestión.
En cuanto a las herramientas a utilizar, tampoco cabe duda que la mejor opción, y la más extendida son las librerías de *Keras* y *TensorFlow*. Pero además, se hace imprescindible usar esas dos, ya que son las únicas que generan modelos compatibles con la ejecución en dispositivos móviles, gracias a la librería TFLite para Android.

6.1.2 Preproceso

La primera fase, y sin duda la más importante, es la del preproceso. Consiste en recolectar el catálogo de fotos, limpiarlo, categorizarlo y certificar su integridad.
Antes de abordar el problema fue necesario decidir con que cantidad de setas se iba a trabajar, y que especies se iban a incluir. Puede parecer una decisión fácil: "Todas las que encuentres". Pero como ya se descubrirá más adelante, cuantas más clases tenga un modelo, menos preciso será, y con cuantas más imágenes se entrene un modelo, más tardará el entrenamiento y la inferencia.
Es necesario encontrar el equilibrio entre el tamaño del dataset, el número de clases y el tiempo de entrenamiento.

Construcción del dataset

Una primera versión del dataset constaba de las 500 setas más comunes dispuestas en la Fungipedia. Pero una vez evaluado el modelo resultante, era evidente la falta de precisión y la cantidad de nombres que hacían referencia a setas minúsculas o extrañas.
Había que elegir más cuidadosamente las especies a estudiar, teniendo en cuenta cuales son más probables que los usuarios se encuentren y quieran analizar.

Así que una segunda versión del dataset consistía en las 100 setas más comunes de Bizkaia. Esta vez, con un tamaño más manejable, y asegurándome de que todas y cada una fuesen relevantes, procedí al siguiente paso.
Una vez conseguida la lista de nombres, el siguiente paso fue desarrollar un script para descargar imágenes de cada especie. Para ello se usaron librerías de web scrapping como BeautifulSoup.
Sin embargo, aquí surge el primer gran reto del proyecto. Dado el aspecto comercial de la aplicación, es imprescindible contar con las licencias adecuadas para todo el material externo que se utilice. Es decir, es ilegal usar cualquier imagen de internet para entrenar un modelo o mostrarlas en la aplicación. Existen leyes de propiedad intelectual y derechos de autor que protegen a creadores de usos indebidos de sus productos.

Figura 6.1: Referencias

Se debía buscar una solución, y había varias alternativas encima de la mesa. Desde pagar las licencias de las imágenes, hasta sacar fotos yo de todas y cada una de las setas. Afortunadamente, adentrándose todos los rincones de internet, se encontraron varios bancos de imágenes públicos alojados en prestigiosas webs como Kaggle, Mushroom world, Mushroom observer y los útiles filtros de *Creative Commons* de Google Imágenes y Bing Imágenes. Todas estas, por supuesto, con la condición de referenciar adecuadamente al autor.

Por lo tanto, se recolectaron imágenes y metadatos de todas las webs mencionadas, e intenté fotografiar la mayor cantidad de setas posible. Consiguiendo así un dataset variado y legítimo.
En total se consiguieron alrededor de 150 imágenes por cada una de las 100 especies. Con un tamaño medio de 1MB por archivo, el conjunto asciende a 15GB de datos.

Tratamiento de los datos

El siguiente paso consistió en organizar las imágenes de cada especie en carpetas, asegurarse se tener aproximadamente la misma cantidad de fotografías para cada clase y eliminar los archivos corruptos, borrosos, sobreexpuestos, subexpuestos, mal encuadrados e incorrectamente clasificados.

Más adelante en el proyecto, y como resultado de innumerables pruebas e investigaciones, con el fin de mejorar la precisión y capacidad del modelo para adaptarse a condiciones adversas de imagenes (mal iluminadas, mal enfocadas, mal encuadradas, ...), se desarrolló un algoritmo de resampling y variación.(Luis Perez, 2017)

El algoritmo consistía en, por cada imagen, crear copias variando la exposición (brillo), el enfoque, el encuadre y la orientación. De esta manera se consiguió disponer de más imágenes para entrenar y enseñar al modelo a adaptarse a distintas circunstancias.
Para conseguirlo, se aplicaron operaciones matriciales a los valores RGB de los pixeles:

- **Desenfoque:** Combinación lineal de la matríz de pixeles con un kernel de distribución Gaussiana con *alpha*=3
- **Temperatura del color:** Manipulación de los vectores RGB
- **Brillo:** Incrementar o decrementar el valor de cada pixel por una constante
- **Encuadre:** Extraer una submatriz de menores dimensiones

Figura 6.2: *Data augmentation*

6.1.3 Modelado

En esta segunda fase, en base al dataset obtenido, se va a construir el clasificador de imágenes, diseñando la arquitectura de la red y optimizando los hiperparámetros (*Fine Tuning*) hasta conseguir un modelo lo suficientemente preciso y ligero.

Estructura de la red

Un enfoque clásico para abordar este problema sería diseñar una red neuronal con varias capas de *Pooling* y convolución, con funciones de activación *Relu*, y entrenarlo con un porcentaje del dataset. Sin embargo, existe un nuevo paradigma para desarrollar modelos: ***Transfer Learning***.(Maithra Raghu, 2019)

El método de *Transfer Learning* consiste en utilizar un modelo preentrenado con colosales datasets y capacidad de computo infinita (normalmente ofrecidos por Google o alguna universidad), y reentrenarlo con tu propio conjunto de datos y tus clases, logrando así un modelo muy preciso, entrenado en una fracción de tiempo en comparación con lo que costaría hacerlo desde cero.

Por lo tanto, sin dudarlo demasiado, elegí el modelo preentrenado más sofisticado hasta el momento: **InceptionV3** (basado en la base de datos de *ImageNet*), y me puse a implementar los algoritmos.
No tuvo que pasar mucho tiempo para que me diese cuenta de la complejidad de la tecnología y de lo mucho que me iba a costar llevarlo a la práctica.
Afortunadamente Google vino a salvar el día. En la sección de ejemplos de la librería *Tensorflow*, tiene a disposición de cualquiera un completísimo y bastante complejo script para hacer exactamente eso: **retrain.py**

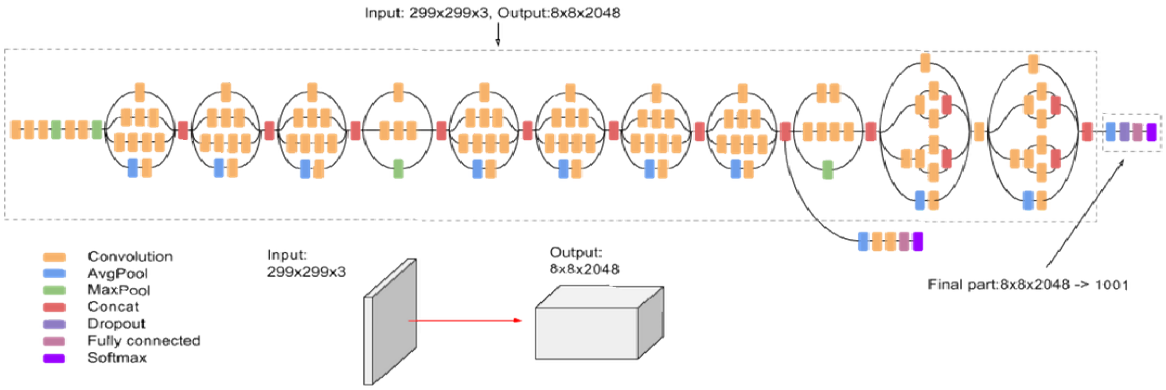


Figura 6.3: Arquitectura de InceptionV3

Por lo tanto, se va a utilizar la estructura de red preestablecida por *InceptionV3*, la cual consiste en un input de 299x299x3 (imágenes RGB cuadradas de 299^2 pixeles), 159 capas, 23.851.784 parámetros, un *accuracy* máximo teórico de 0,937 y un tamaño constante de 93MB.

Por último, mencionar que el proceso de despliegue del modelo en la aplicación movil, requiere comprimir la red y adaptarla al formato *TFLite* (*FlatBuffer* optimizado).

Entrenamiento

Una vez conseguidos el dataset y el algoritmo, el siguiente paso lógico es empezar a entrenar.
En los primeros intentos, traté de ejecutar el script en el ordenador de sobremesa, pero en vista del tiempo y la capacidad de computo que iba a necesitar, decidí trasladar la operación a un servidor. Sin embargo, los servicios de alojamiento que ofrecen empresas como Amazon, Google o Microsoft, limitan sus productos gratuitos a máquinas virtuales con muy poca potencia y espacio. Así que decidí recolectar y reciclar piezas de hardware para montar un servidor en casa. De esta manera podría entrenar de manera indefinida sin ningún coste, e incluso disponer de capacidad para alojar parte del *backend* de la aplicación.

Por lo tanto, migré los archivos y comencé a entrenar con las opciones por defecto establecidas en el script.
Las primeras pruebas fueron con el dataset inicial de 500 especies, y el entrenamiento duró 154 horas. Sin embargo, al evaluar el modelo con imágenes nuevas, la precisión fue catastrófica.

Pasaron meses intentando obtener un modelo decente a base de aumentar el tiempo de entrenamiento (llegando a durar 32 días), probar distintas combinaciones de hiperparámetros e ir variando la cantidad de imágenes por especie. Pero no fue suficiente, había que reducir el número de especies y deshacerse de todas las clases que no aportasen utilidad al modelo.

Lo siguiente que se hizo fue revisar manualmente cada clase y eliminar las especies que sería improbable encontrarse, tenían un aspecto extraño, o simplemente no tenían ningún valor. Este proceso resultó en la reducción a 374 clases. Y esta vez, al entrenar el modelo, los resultados fueron satisfactorios.
Tras una ejecución de 340 horas con diez *epoch*, una evaluación *hold out* (70/30) determinó el resultado de la precisión en un 57%. Que en el momento de probarlo en situaciones reales daba predicciones bastante acertadas y con una confianza de más del 90% en casos contados. Algo aceptable, pero ni mucho menos ideal.

```
Epoch 9/10
1145/1145   [==============================] 63312s 56us/step - loss: 1.2684 - acc: 0.5720 - val_loss: 1.3648 - val_ac
c: 0.5618
Epoch 10/10
1145/1145   [==============================] 63454s 56us/step - loss: 1.1152 - acc: 0.5743 - val_loss: 1.1158 - val_ac
c: 0.5769
1145/2549 [==============================] - 33454s 56us/step
[INFO] accuracy: 57.43%
[INFO] Loss: 1.5719172541517587
Time: 340:26:26.2268195
```

Figura 6.4: Resultados evaluación *Hold out*

Más adelante en el proyecto, tras haber construido el dataset con 100 especies, se reanudaron los esfuerzos.
Esta vez, el tiempo de entrenamiento disminuyó drásticamente, y la precisión del modelo resultante aumentó aún más.

Finalmente, el entrenamiento consistió en 50 *epoch* durante 23 días, para alcanzar un 66% de accuracy. Y por lo tanto, se deduce que en el futuro, para mejorar la precisión, habría que aumentar el número de *epoch* y la cantidad de imágenes por cada clase. Lo cual se podría calcular mediante técnicas de *Power analysis*.

Figura 6.5: Gráfica del entrenamiento

Entonces, se disponen de dos modelos decentes, uno con 374 clases y una precisión del 57%, y otro con 100 clases y una precisión del 66%. Sin embargo, el primero al ser más complejo, tarda unos segundos más en hacer la inferencia. Así que para esta primera versión de la aplicación se va a desplegar la segunda red neuronal, siendo más manejable y ligera, con sus 100 clases y un 66% de precisión.

6.1.4 Evaluación

Ese modelo inicial obtenido se evaluó de distintas maneras. Por una parte, se le sometió a una validación *K-Fold cross validation*, y por otra, se distribuyó a un conjunto de personas para que probasen con imágenes reales la calidad del clasificador.

KFold cross validation

Este procedimiento consiste en dividir el dataset en diez(K) partes iguales y entrenar con el 90% del dataset y evaluarlo con el 10% restante. Repitiendo el proceso diez veces, variando el subconjunto a evaluar.
Se ha utilizado este método de evaluación por ofrecer unos resultados más realistas, al hacer la media de diez iteraciones de entrenamiento distintas.
Estos son los resultados del proceso:

KFold cross validation		
Fold	**Loss**	**Accuracy**
1	2.4094	67.90%
2	1.7682	67.37%
3	2.4695	66.41%
4	2.3637	66.28%
5	2.0837	65.51%
6	2.2160	66.76%
7	1.7227	67.25%
8	2.3571	65.54%
9	1.5531	66.03%
10	2.4262	66.98%
Media	2.1370	**66.60%**

Se deduce una precisión media del 66,6% con una variación de 0,7 y un error de 2,13.
Son datos aceptables, pero con mucha capacidad de mejora. Para la primera versión de la aplicación, servirá.

TelegramBot

Por otra parte, para probar el clasificador en situaciones reales, se utilizó el servidor construido, para alojar un *Bot* de *Telegram*, el cual recibía imágenes, infería el modelo y respondía con las tres estimaciones con mayor probabilidad.
El servicio fue un éxito. Mucha gente se implicó y se recibieron bastantes imágenes (no siempre de setas...).

Se concluyó que reconocía bien la mayoría de las setas, pero tenía problemas con especies visualmente similares. No era difícil encontrarse con una predicción acertada con una confianza superior al 95%

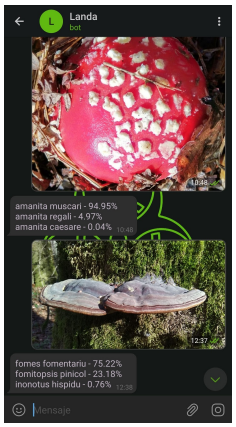


Figura 6.6

6.1.5 Conclusiones

En primer lugar, el dataset obtenido tiene la variedad justa como para ser lo suficientemente util y no complicar demasiado el desarrollo de esta primera versión. Pero no cabe duda que habrá que ampliarlo con el tiempo.

Por otra parte, el uso de tecnologías como *Transfer Learning*, y los scripts proporcionados por *Tensorflow* han facilitado enormemente la construcción de la red neuronal convolucional. Cuyo resultado, a pesar de no ser perfecto, es capaz de discernir lo suficientemente bien entre especies.

Dicho esto, conviene contextualizar el uso que se le va a dar. En ningún momento se pretende ser totalmente exacto. Simplemente se quiere ofrecer una herramienta más en la ayuda a la identificación de hongos.
Es imprescindible la implicación de los usuarios para determinar cual de las setas que le sugiere el modelo es la que tienen delante, mostrando sus características clave, y nunca dependiendo unicamente del aspecto.

6.2 Desarrollo de la aplicación

En este apartado se detalla el proceso seguido para construir la aplicación y su correspondiente infraestructura.
El desarrollo se va a dividir en distintas fases, correspondiéndose con cada una de las funcionalidades. Las cuales son:

- Todo el proceso de identificación de setas.
- El mapa interactivo con carga dinámica de capas con las zonas forestales y las setas que en ellas aparecen.
- Gestión y listado de hallazgos.
- Autenticación y gestión de usuarios.

6.2.1 Identificación de setas

Esta crucial funcionalidad requiere una dedicación especial. El éxito de la aplicación depende de la calidad y el diseño del proceso de identificación de setas.
El procedimiento se divide en los siguientes pasos:

- Vista de cámara para tomar una foto de la seta o elegir una foto de la galería .
- Inferencia del modelo sobre la imagen.
- Carrusel de las setas con mayor probabilidad.
- Selección y muestra de la ficha completa.

Fotografía

Dos maneras de iniciar el proceso. Sacando una foto a través de la cámara o eligiendo una foto de la galería.
Para realizar una fotografía, se usa un *Intent implícito*, que lanza la aplicación nativa de la cámara, toma la imagen y la devuelve.

```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
}
```

En próximas versiones se pretende implementar la *API* de *CameraX* y tener una vista personalizada de la cámara en la misma aplicación.
Por otra parte, para elegir una foto de la galería, también se utiliza un *Intent implícito* que esta vez lanza una vista en la que se muestra un sistema de archivos con las imágenes recientes y la posibilidad de acceder a servicios externos como *Google Photos* o *Google Drive*.

Inferencia

Una vez obtenida la imagen a analizar, es momento de inferir la red neuronal. Para ello se dispone de la librería *TFLite*

```
implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'
```

Antes de nada, antes incluso de obtener la fotografía, conviene cargar en memoria el modelo en segundo plano, ya que dependiendo de la potencia del dispositivo, puede tardar varios segundos.

```
tfliteModel = FileUtil.loadMappedFile(context, "Modelo.tflite");
```

En cuanto se recibe la imagen, se inicia el proceso de inferencia.

```
public List<Recognition> inferencia(Bitmap imageBitmap) {
    TensorImage tensorImage = loadImage(imageBitmap, 90);
    int probabilityTensorIndex = 0;
    int[] probabilityShape = tflite.getOutputTensor(probabilityTensorIndex).shape();
    DataType probabilityDataType = ↩
        ↪ tflite.getOutputTensor(probabilityTensorIndex).dataType();
    TensorBuffer outputProbabilityBuffer = ↩
        ↪ TensorBuffer.createFixedSize(probabilityShape, probabilityDataType);

    tflite.run(tensorImage.getBuffer(), ↩
        ↪ outputProbabilityBuffer.getBuffer().rewind());

    try {
        labels=FileUtil.loadLabels(context, "clases.txt");
    } catch (IOException e) {
        e.printStackTrace();
    }
    probabilityProcessor=new TensorProcessor.Builder().build();
    Map<String, Float> labeledProbability =
            new TensorLabel(labels, ↩
                ↪ probabilityProcessor.process(outputProbabilityBuffer))
                    .getMapWithFloatValue();
    return getTopKProbability(labeledProbability);
}
```

Al finalizar la inferencia, se recibe una lista de setas junto a su probabilidad, y se recuperan las tres más probables.

Carrousel

Con las tres setas con mayor probabilidad, se carga un carrousel con tres imágenes por cada especie, un indicador de la probabilidad y una pestaña inferior desplegable con el nombre y sus características clave.
Es en este momento cuando el usuario tiene que comprobar si alguna de las setas mostradas se corresponde con la que intenta identificar.
Cuando comprueba que efectivamente se trata de la seta mostrada, le da al botón de aceptar.

Esta ventana está formada por un *CarouselView* (imágenes de cada especie) junto a un *Bottom sheet* (nombre e información clave), y un *TextView* (Probabilidad). Todo esto encapsulado dentro de un *ViewPager*, para cada especie.

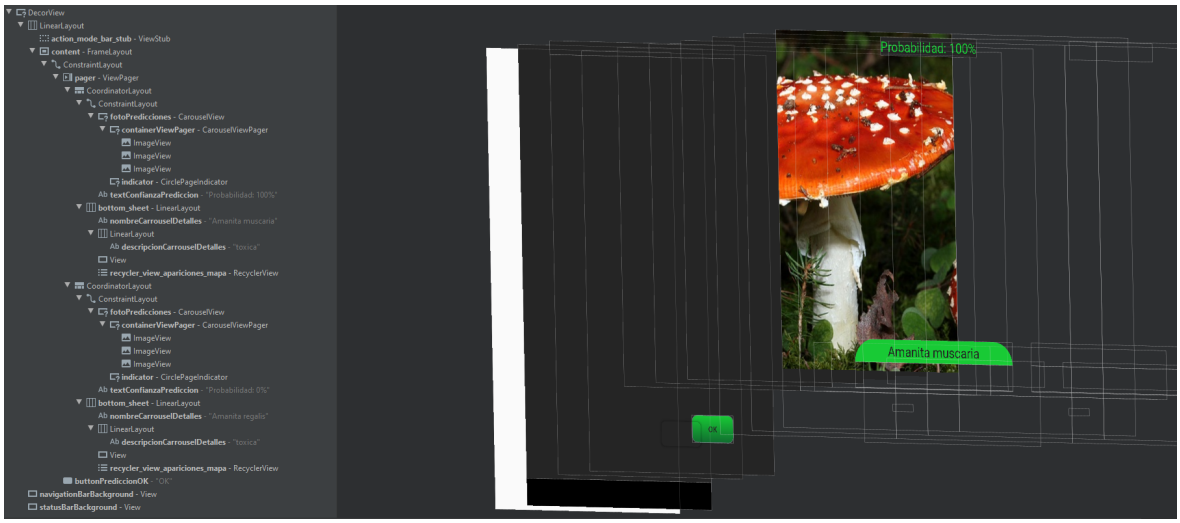


Figura 6.7: *Descomposición del layout del carrousel*

Detalles

Una vez confirmado de que seta se trata, se abre una nueva ventana con toda la información relativa a esa especie, incluyendo un carrousel con varias imágenes, y otro carrousel con tarjetas de las setas con las que es posible confundirla. Además, se mostrará un botón para guardar el hallazgo y acabar con el proceso de identificación.

```
ByteArrayOutputStream bos = new ByteArrayOutputStream();
Bitmap resized = Bitmap.createScaledBitmap(imageBitmap, ↩
    ↪ (int)(imageBitmap.getWidth()*0.5), (int)(imageBitmap.getHeight()*0.5), true);
resized.compress(Bitmap.CompressFormat.JPEG, 70, bos);
byte[] img = bos.toByteArray();
```

En el caso de que la imagen sacada sea excesívamente grande, se reescalará y comprimirá. Para que resulte más manejable y no sature las bases de datos.

6.2.2 Mapa

Esta funcionalidad consiste en ofrecer un mapa en el cual se observen las zonas forestales cercanas, junto a sus características. Así como una lista de las setas que, en base a las condiciones meteorológicas y la fecha, es probable que aparezcan en cada zona.
Además, como funcionalidad extra, se podrá filtrar que tipos de zonas forestales mostrar.

El desarrollo de esta herramienta consta de las siguientes partes:

- Construcción de la base de datos con las zonas forestales de España.
- Prever que setas aparecerán en cada zona en base a la fecha y las condiciones meteorológicas recientes.
- Integrar y personalizar el SDK de Google Maps.
- Cargar y descargar dinámicamente capas del mapa.

Indexado

Para poder ofrecer esta crucial funcionalidad, es necesario disponer de una base de datos con las zonas forestales y sus características, presentes en todos aquellos países en los que se quiera lanzar la aplicación. Para esta primera versión, el desarrollo se va a centrar en dar soporte al territorio Español.

Figura 6.8: MFE50

Convenientemente, el gobierno de España, a través del *Ministerio para la Transición Ecológica y el Reto Demográfico* dispone, de forma pública, una colosal base de datos con todas y cada una de las zonas verdes, así como su información correspondiente al tipo de vegetación, estado y usos.

Sin embargo, fue necesario un laborioso y extenso trabajo para juntar, limpiar y formatear los datos.
Se descargaron los recursos por cada comunidad autónoma y se desarrollarón scripts para iterar sobre todos los archivos y unificarlos en un único archivo de SQLite.

La base de datos está formada por colecciones de GeoJSONs. Estructuras de datos que albergan un polígono formado por sus vertices, y el conjunto de propiedades asociadas a esa zona.


```
{
   "type": "FeatureCollection",
   "features": [
      {
      "type": "Feature",
      "geometry": {
         "type": "Polygon",
         "coordinates": [
         [
            [
            -2.50434,
            43.09487
            ],
            [
            -2.50436,
            43.09488
            ],
            [
            -2.50434,
            43.09487
            ]
         ]
         ]
      },
      "properties": {
         "PROV_MFE_1": "Bizkaia",
         "USOS_GENER": "Arbolado",
         "TIPESTR_DS": "Bosque de Plantación",
         "FORM_ARB_D": "Pinar de pino radiata",
         "TIPO_BOSQU": "Coníferas",
         "SP1_DS": "Pinus radiata",
         "O1": 100,
         "E1_DS": "Fustal",
         "SP2_DS": "sin datos",
         "O2": 0,
         "E2_DS": "Sin datos",
         "SP3_DS": "sin datos",
         "O3": 0,
         "E3_DS": "Sin datos",
         "FCCTOT": 100,
         "FCCARB": 0,
         "DISTRIB_DS": "Uniforme",
         "SUPERFICIE": 0.28165488
      }
      }
   ]
   }
```


Pero para poder cargar en tiempo de ejecución los polígonos cercanos al usuario en un determinado radio, se necesita tener indexada toda la base de datos en función de las coordenadas.

Así que el siguiente paso fue crear un script para iterar la base de datos y calcular las coordenadas del centro de cada polígono. No obstante, la cantidad de artefactos del sistema auguraba problemas serios de rendimiento. Y así fue. En las primeras pruebas de la funcionalidad, la cantidad de polígonos cargados excedía los límites de uso de 250MB establecidos por Android. Así que en una revisión del planteamiento, se decidió segmentar la orografía en fragmentos de décima de coordenada, además de simplificar los datos de las coordenadas a tres decimales. De esta manera, se consigue reducir significativamente el tamaño de los datos.
Y así, en tiempo de ejecución, se puede hacer un acceso directo a un conjunto de polígonos en base a las coordenadas en las que se encuentra el usuario.

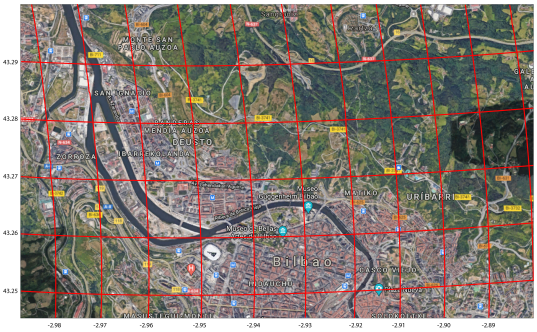


Figura 6.9: Malla indexada del mapa

Previsión de apariciones

Existen dos aproximaciones posibles para abordar la implementación de esta funcionalidad. La primera, con un enfoque determinista, en el cual se estipule en que época y con que condiciones meteorológicas crece cada seta.
O la segunda, un enfoque más pragmático, en el cual teniendo un registro de hallazgos, con la fecha, la localización, y un histórico meteorológico, entrenar e inferir una red neuronal capaz de predecir su aparición.

Claramente es preferible una aproximación más realista qué aprenda a determinar los patrones de muestras reales, y ser capaz de predecir para cada zona en qué momento aparecerá cada seta. Sin embargo existen limitaciones a esa implementación, siendo la principal, la falta de datos en las primeras fases del proyecto. Problema que se solucionará cuando la aplicación consiga una masa crítica de usuarios, y sus hallazgos sean lo suficientemente abundantes cómo para poder emplear algoritmos de Big Data para predecir su aparición.

Por lo tanto, para el desarrollo inicial de la aplicación, se va solventar de forma sencilla, recopilando información de cuando y en que tipos de bosque crece cada seta, y teniendo en cuenta el historial meteorológico de los últimos días, se le presentará al usuario las listas de posibles apariciones en el desplegable de la información de cada zona del mapa.

Para hacer esto posible, únicamente hace falta la base de datos con la información, la fecha del sistema y una API con posibilidad de históricos meteorológicos.

Tras investigar que servicios meteorológicos existían, me decidí por utilizar los límites gratuitos de Open Weather Map, establecidos en un máximo de 60 llamadas por minuto.
Esta API ofrece información meteorológica en tiempo real, a previsión de futuro y de los cinco días anteriores. Solo con enviar una petición con las coordenadas deseadas.

Una vez obtenidos todos los elementos, el desarrollo es tan sencillo como un par de sentencias SQL y unos cuantos *switch*.

Google Maps SDK

Toda esta funcionalidad se construye sobre los mapas de Google ofrecidos a traves de su SDK. El cual ofrece una infinidad de funciones y características.
Entre ellas, la posibilidad de añadir capas, polígonos, formas, marcadores, etc…
El mapa se configura de la siguiente manera:

- Primero, se lanza el fragment del mapa vacío.

```
mapView = view.findViewById(R.id.map);
mapView.onCreate(savedInstanceState);
mapView.onResume();
mapView.getMapAsync(this);
```

- Luego, se actualiza la vista a modo satélite.

```
mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
polygonManager = new PolygonManager(mMap);
```

- A continuación, se obtiene la localización actual del usuario, y se muestra el correspondiente circulo azul.

```
String[] loc =new Localizacion(getContext()).getPosicion().split(",");
mMap.setMyLocationEnabled(true);
mMap.setOnMyLocationClickListener(...){...}
```

- Al mismo tiempo, se crea un elemento de movimiento de cámara, y se enfoca en la posición del usuario, con un zoom de nivel 18 y una ligera inclinación vertical.

```
mMap.animateCamera(CameraUpdateFactory.newCameraPosition(new ↩
    ↪ CameraPosition(new ↩
    ↪ LatLng(Double.parseDouble(loc[0]),Double.parseDouble(loc[1])),18,20,0)));
mMap.setBuildingsEnabled(true);
mMap.setMinZoomPreference(14);
```

- Más adelante, se cargan y añaden marcadores por cada hallazgo.

```
BD.res res=miBD.getHallazgos();
for (int i=0;i<res.posiciones.size();i++) {
    mMap.addMarker(new MarkerOptions()
            .position(new LatLng(res.posiciones.get(i).split(",")[0], ↩
                ↪ res.posiciones.get(i).split(",")[1]))
            .title(res.nombres.get(i))
            .icon(BitmapDescriptorFactory.fromBitmap(res.imagenes.get(i)))
    );
}
```

- Se definen los eventos onClick para cada elemento.

```
layerMaster.setOnFeatureClickListener((GeoJsonLayer.OnFeatureClickListener) ↩
    ↪ feature -> {
    nombre.setText(feature.getProperty("TIPESTR_DS"));
    descripcion.setText(feature.getProperty("FORM_ARB_D"));
    bottomSheetBehavior.setState(BottomSheetBehavior.STATE_COLLAPSED);
});
```

- Y por último, se inicia el proceso de carga de las zonas forestales (GeoJSONs)

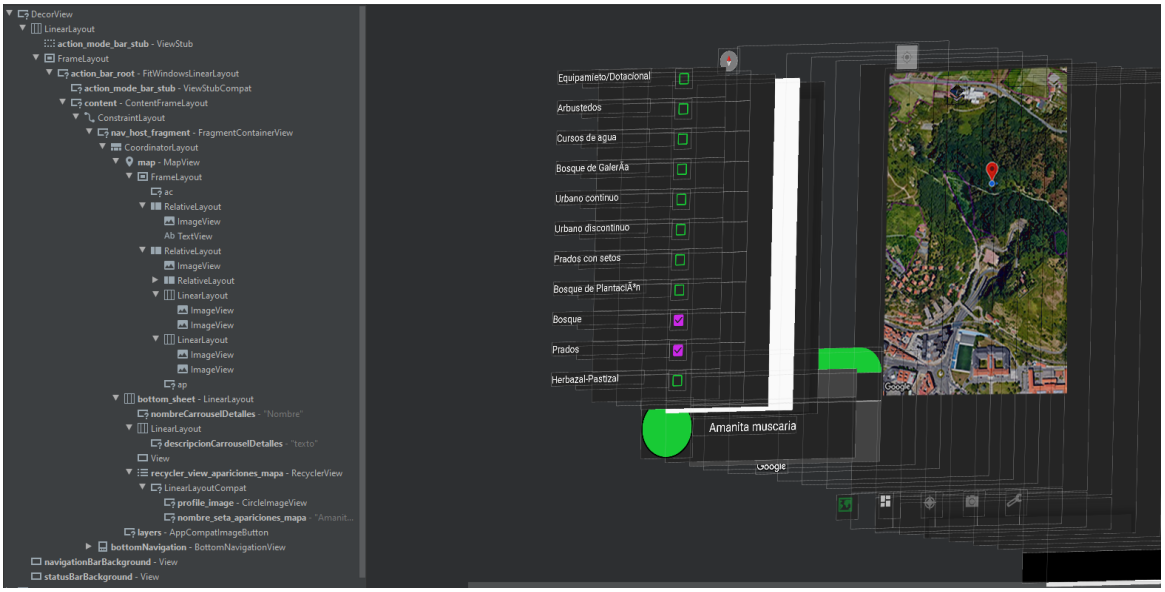


Figura 6.10: *Descomposición del layout del mapa*

Filtro de capas

Un requisito imprescindible de esta funcionalidad consiste en tener un menú flotante en el que poder elegir que tipos de zonas forestales se cargan o no, aplicar los cambios en tiempo real, y guardar las selecciones como preferencias.

Por lo tanto, el proceso de carga de zonas forestales es el siguiente:
Primero, se averigua la posición actual del usuario, y con esas coordenadas se recuperan de la base de datos las secciónes contiguas en un radio determinado. Por cada sección, se extraen los GeoJSONs presentes junto a sus metadatos, y se elabora una lista.
A continuación, se itera sobre todos los polígonos y se clasifican en un *HashMap* en base a su tipo forestal. Para poder acceder así directamente a todas las zonas cercanas al usuario de cada tipo.

El siguiente paso consiste en recuperar las preferencias del menú de capas (por defecto todas seleccionadas), y por cada tipo forestal seleccionado, acceder a sus correspondientes GeoJSONs del *HashMap*, y añadirlos al mapa como **GeoJSONLayers**.

```
SharedPreferences preferences = getContext().getSharedPreferences("capas", ↩
    ↪ android.content.Context.MODE_PRIVATE);
preferences.getAll().forEach((capa, check) -> {
    if ((Boolean) check) {
        capas.get(capa).forEach(GeoJsonLayer::addLayerToMap);
    }
});
```

Sin embargo, el usuario puede seleccionar o deseleccionar tipos forestales en cualquier momento. Por lo tanto, se han definido *listeners* que ejecutarán la carga y descarga de capas dinamicamente.
Para eliminar una capa presente en el mapa, es necesario recorrer todos los elementos cargados y buscar coincidencias en el nombre en base a los metadatos.

```
public boolean onMenuItemClick(MenuItem item) {
    if (item.isChecked()){
        descargarCapa((String) item.getTitle());
    }else{
        cargarCapa((String) item.getTitle());
    }
    item.setChecked(!item.isChecked());
    guardarPreferenciasCapas(item.getTitle(),item.isChecked());
```

Por último, a cada GeoJSONFeature se le dota de un *onClickListener* que desplegará el *bottom sheet* y mostrará su tipo forestal y la lista de setas que es probable encontrarse.

6.2.3 Hallazgos

Cada vez que un usuario completa el proceso de identificación, se le da la opción de guardar el hallazgo. En caso afirmativo, se almacena en la base de datos la imagen, el nombre de la seta que el usuario ha confirmado, y la localización en la que se hizo. En caso de ser usuario premium, se repetirá el mismo proceso contra la base de datos de *Firebase*.

```
            //Meter el hallazgo en sqlite
            mBD.meterHallazgo(getIntent().getExtras().getString("Seta"),img,pos, ↩
                ↪ DateFormat.getInstance().format(System.currentTimeMillis())+"");

            //Subir hallazgo a Firebase
            Storage.getStorage().subirFotoInferencia(img,getIntent().getExtras().getString("Seta"),pos);
```

Desde el menú principal se puede acceder a la sección de hallazgos, en la cual se muestra una lista con las imágenes de los hallazgos.
El principal problema con el desarrollo de esta funcionalidad es el tamaño de las imágenes. En las primeras versiones se hizo evidente que en el momento que cargar la lista de hallazgos, a poco que tuviese más de diez elementos, el rendimiento se veía severamente afectado. Esto se debía a que cada imagen pesa alrededor de los 10MB.
Por lo tanto, a partir de entonces, antes de guardar cada hallazgo, se comprimen las imágenes hasta alcanzar 1MB de tamaño.

6.2.4 Usuarios

Es posible usar la aplicación sin estar registrado, y en ese caso, toda la información se guardará localmente de forma anónima. Pero también se da la opción de identificarse y tener la capacidad de acceder a las funciones premium.
Para simplificar drásticamente la gestión y autenticación de usuarios, se va a implementar el servicio de autenticación con Google que ofrece *Firebase*. De esta manera, los usuarios se registran e identifican con un solo botón, y yo tengo un registro de los usuarios, sin tener que gestionar las contraseñas.

6.3 Experiencia de usuario

Una parte crucial de toda aplicación movil es ofrecer una experiencia de usuario cómoda, satisfactoria y memorable. Y los aspectos que lo componen, son:

- La interfaz gráfica
- Las animaciones
- La navegación
- El rendimiento

Interfaz gráfica

Si hay un aspecto que tiene que ser especialmente trabajado en esta aplicación, ese es la interfaz gráfica. Tiene que ser el factor diferencial que lo distinga del resto de aplicaciones.

La autoridad en cuanto a buenas prácticas de diseño de interfaces se refiere, es Material design. Y a lo largo de toda esta fase se van a usar sus directrices como referencia.

Se ha utilizado una paleta de colores en armonía compuesta RGB con preferencia por los tonos saturados verdes y violetas. Los cuales, combinándolos, por supuesto, con un prominente tema de fondos oscuros, crean la seña de identidad de este proyecto.

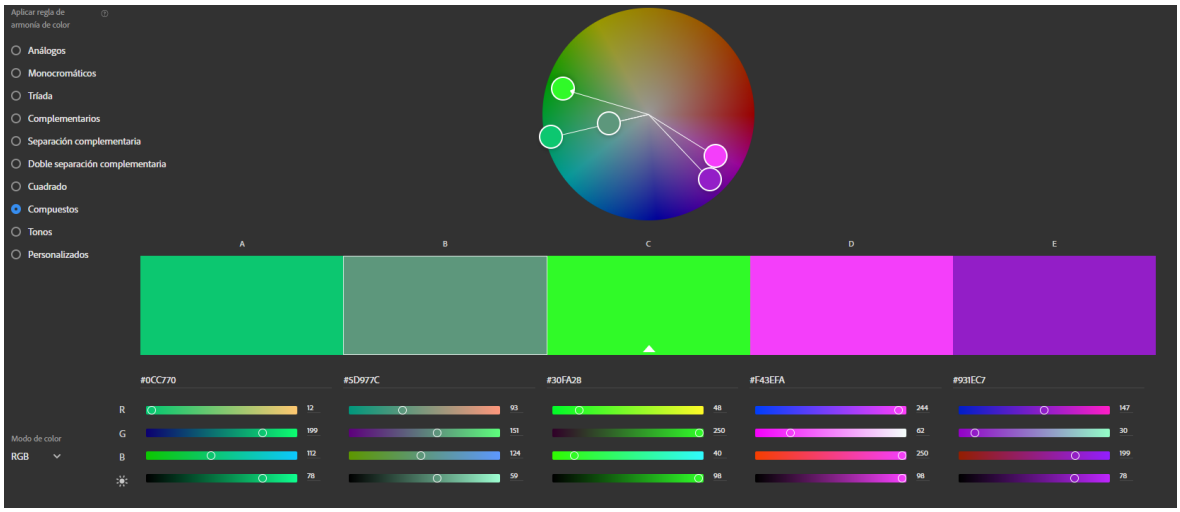


Figura 6.11: *Rueda cromática*

Animaciones

Para dotar de vida la aplicación, se van a implementar una serie de animaciones. La más importante, la escena de la pantalla principal.
Se trata de un diseño vectorial de un paisaje, el cual, mediante un movimiento de cámara de *travelling* vertical descendente revela un plano a nivel de suelo del que, mediante una animación de *slide up*, muestra el *Bottom Navigation Bar*.

Los gráficos se diseñan en *Adobe Illustrator*, luego se animan en *Adobe After Effects*, y finalmente se exportan en formato JSON y se renderizada nativamente en la aplicación gracias a la maravillosa librería de Lottie.

Navegación

La distribución de las funcionalidades, y la navegación entre ventanas tiene que ser lo más ágil y fluida posible. Por eso, se ha diseñado e implementado una navegación en base a un Bottom Navigation Activity y el extenso uso de fragments para albergar las distintas ventanas.
Esto crea una experiencia de usuario más fluida y agradable.

Accesibilidad

Hay que prever el hecho de contar con usuarios presentes en el marco de la diversidad funcional. Por lo tanto, hay que tener en cuenta los siguientes aspectos:

- Incluir *ContentDescription* en todos y cada uno de los elementos de las ventanas.
- Asegurarse de tener el suficiente contraste entre elementos con la paleta de colores diseñada.
- Asegurar la buena visibilidad de todos los elementos en el probable caso de incidencia directa de la luz del sol.

Además de varias recomendaciones en cuanto a la presentación de contenido y su distribución. Como siempre, dictaminadas por Material Design.

Rendimiento

La fluidez y los mínimos tiempos de carga generan una experiencia de usuario agradable y ágil. Sin embargo, es un aspecto que corre peligro en este proyecto, debido al manejo de una gran cantidad de datos y la ejecución local de complejas redes neuronales.

Por lo tanto, conviene prestar especial atención a los momentos de sobrecarga del sistema, y presentarle al usuario animaciones de carga y de proceso que indiquen claramente que se está trabajando en segundo plano.

6.4 Plataformas de distribución

Una vez que la aplicación se ha desarrollado y testeado, es momento de lanzarla al mercado. Para ello, Google ofrece muchas herramientas y procedimientos en su Google Play Console. El proceso seguido en esta plataforma consta de varios pasos:

- Adquirir una cuenta de desarrollador.
- Crear el proyecto y elaborar la página de Google Play.
- Redactar y alojar la política de privacidad y protección de datos.
- Declarar el contenido y solicitar la clasificación.
- Configurar la cuenta de comerciante.
- Configurar y lanzar un segmento de pruebas cerrado: Alpha.
- Configurar y lanzar un segmento de pruebas abierto: Beta.
- Lanzar la aplicación a producción.

A continuación se muestran algunos de los aspectos más relevantes.

Ficha de Play Store

El primer contacto de los potenciales usuarios con este proyecto es a través de la ficha en Google Play. Por lo tanto, hay que asegurarse de transmitir la esencia de la aplicación a través de las imágenes, la descripción, el icono y el *trailer*. Además de incorporar técnicas de ASO(App Store Optimization), para mejorar el posicionamiento.
Todo esto con el fin de llegar a más gente e incentivar las descargas y las valoraciones.

Se puede comprobar el resultado en Google Play

Segmento de pruebas

Ya sea mediante una implementación cerrada(*Alpha*) o abierta(*Beta*), la plataforma ofrece los mecanismos para establecer grupos de *Testers*, desplegar distintas versiones en desarrollo de la aplicación y recibir feedback al respecto.

Para este proyecto, primero se desplegó una versión prematura para un segmento cerrado de personas, y más adelante, cuando el desarrollo estaba en sus fases finales, se lanzó una versión *Beta* al alcance de cualquiera.

Lanzamiento a producción

Cuando se acabe el desarrollo por completo, y la versión *Beta* haya estado en el mercado el suficiente tiempo como para haber detectado y arreglado la mayor parte de problemas, se lanzará la aplicación en su fase de producción, y empezará la vida pública del proyecto como tal.

Este paso debe ir acompañado de la correspondiente campaña de marketing y de un plan de comunicación con los usuarios. Principalmente mediante redes sociales.

7 Validación y pruebas

A la hora de evaluar la calidad de las aplicaciones móviles, hay varios aspectos a los que se debe prestar especial atención:

- Asegurarse del buen desempeño de las funcionalidades.
- Comprobar que la interfaz gráfica responde y se comporta correctamente.
- Comprobar la compatibilidad con distintos dispositivos y distintos tamaños de pantalla.
- Comprobar el uso de los servicios externos.
- Comprobar el rendimiento de la aplicación en distintas condiciones.
- Comprobar el funcionamiento sin conexión.
- Comprobar el correcto despliegue e instalación.

A continuación que detallan las herramientas utilizadas, los procesos seguidos y las pruebas definidas.

7.1 Pruebas unitarias

El primer tipo de pruebas que se van a realizar son las programáticas. Se definen secuencias de comandos para verificar que internamente todos los métodos funcionan correctamente. Estas aseguran que en el momento de ampliar la funcionalidad, o cambiar algún aspecto, todo lo construido hasta ese momento sigue funcionando correctamente.

Algunas de las pruebas más relevantes definidas, son:

Pruebas programáticas		
Acción	Resultado esperado	Resultado obtenido
Sacar una foto con una cámara con demasiada resolución	La imagen se comprime y se infiere el modelo	Lo esperado
Sacar una foto con una cámara con resolución mínima	Se infiere el modelo	Lo esperado
Sacar una foto en horizontal	Se infiere el modelo	Se interrumpe la ejecución: Es necesario definir la orientación de la imagen en TFLite antes de inferir el modelo
Sacar una foto en vertical	Se infiere el modelo	Lo esperado
Sacar una foto de una seta	Se infiere el modelo	Lo esperado
Sacar una foto de algo que no sea una seta	Se infiere el modelo y no da nada significativo como resultado	La probabilidad de los resultados es inferior al 30%
Elegir de la galería una foto excesivamente grande	La imagen se comprime y se infiere el modelo	Lo esperado
Elegir de la galería una foto en horizontal	Se infiere el modelo	Lo esperado
Elegir de la galería una foto en vertical	Se infiere el modelo	Lo esperado
Elegir de la galería una foto de una seta	Se infiere el modelo	Lo esperado
Elegir de la galería una foto que no contenga una seta	Se infiere el modelo y no da nada significativo como resultado	La probabilidad de los resultados es inferior al 30%
Cargar los hallazgos cuando no hay ninguno	No se carga nada	Lo esperado
Cargar los hallazgos cuando solo hay uno	Solo sale un hallazgo	Lo esperado
Cargar los hallazgos cuando hay muchos	Se carga toda la lista y se puede hacer scroll	Lo esperado
Guardar un hallazgo de una imagen excesivamente grande	La imagen se comprime y se guarda	Lo esperado

Pruebas programáticas		
Acción	Resultado esperado	Resultado obtenido
Guardar un hallazgo sin poder acceder a la localización	Se guarda sin la localización	Lo esperado
Guardar un hallazgo con una imagen normal y la localización	Se guarda el hallazgo	Lo esperado
Cargar el mapa con hallazgos	Se cargan los marcadores de los hallazgos	Lo esperado
Cargar el mapa sin hallazgos	No se cargan marcadores	Lo esperado
Cargar el mapa sin acceso a la localización	No se carga el mapa	Se interrumpe la ejecución: Es necesario comprobar antes si se tiene acceso a la localización
Cargar el mapa sin zonas cercanas	No se carga ninguna zona	Lo esperado
Cargar el mapa con demasiadas zonas cercanas	En cuanto se acerque al límite, se dejan de cargar	Lo esperado
Identificarse con una cuenta de Google	Se identifica correctamente	Lo esperado
Cerrar sesión	Se cierra la sesión	Lo esperado
Borrar la cuenta	Se borra la cuenta y toda la información	Lo esperado

7.2 Pruebas de la interfaz gráfica

Para comprobar que la interfaz gráfica funciona correctamente y responde adecuadamente a las acciones, se han definido una serie de recorridos y acciones en la aplicación para comprobar el buen funcionamiento de todas las interfaces de la aplicación.
Afortunadamente, existen herramientas que automatizan este proceso y no requieren tener que repetir las acciones cada vez que se hagan cambios en la implementación.

La herramienta que se va a utilizar se llama Espresso, y está integrada en *Android Studio*. Su funcionamiento consiste en grabar a través del emulador las acciones que se realizan, luego automatizar su ejecución y comprobar el resultado. Además, permite la comprobación automática de los distintos requisitos de accesibilidad (*AccessibilityChecks*), para garantizar el correcto desempeño en el marco de la diversidad funcional.

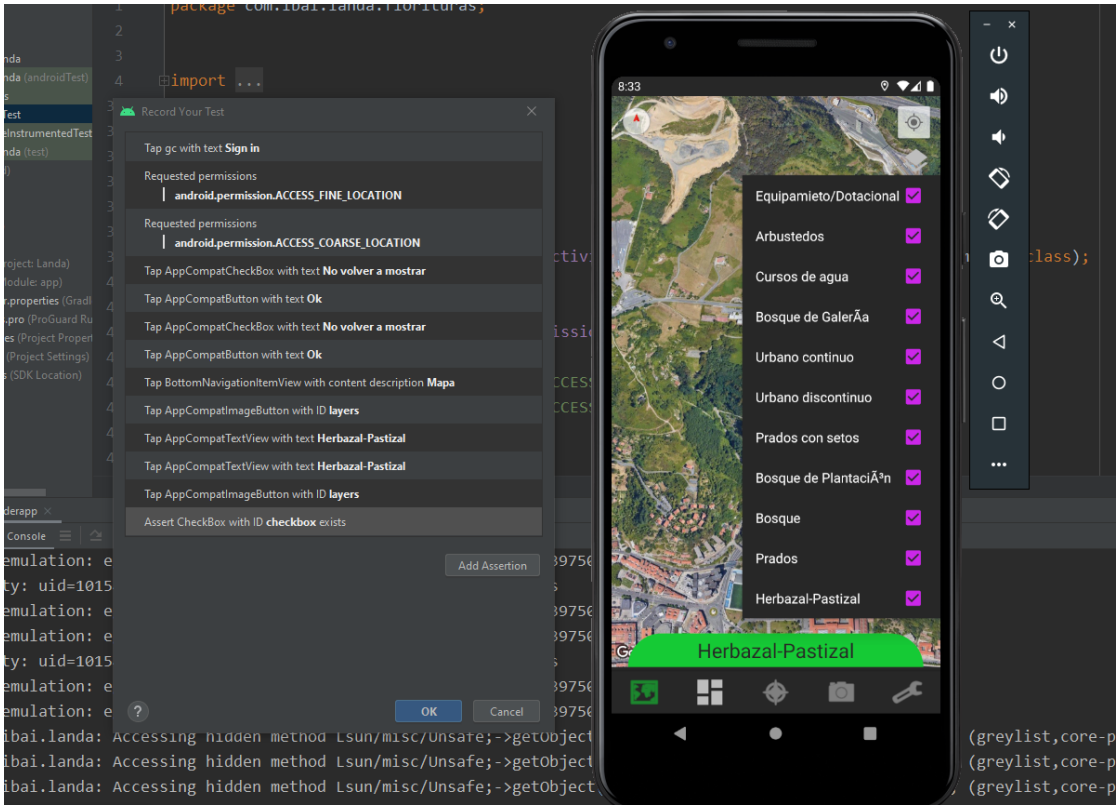


Figura 7.1: Grabador de tests de Espresso

7.3 Evaluación con usuarios

Una vez habiéndonos asegurado de que la aplicación funciona correctamente, el siguiente paso es presentar la aplicación a usuarios finales, proponerles unos cuestionarios y observar su acogida. Este proceso se ha llevado a cabo en tres fases:

- **Alpha:** Esta primera aproximación a la plataforma de distribución *Google Play*, consta de un segmento cerrado de testers, y pretende poner a prueba la infraestructura completa en situaciones reales.
- **Evaluación de expertos:** A continuación, se le pedirá la opinión a un conjunto de personas entendidas en el mundo de la micología, para detectar fallos de concepto o riesgos en la implementación.
- **Beta:** Por último, antes de lanzar la aplicación a producción, se implementará una versión *beta* pública para hacerse una idea a pequeña escala de lo que ocurrirá en producción.

8 Legislación

El desarrollo de aplicaciones móviles está subordinado a leyes que garantizan el correcto uso de los datos de los usuarios. En concreto, las legislaciones vigentes en materia de protección de datos en apps móviles son:

- RGPD (Reglamento General de Protección de Datos)
- LOPDGDD (Ley Orgánica de Protección de Datos y Garantía de los Derechos Digitales)
- LSSI (Ley de Servicios de la Sociedad de la Información y el Comercio Electrónico)

Mediante las cuales se decreta una serie de medidas de imperativo cumplimiento para asegurar los derechos de los usuarios. Lan más importantes son:

- Contar con las apropiadas licencias de derechos de los recursos propios y de terceros. En este caso, contar con las licencias para todas y cada una de las imágenes utilizadas, y disponer de derechos sobre la información del catalogo de setas de elaboración propia.
- Elaborar y transmitir los términos y condiciones de uso.
- Elaborar y transmitir la política de privacidad.
- Restringir el acceso a la aplicación a menores de 14 años.
- Explicar el propósito de la utilización de datos de localización, pedir el consentimiento, desactivarlo por defecto y dar la posibilidad de revocar el permiso en cualquier momento.
- Informar de la posibilidad de ejercer los derechos ARSULIPO (Acceso, Rectificación, Supresión, Limitación, Portabilidad, Oposición) del tratamiento de datos.

Por otra parte, en el futuro se considerará realizar diferentes registros en la OEPM (Oficina Española de Patentes y Marcas), con el fin de proteger la propiedad intelectual de este proyecto.

A continuación de presentan las versiones iniciales de los términos y condiciones de uso, y la política de privacidad.

8.1 Términos y condiciones de uso

1. Estos Términos y Condiciones de Uso regulan las reglas a que se sujeta la utilización de la APP ***Landa*** (en adelante, la APP), que puede descargarse desde el dominio *play.google.com*. La descarga o utilización de la APP atribuye la condición de Usuario a quien lo haga e implica la aceptación de todas las condiciones incluidas en este documento y en la Política de Privacidad y el Aviso Legal de dicha página Web. El Usuario debería leer estas condiciones cada vez que utilice la APP, ya que podrían ser modificadas en lo sucesivo.

2. Sin perjuicio de las obligaciones legales derivadas de la aplicación de la normativa española y europea, este software se proporciona sin ningún tipo de garantía expresa o implícita con relación a su calidad, fiabilidad, compatibilidad, seguridad, rendimiento propósito, precisión o exactitud y no infracción de derechos o cumplimientos legales que puedan sobrevenir de su utilización con independencia del motivo que lo origine. El uso de este software implica que:

   a) Usted acepta los riesgos que pudieran sobrevenir como resultado de la utilización de este software.

   b) Usted acepta las consecuencias que pudieran resultar del uso de cualquiera de los servicios que este software pueda proporcionar indistintamente de la funcionalidad y objeto de su uso.

   c) Usted acepta la responsabilidad de cualquier hecho que pudiera sobrevenir que implique un perjuicio para el usuario o un tercero.

   d) Usted exime a *Ibai Irastorza* de cualquier responsabilidad sobre los resultados negativos o consecuencias no deseadas que impliquen infracción o delito de cualquier tipo.

   e) Usted es responsable de cualquier uso ilícito que pudiera realizar con *Landa* que suponga cualquier incumplimiento normativo o legal.

   f) Usted exime a *Ibai Irastorza* de cualquier responsabilidad frente a la imposibilidad de acceder a su información de manera temporal o de la perdida definitiva de parte o toda la información del usuario depositada en el servicio ya sea privada o pública.

   g) En ningún caso se garantiza la fiabilidad, exactitud y veracidad de la información presente en la APP, por lo que tampoco se asume responsabilidad alguna en cuanto a hipotéticos perjuicios que pudieran originarse por el uso de dicha información. *Ibai Irastorza* le invita a enviarle un mensaje con las discrepancias y correcciones, e intentará en caso de que lo considere y en la medida de lo posible solventarlas.

3. El uso de *Landa* por menores podrá dar lugar al tratamiento de los datos personales de mayores de catorce años salvo en aquellos casos en los que la ley exija para el tratamiento de sus datos personales la asistencia de los titulares de la patria potestad o tutela, y en el caso de los menores de catorce deben contar con el consentimiento de los padres o tutores. Si eres menor de edad para contraer este tipo de contratos según las leyes vigentes consulta con tus padres o tu tutor legal.

4. Cargos: eres responsable del pago de todos los costes o gastos en los que incurras como resultado de descargar y usar la Aplicación de *Ibai Irastorza*, incluido cualquier cargo de red de operador o itinerancia. Consulta con tu proveedor de servicios los detalles al respecto.

5. Estadísticas anónimas: *Ibai Irastorza* se reserva el derecho a realizar un seguimiento de tu actividad en la Aplicación de y a informar de ello a nuestros proveedores de servicios estadísticos de terceros. Todo ello de forma anónima.

6. Protección de tu información personal: queremos ayudarte a llevar a cabo todos los pasos necesarios para proteger tu privacidad e información. Consulta la Política de privacidad de *Landa* y los avisos sobre privacidad de la Aplicación para conocer qué tipo de información recopilamos y las medidas que tomamos para proteger tu información personal.

7. Queda prohibido alterar o modificar ninguna parte de la APP a de los contenidas de la misma, eludir, desactivar o manipular de cualquier otra forma (o tratar de eludir, desactivar o manipular) las funciones de seguridad u otras funciones del programa y utilizar la APP o sus contenidos para un fin comercial o publicitario. Queda prohibido el uso de la APP con la finalidad de lesionar bienes, derechos o intereses de *Ibai Irastorza* o de terceros. Queda igualmente prohibido realizar cualquier otro uso que altere, dañe o inutilice las redes, servidores, equipos, productos y programas informáticos de *Ibai Irastorza* o de terceros.

8. La APP y sus contenidos (textos, fotografías, gráficos, imágenes, tecnología, software, links, contenidos, diseño gráfico, código fuente, etc.), así como las marcas y demás signos distintivos son propiedad de *Ibai Irastorza* o de terceros, no adquiriendo el Usuario ningún derecho sobre ellos por el mero uso de la APP. El Usuario, deberá abstenerse de:

   a) Reproducir, copiar, distribuir, poner a disposición de terceros, comunicar públicamente, transformar o modificar la APP o sus contenidos, salvo en los casos contemplados en la ley o expresamente autorizados por *Ibai Irastorza* o por el titular de dichos derechos.

   b) Reproducir o copiar para uso privado la APP o sus contenidos, así como comunicarlos públicamente o ponerlos a disposición de terceros cuando ello conlleve su reproducción.

   c) Extraer o reutilizar todo o parte sustancial de los contenidos integrantes de la APP.

   d) Con sujeción a las condiciones establecidas en el apartado anterior, *Ibai Irastorza* concede al Usuario una licencia de uso de la APP, no exclusiva, gratuita, para uso personal, circunscrita al territorio nacional y con carácter indefinido. Dicha licencia se concede también en los mismos términos con respecto a las actualizaciones y mejoras que se realizasen en la aplicación. Dichas licencias de uso podrán ser revocadas por *Ibai Irastorza* unilateralmente en cualquier momento, mediante la mera notificación al Usuario.

e) Corresponde al Usuario, en todo caso, disponer de herramientas adecuadas para la detección y desinfección de programas maliciosos o cualquier otro elemento informático dañino. *Ibai Irastorza* no se responsabiliza de los daños producidos a equipos informáticos durante el uso de la APP. Igualmente, *Ibai Irastorza* no será responsable de los daños producidos a los Usuarios cuando dichos daños tengan su origen en fallos o desconexiones en las redes de telecomunicaciones que interrumpan el servicio.

f) IMPORTANTE: Podemos, sin que esto suponga ninguna obligación contigo, modificar estas Condiciones de uso sin avisar en cualquier momento. Si continúas utilizando la aplicación una vez realizada cualquier modificación en estas Condiciones de uso, esa utilización continuada constituirá la aceptación por tu parte de tales modificaciones. Si no aceptas estas condiciones de uso ni aceptas quedar sujeto a ellas, no debes utilizar la aplicación ni descargar o utilizar cualquier software relacionado. El uso que hagas de la aplicación queda bajo tu única responsabilidad. No tenemos responsabilidad alguna por la eliminación o la incapacidad de almacenar o trasmitir cualquier contenido u otra información mantenida o trasmitida por la aplicación. No somos responsables de la precisión o la fiabilidad de cualquier información o consejo trasmitidos a través de la aplicación. Podemos, en cualquier momento, limitar o interrumpir tu uso a nuestra única discreción. Hasta el máximo que permite la ley, en ningún caso seremos responsables por cualquier pérdida o daño relacionados.

g) El Usuario se compromete a hacer un uso correcto de la APP, de conformidad con la Ley, con los presentes Términos y Condiciones de Uso y con las demás reglamentos e instrucciones que, en su caso, pudieran ser de aplicación El Usuario responderá frente a *Ibai Irastorza* y frente a terceros de cualesquiera daños o perjuicios que pudieran causarse por incumplimiento de estas obligaciones.

h) Estos Términos y Condiciones de Uso se rigen íntegramente por la legislación española. Para la resolución de cualquier conflicto relativo a su interpretación o aplicación, el Usuario se somete expresamente a la jurisdicción de los tribunales de Bilbao (España)

8.2 Política de privacidad

De conformidad con el Reglamento (UE) 2016/679, del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (Reglamento General de Protección de Datos - RGPD), *Ibai Irastorza*, informa a los usuarios de la aplicación *Landa* (en adelante, la Aplicación), acerca del tratamiento de los datos personales, que ellos voluntariamente hayan facilitado durante el proceso de registro, acceso y utilización del servicio.

Identificación del responsable del tratamiento

Ibai Irastorza, con NIF nº: 79006296E y domicilio a efectos de notificaciones en: Barrio Incedo, 1, 39808, Incedo, Cantabria, es la entidad responsable del tratamiento de los datos facilitados por los clientes de la Aplicación (en adelante, el/los Usuario/s).

Finalidad del tratamiento de datos

Para proceder al registro, acceso y posterior uso de la Aplicación, el Usuario deberá facilitar (de forma voluntaria), datos de carácter personal (esencialmente, identificativos y de contacto), además de las imágenes y localizaciones correspondientes a los hallazgos, los cuales serán incorporados a soportes automatizados titularidad de *Ibai Irastorza* La recogida, almacenamiento, modificación, estructuración y en su caso, eliminación, de los datos proporcionados por los Usuarios, constituirán operaciones de tratamiento llevadas a cabo por el Responsable, con la finalidad de garantizar el correcto funcionamiento de la Aplicación, mantener la relación de prestación de servicios y/o comercial con el Usuario, y para la gestión, administración, información, prestación y mejora del servicio. Los datos personales facilitados por el Usuario (especialmente, el correo electrónico o e-mail) podrán emplearse también para remitir boletines (newsletters), así como comunicaciones comerciales de promociones y/o publicidad de la Aplicación, siempre y cuando, el Usuario haya prestado previamente su consentimiento expreso para la recepción de estas comunicaciones vía electrónica.

Legitimación

El tratamiento de los datos del Usuario, se realiza con las siguientes bases jurídicas que legitiman el mismo:

- La solicitud de información y/o la contratación de los servicios de la Aplicación, cuyos términos y condiciones se pondrán a disposición del Usuario en todo caso, con carácter previo, para su expresa aceptación.

- El consentimiento libre, específico, informado e inequívoco del Usuario, poniendo a su disposición la presente política de privacidad, que deberá aceptar mediante una declaración o una clara acción afirmativa, como el marcado de una casilla dispuesta al efecto.

Conservación de los datos personales

Los datos personales proporcionados por el Usuario, se conservarán en los sistemas y bases de datos del Responsable del Tratamiento, mientras aquél continúe haciendo uso de la Aplicación, y siempre que no solicite su supresión. Con el objetivo de depurar las posibles responsabilidades derivadas del tratamiento, los datos se conservarán por un período mínimo de cinco años.

Destinatarios

Los datos no se comunicarán a ningún tercero ajeno a *Ibai Irastorza*, salvo obligación legal o en cualquier caso, previa solicitud del consentimiento del Usuario. De otra parte, *Ibai Irastorza* podrá dar acceso o transmitir los datos personales facilitados por el Usuario, a terceros proveedores de servicios, con los que haya suscrito acuerdos de encargo de tratamiento de datos, y que únicamente accedan a dicha información para prestar un servicio en favor y por cuenta del Responsable.

Retención de datos

Ibai Irastorza, informa al Usuario de que, como prestador de servicio de alojamiento de datos y en virtud de lo establecido en la Ley 34/2002 de 11 de julio de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI), retiene por un período máximo de 12 meses la información imprescindible para identificar el origen de los datos alojados y el momento en que se inició la prestación del servicio. La retención de estos datos no afecta al secreto de las comunicaciones y sólo podrán ser utilizados en el marco de una investigación criminal o para la salvaguardia de la seguridad pública, poniéndose a disposición de los jueces y/o tribunales o del Ministerio que así los requiera. La comunicación de datos a las Fuerzas y Cuerpos de Seguridad del Estado, se hará en virtud de lo dispuesto por la normativa sobre protección de datos personales, y bajo el máximo respeto a la misma.

Protección de la información alojada

El Responsable del Tratamiento, adopta las medidas necesarias para garantizar la seguridad, integridad y confidencialidad de los datos conforme a lo dispuesto en el Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de los mismos. Si bien el Responsable, realiza copias de seguridad de los contenidos alojados en sus servidores, sin embargo no se responsabiliza de la pérdida o el borrado accidental de los datos por parte de los Usuarios. De igual manera, no garantiza la reposición total de los datos borrados por los Usuarios, ya que los citados datos podrían haber sido suprimidos y/o modificados durante el periodo de tiempo transcurrido desde la última copia de seguridad. Los servicios facilitados o prestados a través de la Aplicación, excepto los servicios específicos de backup, no incluyen la reposición de los contenidos conservados en las copias de seguridad realizadas por el Responsable del Tratamiento, cuando esta pérdida sea imputable al usuario; en este caso, se determinará una tarifa acorde a la complejidad y volumen de la recuperación, siempre previa aceptación del usuario. La reposición de datos

borrados sólo está incluida en el precio del servicio cuando la pérdida del contenido sea debida a causas atribuibles al Responsable.

Ejercicio de derechos

Ibai Irastorza, informa al Usuario de que le asisten los derechos de acceso, rectificación, limitación, supresión, oposición y portabilidad, los cuales podrá ejercitar mediante petición dirigida al correo electrónico: iirastorza009@ehu.eus Asimismo, el Usuario tiene derecho a revocar el consentimiento inicialmente prestado, y a interponer reclamaciones de derechos frente a la Agencia Española de Protección de Datos (AEPD).

Comunicaciones comerciales vía electrónica

En aplicación de la LSSI (Ley de Servicios de la Sociedad de la Información), *Ibai Irastorza*, no enviará comunicaciones publicitarias o promocionales por correo electrónico u otro medio de comunicación electrónica equivalente que previamente no hubieran sido solicitadas o expresamente autorizadas por los destinatarios de las mismas. En el caso de usuarios con los que exista una relación contractual, jurídica o de servicios previa, el Responsable del Tratamiento, sí está autorizado al envío de comunicaciones comerciales referentes a productos o servicios del Responsable que sean similares a los que inicialmente fueron objeto de contratación con el cliente. En caso de que el Usuario quiera darse de baja a la hora de recibir las citadas comunicaciones, podrá hacerlo remitiendo su voluntad por e-mail al correo electrónico: iirastorza009@ehu.eus.

9 Conclusiones

Objetivos cumplidos

Muchos fueron los objetivos establecidos al comienzo del proyecto, y hay que decir que todos excepto uno se han cumplido.

- Se ha construido un sistema de identificación de setas funcional.
- Se ha construido un catálogo de setas con imágenes e información.
- Se ha desarrollado un sistema de exploración basado en el mapa de las zonas forestales de España.
- Se ha conseguido una relativa la estabilidad y robustez en la aplicación.
- Se ha diseñando e implementado la animación de la pantalla principal.
- Se ha establecido el modelo de negocio.
- Se ha diseñado y redactado mecanismos para el cumplimiento de la legislación vigente.
- Se ha configurado la plataforma de distribución.

Sin embargo, lo único que no se ha conseguido ha sido una precisión de al menos un 90% en el modelo. Lo cual, mirándolo fríamente, es un objetivo bastante ambicioso para la cantidad de setas que se pretenden identificar.

Pero esto no implica que la red neuronal haya sido un fracaso. Es lo suficientemente precisa como para resultar util, y siempre hay tiempo de mejorarla.

Cambios en la planificación

De lo que se planeó en un principio, a lo que acabó sucediendo, hay un abismo de diferencia. Cuando se comenzó el proyecto, hace año y medio, la aplicación consistía en sacar una foto y recibir las predicciones. Pero con el paso del tiempo, el concepto fue pivotando y evolucionando hasta lo que es hoy en día.

Además, las dotes organizativas que tenía en su momento, no tienen nada que ver con la visión que tengo ahora. La poca planificación que hice entonces, no imaginaba a reflejar la cantidad de tiempo que requieren los proyectos de este tipo.
En ningún caso se previó las 2.320 horas que al final ha requerido este TFG.

Resultado

Tras todo este esfuerzo, se ha conseguido construir la aplicación tal y como se había diseñado. Se encuentra alojada y disponible de forma pública en en Google Play.

Lineas futuras

Una vez acabada y lanzada esta primera versión, hay que ir a por la siguiente. Se deberán corregir errores, pivotar algunas funcionalidades, quitar algunas y añadir otras. Todo en función de la acogida del mercado y el feedback recibido.

Algunas de las funcionalidades que se podrían añadir serían:

- Crear una vista de la cámara personalizada e integrada en la aplicación.
- Un sistema de gamificación basado en la rareza de los hallazgos.
- Un catálogo de setas más amplio.
- La construcción de otra red neuronal para predecir que setas aparecerán en que zonas y cuando.
- Un sistema de recetas.
- Ampliar el algoritmo del identificador de setas para poder procesar vídeos.

Reflexiones

Este proyecto es la culminación de cuatro años de carrera y la sinergia de varias de mis pasiones.

Echando la mirada atrás después de todo este tiempo y esfuerzo, me queda una sensación de satisfacción con el trabajo realizado. De ninguna manera es perfecto, pero considero que es lo suficientemente bueno, para haber sido la primera vez que me implico en un proyecto de esta magnitud.

Sin embargo, no puedo evitar pensar si todo este esfuerzo ha merecido la pena. Ha supuesto una dedicación absoluta durante más de un año y medio. Y el éxito del proyecto depende de la acogida que tenga en el mercado.

Bibliografía

Andrew G. Howard, B. C. D. K. W. W. T. W. M. A. H. A., Menglong Zhu. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. Descargado de https://arxiv.org/abs/1704.04861

Bonner, A. (2019). *The complete beginner's guide to deep learning: Convolutional neural networks and image classification.* Descargado de https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb

Brownlee, J. (2017). *A gentle introduction to transfer learning for deep learning.* Descargado de https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Karnes, K. (2020). *Marketing funnel for apps: Must-know mobile guidelines.* Descargado de https://clevertap.com/blog/marketing-funnel/

Luis Perez, J. W. (2017). The effectiveness of data augmentation in image classification using deep learning. , 8. Descargado de https://arxiv.org/abs/1712.04621

Lupis, J. (2017). *Most app professionals say that 5fewer monthly active users make regular in-app purchases.* Descargado de https://www.marketingcharts.com/digital/mobile-phone-81551

Maithra Raghu, C. Z. (2019). Understanding transfer learning for medical imaging. , 1. Descargado de https://ai.googleblog.com/2019/12/understanding-transfer-learning-for.html

Mikael. (2015). *Mushrooms classification - common genus's images.* Descargado de http://www.mushroom.world/mushrooms/search

Peris, D. (2018). *How to optimize your google play store app details page.* Descargado de https://www.apptamin.com/blog/optimize-play-store-app/

Protección de datos en aplicaciones móviles (apps). (s.f.). Descargado de https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/#Como_deben_cumplir_los_desarrolladores_con_la_proteccion_de_datos_en_aplicaciones_moviles

Scott, T. (2019). *How to build an app: Everything you didn't know you needed to know.* Descargado de https://www.youtube.com/playlist?list=PL96C35uN7xGJu6skU4TBYrIWxggkZBrF5

See, M. (2018). *Mushrooms classification - common genus's images.* Descargado de https://www.kaggle.com/maysee/mushrooms-classification-common-genuss-images

Teemu Koivisto, J. H., Tuomo Nieminen. (2017). *Deep shrooms: classifying mushroom images.* Descargado de https://tuomonieminen.github.io/deep-shrooms/