TRABAJO FIN DE GRADO

ESCUELA UNIVERSITARIA POLITECNICA

Departamento de Ciencias Politécnicas
Grado en Ingeniería Informática

Aplicación para la detección temprana de lesiones de piel mediante Inteligencia Artificial

Autor:
Alejandro Brugarolas Sánchez-Lidón

Directores:
Dr. D. Andrés Bueno Crespo
Dr. D. Francisco Arcas Túnez

Murcia, junio de 2021

TRABAJO FIN DE GRADO

UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA UNIVERSITARIA POLITECNICA

Departamento de Ciencias Politécnicas
Grado en Ingeniería Informática

Aplicación para la detección temprana de lesiones de piel mediante Inteligencia Artificial

Autor:
Alejandro Brugarolas Sánchez-Lidón

Directores:
Dr. D. Andrés Bueno Crespo
Dr. D. Francisco Arcas Túnez

Murcia, junio de 2021

AGRADECIMIENTOS

A mi madre, por apoyarme siempre y regalarme todos los buenos consejos que conocen desde que escogí este camino académico.

A mis abuelos Miguel y Encarnita y a mi hermana, por celebrar cada uno de mis logros

A mi pareja Ana, por toda la ayuda y el apoyo que me brinda.

A mis profesores Tomás, Esteban, Víctor y Andrés de la EFA El Campico, por ayudarme a dar los primeros pasos en la informática y darme el empujón para continuar mis estudios en la universidad.

A todos los docentes de esta institución que día a día se vuelcan de lleno y que me han enseñado a la vez que han conseguido despertarme interés en muchos de los campos que trabajan. Mis directores Andrés Bueno y Francisco Arcas por su paciencia, interés y disponibilidad para llevar a buen término este proyecto.

A todos ellos, mil gracias.

Índice

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

RESUMEN

El objetivo principal de este Trabajo de Fin de grado es la creación de una aplicación que permita a médicos de cabecera y otros trabajadores generales de la salud mejorar sus capacidades en materia de detección de potenciales lesiones en la piel a la vez que se obtienen imágenes y etiquetas para seguir entrenando y mejorar un modelo de inteligencia artificial que sea capaz de ejecutar esas mismas detecciones.

Se pretende crear una aplicación para dispositivos Android que sirva como punto de unión entre el conocimiento de dermatólogos especialistas que creen y compartan imágenes dermatoscópicas con su diagnóstico, y médicos de cabecera que entrenen sus habilidades en materia de detección de posibles lesiones de su piel.

A su vez, todas las imágenes y diagnósticos serán almacenados de forma anonimizada para el posterior entreno de una inteligencia artificial. El sistema también almacenará estadísticas sobre todas las consultas creadas para su estudio y control.

Tanto las imágenes como las estadísticas antes mencionadas serán accesibles desde una web desarrollada también dentro del ámbito de este Trabajo de Fin de Grado.

Con el fin de seguir una metodología basada en iteraciones que permita una buena adaptación al cambio, se utilizará la metodología ágil SCRUM.

***Palabras clave*:** Melanoma, Aprendizaje Profundo, *Android, Web, Inteligencia Artificial, SCRUM.*

ABSTRACT

The main objective of this Final Degree Project is the creation of an application that allows GPs and other general health workers to improve their abilities in terms of detecting potential skin injuries while obtaining images and labels to continue training and improve an artificial intelligence model that is capable of executing those same detections.

The intention is to create an application for Android devices that serves as a meeting point between the knowledge of specialist dermatologists who create and share dermoscopic images with their diagnosis, and family doctors who train their skills in detecting possible skin injuries.

At the same time, all the images and diagnoses will be stored anonymously for the subsequent training of an artificial intelligence. The system will also store statistics on all the queries created for its study and control.

Both, the images and the aforementioned statistics will be accessible from a website also developed within the scope of this Final Degree Project.

In order to follow a methodology based on iterations that allows a good adaptation to change, the agile SCRUM methodology will be used.

***keywords*:** Melanoma, Deep Learning, *Skin lesions, Android, Web, Artificial Intelligence, SCRUM.*

1. INTRODUCCIÓN

La inteligencia artificial es ya una realidad en el día a día de las personas. Las aplicaciones de muchas de las empresas más conocidas, dentro del mundo de la tecnología, utilizan inteligencia artificial para mejorar el funcionamiento, captar y retener usuarios, mostrar publicidad acorde con los gustos de la persona que la visualiza y mucho más.

Dentro de esta amplia rama se encuentra la tecnología conocida como aprendizaje profundo o «Deep Learning»: un conjunto de algoritmos utilizados para que una computadora «aprenda» a modelar datos con un nivel de abstracción muy alto, simulando, de esta forma, el aprendizaje humano.

Esta tecnología es cada vez más usada dentro del ámbito de la medicina, donde ha demostrado su efectividad en repetidas ocasiones, sobre todo, en la detección de posibles enfermedades, cánceres y tumores.

1.1 Motivación

Los médicos de atención primaria son la primera línea en la cadena del sistema sanitario y por ello reciben pacientes con todo tipo de síntomas, muchos de ellos muy concretos que requieren ser derivados a médicos especialistas en la materia de dichos síntomas.

El proceso que abarca el reservar una cita con un médico de cabecera hasta que el paciente es recibido por un médico especialista puede llevar días, incluso semanas. Un paciente con un posible cáncer de piel debe ser diagnosticado cuanto antes ya que cada día cuenta. Si los médicos de cabecera pudieran discernir con un simple vistazo o una imagen dermatoscópica entre una lesión benigna y un posible cáncer de piel el proceso de derivación sería mucho más ágil, ayudando, enormemente, al pronto tratado del paciente por un dermatólogo especializado.

Actualmente existe un número muy limitado de conjuntos de datos o «datasets» con imágenes de lesiones de piel, por lo que es difícil crear una inteligencia artificial que detecte estas lesiones con un alto porcentaje de fiabilidad.

Juntando estos dos escenarios descritos, surge la idea de crear una aplicación que ayude a mejorar la detección por parte de médicos de cabecera a la vez que se almacenan imágenes que sirvan para crear un dataset y entrenar una inteligencia artificial que brinde seguridad a diagnósticos estimados automáticos.

1.2 Definición

En este proyecto se desarrollará una aplicación Android que permitirá a dermatólogos crear consultas mediante una imagen y un diagnóstico.

Estas consultas serán recibidas por parte de médicos generales los cuales darán su diagnóstico de forma que la aplicación les indique si han, o no, coincidido con el diagnóstico que dio el médico especialista.

La aplicación también contendrá un modelo entrenado que servirá de estimación para el médico especialista y de aliciente para el médico de cabecera.

Las imágenes serán almacenadas junto con su diagnóstico y una serie de campos opcionales como la zona del cuerpo, fototipo del paciente, edad, etc.

Todo esto permitirá generar también distintas estadísticas que se podrán visualizar en un entorno web, el cual a su vez deberá otorgar la posibilidad de gestionar los distintos usuarios de la aplicación y descargar las imágenes junto con su diagnóstico de forma que puedan ser fácilmente exportadas a un nuevo dataset.

1.3 Objetivos propuestos

Se crearán una aplicación Android y una web desde cero, la aplicación Android integrará un modelo de inteligencia artificial, previamente entrenado, utilizando el dataset HAM 10000 (Tschandl, 2018).

El objetivo principal será recoger nuevas imágenes para ampliar dicho dataset a medida que los usuarios utilizan la aplicación Android. La parte web se utilizará para tener una visualización rápida del estado de la aplicación móvil, así como para ofrecer una interfaz de descarga de las nuevas imágenes almacenadas.

De forma más concreta, los objetivos se definen en las siguientes secciones:

- Desarrollo de una aplicación Android utilizando Java que contenga la funcionalidad de crear consultas para el médico especialista y la de diagnosticar dichas consultas para médicos de cabecera.
- Integrar el modelo ya entrenado para que dé su estimación de diagnóstico y servir de ayuda para el especialista y de aliciente para el médico de cabecera.
- Recoger datos estadísticos de las distintas interacciones con la aplicación.
- Almacenar los usuarios, las imágenes y todo el resto de información en los servicios «Firebase Authentication», «Firebase Firestore» y «Firebase Storage».
- Desarrollo de una aplicación Web utilizando el framework ExtJS que contenga las pestañas para la visualización de datos, la descarga bulk de imágenes diagnosticadas y la administración de los usuarios.
- Servir la aplicación web mediante el uso del servicio Firebase Hosting.
- Controlar todo el desarrollo con la herramienta Git y almacenar el código en la plataforma GitHub.

2. ESTADO DEL ARTE

2.1 Conceptos relevantes del dominio de aplicación

El cáncer de piel es el cáncer más común en países como Estados Unidos, presentando un gran problema para la salud pública (P. Guy, R. Machlin, U. Ekwueme, & Robin Yabroff, 2015).

La detección temprana de este tipo de cáncer es clave para un correcto tratamiento, por ello es esencial que exista una forma más rápida, a la vez que precisa, de detectarlo. Para lograrlo el Deep Learning puede ser de gran ayuda.

El Deep Learning es un subconjunto del Machine Learning y, a su vez, de la Inteligencia Artificial, que consiste en una serie de algoritmos que intentan modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial. (Bengio, Courville, & Pasval, 2013)

Es decir, es un proceso iterativo donde dada una entrada, se extrae la información que el algoritmo cree relevante para posteriormente probarla. Este proceso se hace múltiples veces en paralelo, y los parámetros de salida que han dado mejores resultados servirán como parámetros de entrada para la siguiente generación. (Kaur Grill, 2017)

Este proceso se realiza filtrando la información a través de capas relacionadas entre ellas que conectan la entrada con la salida. La peculiaridad del Deep Learning es que presenta múltiples capas ocultas como se puede observar en la Ilustración 1.

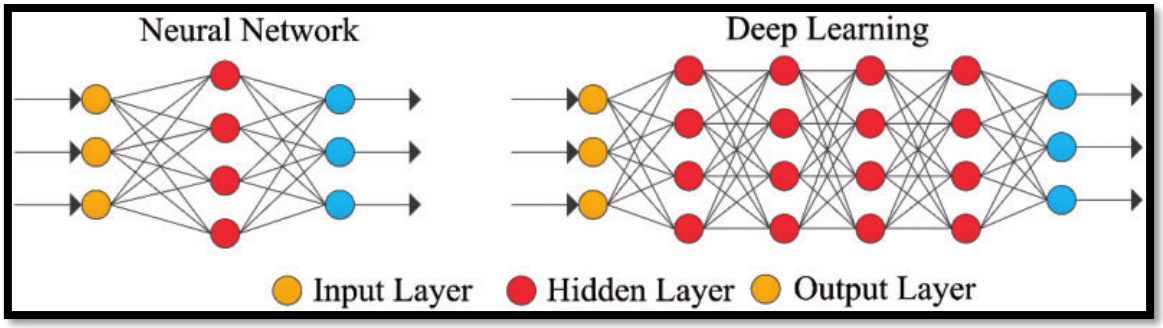

Ilustración 1. Comparativa entre una red neuronal simple y una estructura de Deep Learning. (Dongping & Wanli, 2018)

Estos conjuntos de capas se diseñaron específicamente para simular en estructura y proceso a una neurona biológica (Véase Ilustración 2), pero de manera artificial.

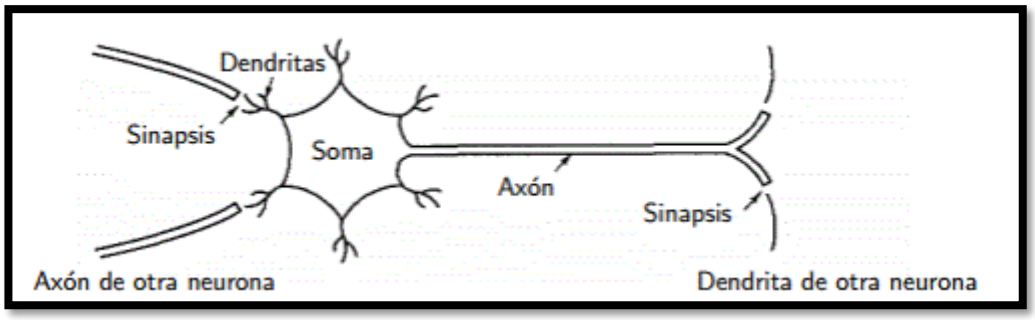

Ilustración 2. Esquema simplificado de una neurona biológica (Larranaga & Moujahid, 2021)

2.1.2 Funcionamiento de las Redes Neuronales Convolucionales (CNN)

Para comprender por qué se necesitan distintas imágenes para entrenar un modelo de Deep Learning es necesario entender el proceso que usa para «aprender».

Una red neuronal convolucional es un tipo de red neuronal artificial compuesta por capas que imita el cortex visual humano para identificar distintas características en una imagen. Contiene varias capas ocultas especializadas y con una jerarquía. Puede clasificar características muy abstractas e identificar objetos de manera eficiente (Qi, 2016).

En primer lugar, hay que entender una imagen a color como tres matrices de píxeles cuyos valores van del 0 al 255. Para un mejor entendimiento se presenta la Ilustración 3.

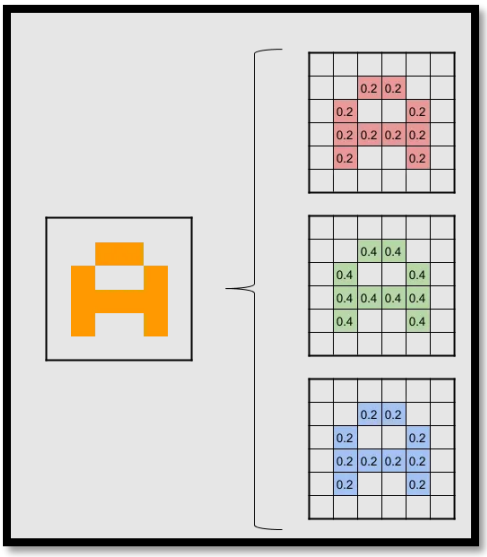

Ilustración 3. Representación de una imagen a color como tres matrices de píxeles que representarían los tres canales RGB. (Bagnato, 2018)

A cada imagen se le puede aplicar un preprocesamiento y normalización antes de ser pasada a la red neuronal. Esto se hace transformando la imagen a blanco y negro, lo que provocaría que las tres matrices se simplificaran en una sola, y normalizando los valores del 0 al 255 para que vayan del 0 al 1 (Véase Ilustración 4).

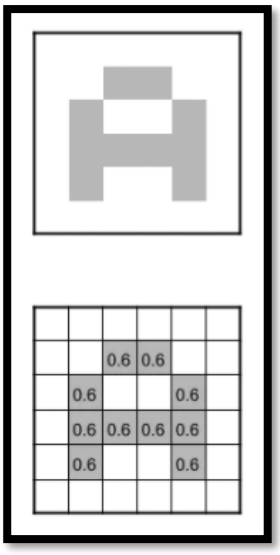

Ilustración 4. Representación de una imagen en blanco y negro como una matriz de píxeles normalizados cuyos valores van del 0 al 1. (Bagnato, 2018)

A partir de aquí, a la imagen se le realizará un producto escalar que resultará en una matriz de menor tamaño que la de entrada, conocida como «kernel» (Véase Ilustración 5). Este proceso se realiza múltiples veces.

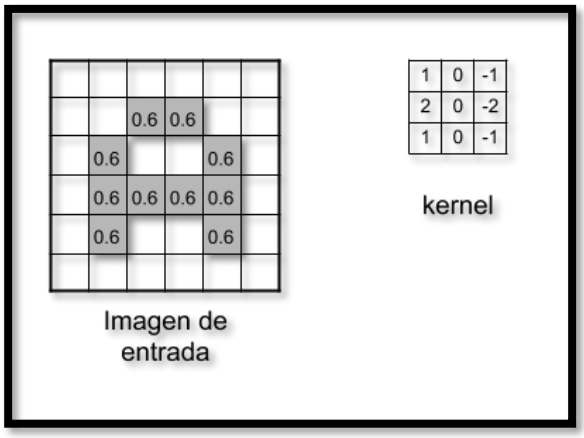

Ilustración 5. El kernel tomará inicialmente valores aleatorios (Bagnato, 2018) y se irán ajustando mediante backpropagation (Qi, 2016).

Con el proceso anterior lo que se consigue es resaltar ciertas características de la imagen justo antes de ser pasadas a las capas que se encargarán de su clasificación. La clasificación se realiza utilizando una función de activación. Existen muchas, pero una de las más utilizadas es ReLU o «Rectifier Linear Unit» definida como $f(x) = \max(0, \mathrm{x})$.

Esta función hace que las características cuyos valores sean negativos se establezcan a 0, para más tarde permitir pasar a la siguiente capa solo a las características con un valor distinto de 0.

En este punto, las características clave ya han sido clasificadas, los dos últimos conjuntos de capas son las de agrupación, que reducen el número de parámetros que requiere la red para aprender, y la de clasificación, que es la que produce la salida.

Estas operaciones serán repetidas a lo largo de todo el conjunto de capas que tenga la red con el objetivo de que cada capa aprenda a identificar características diferentes (MathWorks).

En la Ilustración 6 se muestra de forma esquematizada el diseño por capas que presenta una red convolucional.

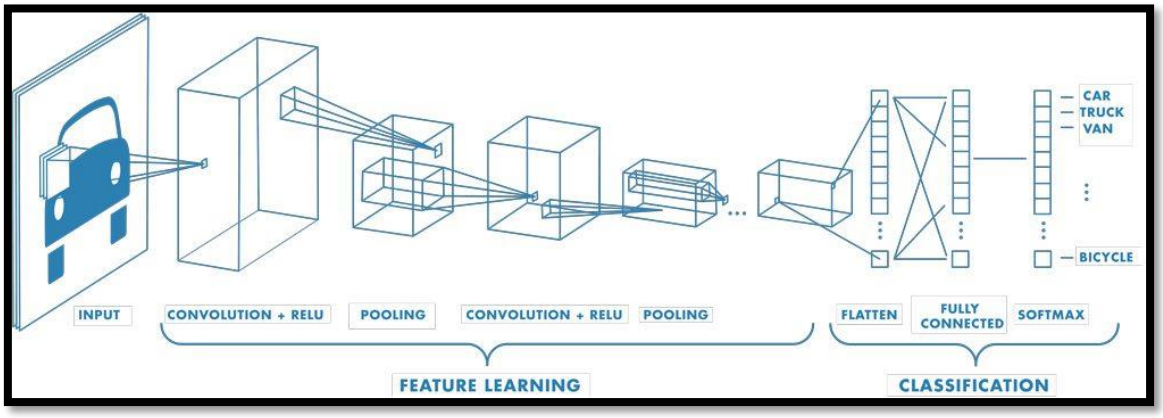

Ilustración 6. Ejemplo de una red con varias capas convolucionales. Se aplican filtros a cada imagen de entrenamiento con distintas resoluciones, y la salida de cada imagen convolucionada se emplea como entrada para la siguiente capa. (MathWorks)

Una vez el modelo ya está entrenado, se puede exportar en distintos formatos, uno de ellos es «tflite», un formato creado por TensorFlow que permite la implementación en TensorFlow Lite, una versión de este framework orientado a sistemas con menos recursos, como los dispositivos móviles.

Al entrenar un modelo con TensorFlow, se suele utilizar la versión de computadora ya que el entrenamiento es más rápido y la implementación más sencilla, esto provoca que los modelos entrenados se exporten en otros formatos como «hdf5». Para solucionar estas incompatibilidades, TensorFlow proporciona funciones nativas para convertir a «tflite». (TensorFlow, s.f.)

Este formato es el utilizado para cargar y utilizar el modelo entrenado en la aplicación Android que se pretende desarrollar.

2.2 Relación con proyectos con la misma funcionalidad

Actualmente existen numerosos proyectos de distintas envergaduras para conseguir mejorar la detección de posibles enfermedades haciendo uso de la inteligencia artificial.

También existen aplicaciones de uso personal enfocadas a la detección, documentación y análisis de condiciones de piel, incluyendo el cáncer de piel, como DermEngine (MetaOptima, 2013).

Para el correcto funcionamiento de un modelo orientado a la visualización y clasificación de imágenes se requiere, además de un correcto diseño de la estructura de capas, de un gran número de imágenes tanto para entrenar como para realizar las pruebas correspondientes. Actualmente, el número de datasets de uso público que contengan imágenes dermatoscópicas de la piel es muy limitado o no está correctamente equilibrado, esto último ocurre en el caso de HAM 10000, como se puede observar en la Ilustración 7.

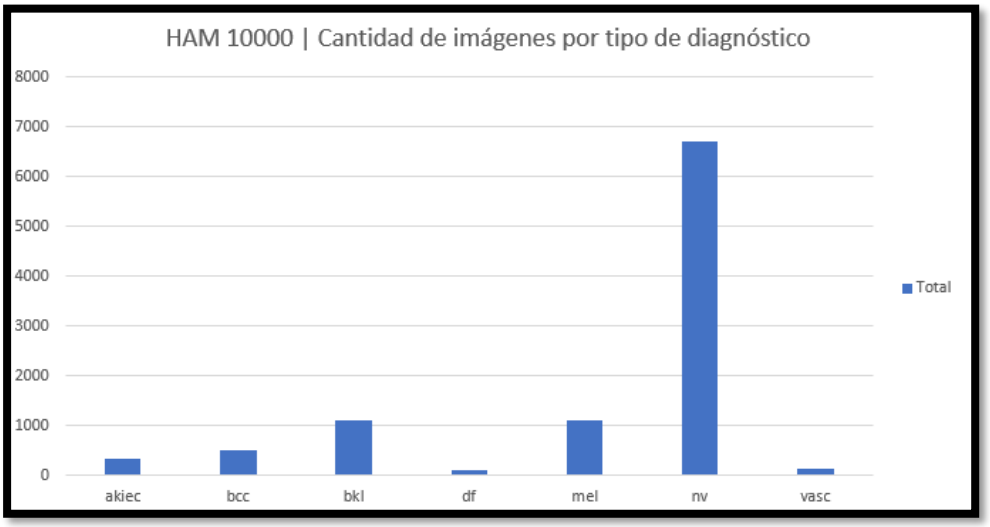

Ilustración 7. Cantidad de imágenes por tipo de diagnóstico contenidas en el dataset HAM10000

Existen artículos que indican que para entrenar redes neuronales para la clasificación de imágenes en su primera versión se necesitan alrededor de 1000 imágenes de cada clase (Warden, 2017). En el caso de imágenes dermatoscópicas, la imagen puede variar debido a otros factores como la localización de la mancha o el fototipo de la persona que la posee, por lo tanto, es necesario un mayor número de ellas.

En la Ilustración 8 se muestra un gráfico que relaciona la precisión de un modelo con el tamaño del dataset utilizado para entrenarlo, nótese que en este caso sólo existía una etiqueta y el máximo de precisión obtenido es de un 84%.

Esto puede servir para proyectos donde no es necesario un alto porcentaje de precisión, pero para diagnósticos médicos no es viable. Aun así, en este caso se muestra el nivel de precisión más alto al utilizar un dataset con alrededor de 5000 muestras.

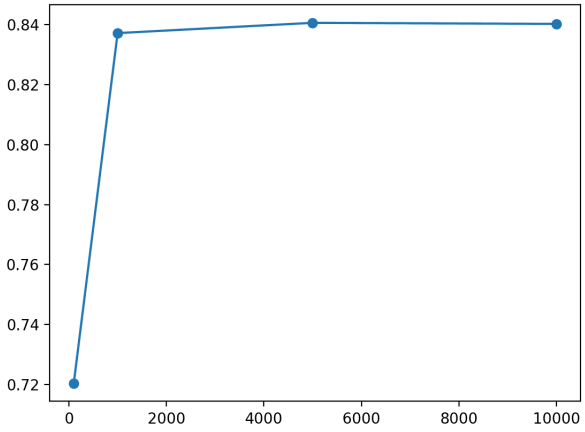

Ilustración 8. Gráfico lineal del tamaño del dataset de entrenamiento contra la precisión obtenida para un problema concreto. (Brownlee, 2019)

Con todo lo anterior sobre la mesa surge la idea de hacer una aplicación que permita obtener nuevas imágenes dermatoscópicas anonimizadas a la vez que médicos generales mejoran también sus habilidades en la detección de lesiones de piel. De esta forma, se conseguirán nuevas imágenes ya diagnosticadas e información de utilidad para crear un dataset todavía más completo.

2.3 Estudio de viabilidad

2.3.1 Alcance del proyecto

El alcance del proyecto se dividirá en dos puntos: la aplicación móvil y la aplicación web.

En cuanto a la aplicación móvil debe ser capaz de gestionar los roles de médico especialista (o dermatólogo) y médico general. El médico especialista se encargará de crear consultas mediante una fotografía, un diagnóstico y la seguridad con la que da el mismo. En el momento de la subida de la fotografía, la inteligencia artificial deberá mostrarle una predicción de su diagnóstico a modo meramente orientativo.

Al dar de alta la consulta se le mostrará un listado con todas las que ya ha creado. Podrá pulsar en cualquiera de ellas para añadir más información sobre el paciente, como la zona del cuerpo, si tiene antecedentes, edad o fototipo.

El acceso a la aplicación se realizará mediante usuario y contraseña pudiendo, los médicos generales, registrarse desde la propia aplicación. Los usuarios de los médicos especialistas se crearán desde la aplicación web.

Cuando un médico general accede a la aplicación, se le mostrará una consulta automáticamente y pseudo-aleatoriamente asignada, al entrar, verá la imagen y un selector de posibles diagnósticos. Al seleccionar un diagnóstico, la aplicación mostrará el diagnóstico estimado del modelo de inteligencia artificial, la del médico especialista y la del patólogo, si se ha rellenado previamente. Tras un breve periodo de tiempo, la aplicación automáticamente llevará al médico general a una nueva consulta para que repita el proceso anterior.

Ambos roles tendrán una pantalla de perfil donde podrán editar su nombre completo y alias. En el caso de los médicos generales, también se mostrará una gráfica con sus diagnósticos totales, aciertos y fallos.

Durante la interacción de los usuarios, la aplicación recogerá y almacenará información estadística que más tarde se podrá visualizar en el entorno web.

Todo lo descrito anteriormente se desarrollará para Android, utilizando Java y TensorFlow Lite para la integración de la inteligencia artificial.

La información será almacenada en «Firebase Firestore», las imágenes en «Firebase Storage» y la gestión de acceso se realizará mediante «Firebase Authentication».

En lo respectivo a la web, existirá también una pantalla de acceso mediante email y contraseña. Al acceder se mostrará una ventana con tres pestañas. La primera «Cuadro de mando» mostrará mediante una tabla y gráficas las estadísticas generales de la aplicación, como el número de consultas totales o agrupadas por diagnóstico.

La segunda pestaña, «Descarga Bulk», permitirá filtrar y descargar en un archivo el conjunto de imágenes cuyo diagnóstico se haya almacenado en el periodo comprendido entre las fechas filtradas. El archivo descargado contendrá todas las imágenes con la siguiente nomenclatura: «UCAM_diagnóstico.jpg».

Por último, la pestaña «Gestión de Usuarios» mostrará los usuarios dados de alta y permitirá crear nuevos y modificar o borrar los existentes.

Esta web se desarrollará con el framework de «Javascript» «ExtJS» y se brindará a través del servicio «Firebase Hosting».

2.3.2 Estudio de la situación actual

Como se ha mencionado anteriormente, existen proyectos capaces de identificar manchas en la piel, así como otros que permiten a usuarios entrenar sus capacidades mediante juegos. No obstante, y debido a la limitación de los datasets de uso público en esta materia, esta aplicación presenta un nuevo paso que permitirá generar nuevos datasets para posteriores proyectos.

Con esta aplicación se logrará que los equipos médicos de atención primaria mejoren sus habilidades en detección de lesiones de piel agilizando el proceso de derivado a un especialista y reduciendo el tiempo de espera habitual que existe en la actualidad al priorizar las lesiones que se puedan identificar a simple vista, esto tendrá afectación directa sobre la salud de pacientes finales cuyas lesiones sean más alarmantes, ayudando a la pronta detección y tratado de posibles enfermedades muy graves.

2.3.3 Estudio y valoración de las alternativas de solución

Actualmente existen proyectos que aplican inteligencia al software dermatológico para la toma de imágenes, documentación y análisis de lesiones de piel, incluyendo el cáncer de piel. A través de la inteligencia artificial, sistemas como estos ayudan a la gestión de flujos de trabajo entre médico y paciente (MetaOptima, 2013).

También existen datasets públicos con imágenes de distintas lesiones de piel de varios sesgos poblacionales como HAM10000, otorgadas para uso público académico en materia de entrenamiento de inteligencias artificiales. (Tschandl, 2018).

El problema que representa este dataset es que debido a la dificultad de obtención de imágenes dermatoscópicas, existen abundantes imágenes de ciertos diagnósticos y escasas de otros, lo que provoca que una inteligencia artificial no se pueda entrenar correctamente.

A pesar de la existencia de estas soluciones que aportan gran valor en el campo y sector en el que se especializan, no existe ninguna aplicación orientada a la creación de nuevos datasets de este tipo, por ello este es el objetivo principal del proyecto.

2.3.4 Selección de la solución.

Tras la realización de la investigación expuesta en los apartados anteriores, se determina que no existe ningún software actual que logre o pretenda lograr los hitos definidos para este proyecto.

Por este motivo se pretende construir las aplicaciones mencionadas desde cero, con la excepción del modelo ya entrenado que se integrará dentro de la aplicación Android.

3. METODOLOGÍA UTILIZADA

La metodología es la forma en la que se desarrolla un proyecto a nivel de gestión de tareas y consecución de hitos. Las metodologías existentes se pueden clasificar en dos apartados: metodologías tradicionales y metodologías ágiles.

A continuación, se expondrán brevemente y a modo resumen las características más importantes de algunas de las metodologías más relevantes contenidas en ambos grupos.

3.1 Metodologías tradicionales

Las metodologías tradicionales existen desde el inicio del concepto de proyecto de desarrollo software, aunque en un primer momento no recibieran este nombre. En su mayoría fueron importadas y adaptadas desde otros sectores de industria.

Estas metodologías se basan en un desarrollo secuencial y una documentación exhaustiva. También destacan por tener altos costos al implementar un cambio y no ofrecer una buena solución para proyectos donde el entorno es volátil (Figueroa, Solís, & Cabrera, 2008).

3.1.1 Cascada

La metodología en cascada se define como una secuencia de fases, que al final de cada etapa reúne toda la documentación para garantizar que cumple con los requerimientos y especificaciones (Montero, Cevallos, & Cuesta, 2018).

Fue una de las metodologías más utilizadas y su pilar principal era tenerlo todo planificado y documentado antes de empezar a escribir código. Esto hacía que los cambios tuvieran altos costes y no se adaptase bien a proyectos donde los requerimientos son cambiantes.

3.1.2 Espiral

El modelo en espiral está basado en cuatro etapas básicas llevadas a cabo de manera secuencial, que se repiten desde el inicio cuando se acaban: especificación, alternativas, evaluación y desarrollo (de Areba, 2001).

Al final de las cuatro etapas se consigue un prototipo que servirá de prueba si se han cumplido los requisitos y que, eventualmente, llegará a convertirse en el producto final.

El riesgo que representa esta metodología es que la espiral nunca termine, siendo un proceso iterativo muy alargado en el tiempo.

3.2 Metodologías ágiles

Las metodologías ágiles surgen como una alternativa a las tradicionales que se centran en proponer alternativas donde el coste de adaptación a los cambios en los requisitos sea menor.

Las metodologías ágiles se basan en desarrollar software que funcione más que conseguir una buena documentación. Esto se consigue prestando más atención a las personas que a los procesos e integrando al cliente en el proceso de desarrollo. Finalmente busca responder a los cambios más que seguir estrictamente un plan.

3.2.1 Extreme Programming (XP)

El «extreme programming» o programación extrema, en castellano, es una de las metodologías ágiles más exitosas de los últimos tiempos.

La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance, donde, como máximo, tres de ellas podrán ser fijadas por actores externos (clientes y jefes de proyecto). El valor de la variable restante es definido por el equipo de desarrollo en función de los valores de las otras tres, de tal forma que si el cliente determina calidad y

alcance, y el jefe de proyecto el precio, el equipo de desarrollo tendrá la libertad para determinar la duración del proyecto (Joskowicz, 2008).

3.2.2 SCRUM

SCRUM es un proceso para desarrollar software, incrementalmente, en entornos complejos donde los requisitos no están claros o cambian con mucha frecuencia.

El objetivo es proveer de un proceso conveniente para los proyectos y el desarrollo orientado a objetos. Es menos burocrático y está más orientado a la productividad (L, 2015).

Este proceso está basado en iteraciones o «sprints» donde, en cada uno, se determina el conjunto de requisitos a implantar y al final se realiza una demostración con cliente y el equipo de desarrollo. Entre medias, se mantiene una reunión diaria donde se explican avances y obstáculos en los desarrollos del sprint actual.

Uno de los principales componentes de SCRUM es el «backlog», que es el conjunto de necesidades para la implementación. De ahí se obtendrán y priorizarán las necesidades a integrar en cada sprint.

Este proceso ha demostrado su eficacia ya que se centra en la productividad y las personas, tratando al equipo como un todo y motivándolos para aceptar nuevos cambios a los desarrollos en lugar de rechazarlos.

3.3 Elección de la metodología y explicación

Tras realizar el estudio de todas las metodologías disponibles, se ha optado por una metodología ágil.

Además de las ya explicadas anteriormente, existen otras que se han descartado por no adaptarse a las necesidades del proyecto. Finalmente se ha

decido optar por **SCRUM** puesto que XP no se adaptaba en cuanto a composición del equipo de desarrollo ni a la figura del cliente.

Debido a que los requisitos en este proyecto pueden ser cambiantes, la metodología elegida ayudará a que el impacto en tiempo y coste de los cambios sea mucho menor que con una metodología en cascada y, con las revisiones tras cada sprint, se consigue que los supervisores se mantengan siempre al corriente del proyecto y puedan detectar cambios o nuevas necesidades cuanto antes.

4. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO

4.1 Tecnologías alternativas para la realización del proyecto.

A continuación, se detallan distintas tecnologías que se han considerado y que representarían alternativas a las utilizadas divididas en cada uno de los apartados que contiene el proyecto.

4.1.1 Parte Móvil

Flutter es un conjunto de herramientas de interfaz de usuario creado por Google para crear aplicaciones que compilan de forma nativa en móvil, web y escritorio, con un solo desarrollo de código (Google, 2017).

Kotlin es un lenguaje de programación, fuertemente tipado, desarrollado por JetBrains. Es 100% compatible e interoperable con Java y permite reducir errores comunes. Es multiplataforma y permite exportar el código incluso a Javascript (JetBrains, 2016).

4.1.2 Parte Web

Angular es un framework de Javascript para el desarrollo de aplicaciones web creado y mantenido por Google, permite controlar la velocidad y escalado al máximo mediante el uso de web workers y server-side rendering (Google, 2016).

React es un framework de Javascript desarrollado por FaceBook orientado a la construcción de interfaces de usuarios de forma sencilla. Permite desarrollar una aplicación por componentes que serán recargados automáticamente, cuando la información que muestran cambie, para llegar a crear aplicaciones complejas escribiendo menos código (Facebook, 2013).

4.1.3 Base de datos

MySQL es un sistema gestor de base de datos relacional que permite despliegues en entornos locales y en la nube, fue adquirido por Oracle que es el actual mantenedor (Oracle, 2010).

Actualmente es el segundo sistema gestor de base de datos más popular, solo superado por Oracle Database (Shanhong, 2020).

MongoDB es un sistema gestor de base de datos no relacional orientado a documentos que permite una mayor escalabilidad y flexibilidad, centrándose en lecturas sobre escrituras y asegurando la indexación de la información necesaria (MongoDB, Inc., 2009).

4.1.4 Backend Web

PHP es un lenguaje de programación de código abierto muy popular, orientado especialmente a desarrollo web con una simplicidad extrema para un principiante y características muy avanzadas para desarrolladores profesionales (The PHP Group, 2001).

NodeJS es un entorno de ejecución para JavaScript de código abierto construido con el motor de JavaScript V8 que permite la ejecución de JavaScript en el servidor utilizando una arquitectura orientada a eventos (Open JS Foundation, 2010).

4.2 Infraestructura software utilizada

Tras realizar un estudio de las tecnologías disponibles para el desarrollo del proyecto se han optado por las siguientes:

Java es un lenguaje de programación basado en clases y orientado a objetos que se compila a un código intermedio conocido como «bytecode». Este código se puede ejecutar sobre un entorno virtualizado incluido dentro de los propios paquetes de Java, lo que lo convierten en un lenguaje para

desarrollo multiplataforma. Durante años fue el lenguaje por excelencia para desarrollo Android nativo.

Firebase es una plataforma, desarrollada por Google, que ofrece multitud de servicios para desarrolladores, compatibles con varios lenguajes de programación. Los servicios que se utilizarán son:

- **Firebase Firestore**: Como base de datos no relacional basada en documentos.
- **Firebase Storage:** Como sistema de almacenamiento para las imágenes.
- **Firebase Authentication:** Como sistema de gestión de usuarios mediante email y contraseña.
- **Firebase Hosting:** Para el hospedaje de la aplicación web.

TensorFlow Lite es una versión simplificada del framework TensorFlow orientada a dispositivos con menos recursos (como un smartphone). En este caso se utilizará para integrar el modelo entrenado y realizar las estimaciones de diagnósticos.

ExtJS es un framework de Javascript para desarrollar aplicaciones orientadas a la muestra de grandes conjuntos de información. Está desarrollado por Sencha y contiene más de 140 componentes pre integrados y probados para asegurar que ofrecen un alto rendimiento (Sencha, s.f.). Se utilizará para el desarrollo de la parte Web.

Android Studio es el entorno de desarrollo oficial para la plataforma Android que remplazó a Eclipse. Proporciona todas las herramientas necesarias para desarrollar aplicaciones para cualquier tipo de dispositivo Android. Está desarrollado por la empresa JetBrains. Se utilizará como entorno de desarrollo de la aplicación Android.

Visual Studio Code es un editor de código open source redefinido y optimizado para construir y depurar aplicaciones web modernas y aplicaciones

en la nube. Es gratuito y contiene gran variedad de extensiones que permiten su escalado hasta el nivel que el usuario requiera.

Trello es un software de administración de proyectos con interfaz simple y flexible, tiene clientes web, iOS y Android. Se utilizará para la gestión de las historias de usuario.

Git es un sistema de control de versiones distribuido, gratis y de código abierto. Diseñado para gestionar proyectos, de todos los tamaños, con velocidad y eficiencia. Actualmente es el controlador de versiones más utilizado, superando a sus predecesores, incluyendo al que ocupa el segundo lugar, Subversion (Git).

GitHub es una plataforma de colaboración que permite, entre otros, alojar repositorios de Git y compartirlo con terceras personas u ofrecerlo a todos los usuarios para que obtengan una copia y/o modifiquen el código alojado. Fue adquirida por Microsoft en 2018. Se utilizará para almacenar el código fuente de las aplicaciones Android y web.

4.3 Infraestructura Hardware

Se utilizará un ordenador de sobremesa y un portátil para el desarrollo general y las pruebas, a partes iguales, tanto de la aplicación Android como de la web. La aplicación será probada en distintos emuladores, pero no requerirán de hardware adicional.

4.3.1 Ordenador de sobremesa para desarrollo y pruebas

- **Procesador:** Intel(R) Core (TM) i7-6700 CPU @ 3.40GHz
- **Memoria:** 16 GB de RAM, 1 TB NVM M.2 y 1 TB SSD.
- **Gráficos:** NVIDIA GeForce GTX 960

4.3.2 Ordenador portátil para desarrollo y pruebas

- **Procesador:** Intel Quad Core i7-7700HQ CPU @ 3.80GHz
- **Memoria:** 16 GB de RAM y 512 GB SSD.
- **Gráficos:** NVIDIA GeForce GTX 1050

5. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

5.1 Sprint 0

El Sprint 0 es el primer paso en un nuevo proyecto atendiendo a la metodología SCRUM. En él se realizan tareas como las siguientes:

- Se preparan y dejan listos los entornos de desarrollo.
- Se trabaja en el «backlog» en materia de preparación de estimación y priorización de historias de usuario.
- Se hace una previsión del reparto de historias de usuario por iteración.
- Se hace un estudio de la arquitectura.

Distintos autores consideran el Sprint 0 como una planificación previa al inicio del proyecto (Garzas, El Sprint cero y el Sprint de release. Javier Garzas, 2013).

5.1.1 Planificación

Como se mencionó en el estudio de la metodología SCRUM, la descripción y planificación de las tareas a realizar está basada en historias de usuario.

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Su objetivo es responder, en un texto no más grande que un Post-It (Véase Ilustración 9), a las siguientes cuestiones:

- Rol que utilizará la funcionalidad.
- Objetivo de la funcionalidad.
- Beneficio que le otorgará la funcionalidad al usuario una vez implementada.
- Adicionalmente, puede contener el criterio de aceptación de la funcionalidad una vez implementada.

Ilustración 9. Ejemplo de plantilla de historia de usuario representada como un Post-It.

Si bien las historias deben ser de similar complejidad (Garzas, Buscando desperdicios Ágiles: los Puntos Historia. Javier Garzas, 2019), existen casos donde habrá historias más complejas o pesadas de implementar que otras. Para ello existe un tipo de ponderación de las historias de usuario en SCRUM conocida como planificación por «Puntos Historia».

Los puntos historia son una puntuación, normalmente del 1 al 10, que permite definir la complejidad de una historia. Para hacer la asignación de éstos existen distintas técnicas. En este proyecto se utilizará una de las más famosas, conocida por el nombre de «Planning Póker».

Planning Póker es una técnica definida en sus orígenes para ser usada en Extreme Programming, pero que ha sido implementada por otras metodologías ágiles debido a su popularidad y efectividad.

El funcionamiento de Planning Póker es sencillo, el cliente, o jefe de producto en el caso de scrum, lee una historia de usuario en voz alta delante de todo el equipo de desarrollo, se hace una discusión previa para clarificar los detalles de la historia de forma que quede lo más claro posible. Hecho esto, cada desarrollador escribe en un papel (o carta) un número que representa su estimación sobre la historia, sin mostrarlo al resto de compañeros. Cuando todos los desarrolladores han escrito su estimación, se muestran todas las cartas, si hay un consenso perfecto, se puede pasar a la siguiente historia.

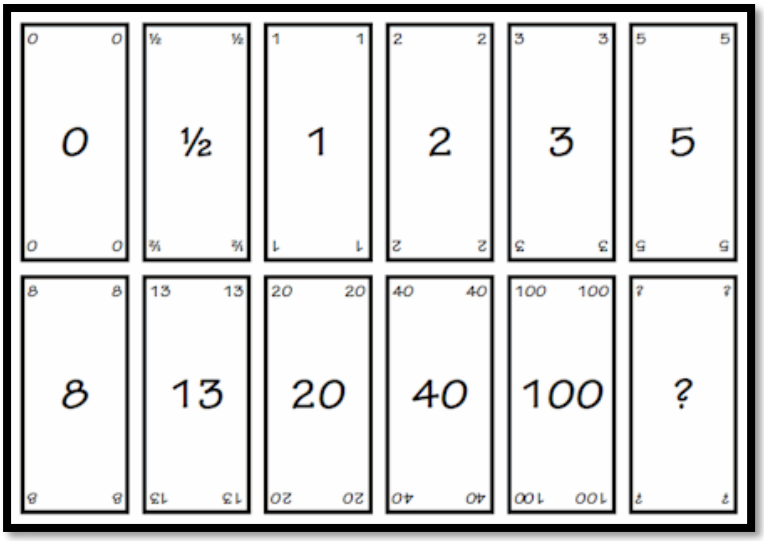

Ilustración 10. Baraja de Planning Póker, para este proyecto únicamente se utilizarán valoraciones del 1 al 10.

Si no existe un consenso, los desarrolladores discutirán sus estimaciones para intentar llegar a uno. Esto no ocurrirá en este proyecto puesto que el equipo está compuesto por un único desarrollador.

Tras definir el método que se usará para la planificación y la estimación de la complejidad, se muestra, a continuación, la Tabla 1 que contiene todas las historias que conformarán el backlog.

Tabla 1. Backlog de historias de usuario que se implementarán a lo largo del proyecto.

ID	Sprint	Descripción	Prioridad	P. Historia
1	0	Instalación Android Studio	1	1
2	0	Instalación VSCode	1	1
3	0	Creación proyecto Firebase	1	2
4	0	Instalación de ExtJs	1	4
5	1	Creación de paleta de colores	2	1
6	1	Creación estructura de documentos de la base de datos.	1	6

7	1	Creación estructura de clases Java.	1	6
8	1	Creación logo.	4	2
9	2	Creación pantalla de registro.	2	2
10	2	Pantalla de acceso aplicación Android.	1	2
11	2	Creación de la barra inferior de navegación	1	2
12	2	Creación de la pantalla de perfil.	3	1
13	2	Interfaz de consultas pendientes.	1	4
14	3	Interfaz de alta de consultas.	1	2
15	3	Implementación captura de imágenes desde galería o cámara.	1	4
16	3	Modelo pre entrenado para estimación de diagnósticos	1	10
17	3	Implementación del guardado de nueva consulta en Firebase Firestore.	1	6
18	4	Listado de consultas por rol	1	3
19	4	Pantalla de consulta por rol	1	7
20	4	Edición de datos de usuario.	3	2
21	4	Implementación del cierre de sesión.	4	1
22	5	Edición de consulta para rol especialista.	1	4
23	5	Diagnóstico de consulta para rol general	1	7
24	6	Guardado de credenciales y acceso automático.	3	3
25	6	Implementación de cálculo y almacenamiento de estadísticas.	1	7
26	7	Implementación de estadísticas en el perfil del rol general.	4	2

27	7	Implementación de algoritmo de obtención de nueva consulta para rol general.	1	5
28	7	Rotación aleatoria en la pantalla de diagnóstico de consultas para rol general.	1	3
29	8	Estructura de ficheros aplicación web.	1	3
30	8	Integración Firebase en la aplicación web.	1	2
31	8	Creación interfaz cuadro de mando.	2	2
32	8	Creación interfaz descarga bulk.	1	1
33	9	Creación interfaz gestión de usuarios.	3	3
34	9	Desarrollo pantalla gestión de usuarios	1	4
35	10	Desarrollo pantalla descarga bulk.	1	8
36	10	Desarrollo cuadro de mando en versión tabla.	1	5
37	11	Implementación de gráficas en cuadro de mando.	1	7
38	11	Implementación de inicio y cierre de sesión.	3	3
39	12	Pruebas aplicación Android y solución de posibles incidencias	1	5
40	12	Pruebas aplicación Web y solución de incidencias	1	4

Tras definir el backlog y atendiendo a la definición de este apartado sobre qué es el Sprint 0, otra de las tareas a acometer es el preparar el entorno de desarrollo. Esto incluye la instalación de Android Studio, Visual Studio Code y ExtJS y la creación del proyecto en Firebase y GitHub.

La realización de cada una de estas preparaciones está definida detalladamente y con capturas de pantalla en la sección **anexos,** incluida al final de este documento (Véanse Anexos A, B, C y D).

Tras estimar los puntos historia de cada tarea contenida en el backlog, se puede crear una idea general de la complejidad del proyecto, no obstante, también es necesario calcular la velocidad de trabajo para conocer el coste y fecha de finalización del proyecto.

La velocidad de trabajo en SCRUM, y otras metodologías ágiles, es el número de unidades de trabajo realizadas en un cierto intervalo de tiempo (Proagilist , 2016).

Existen dos formas de calcular la velocidad de trabajo, mediante los puntos historia, o mediante el número de historias finalizadas en cada sprint. La segunda medida solo es efectiva en caso de que todas las historias tengan la misma o parecida complejidad (Garzas, Buscando desperdicios Ágiles: los Puntos Historia. Javier Garzas, 2019) y, puesto que en este caso eso no ocurre, la velocidad será medida por puntos historia por hora.

Para calcular la velocidad de trabajo, es muy común utilizar la velocidad de los primeros sprints una vez finalizados y extenderlos al resto de sprints como estimación (Mahnic, 2011). No obstante, también es posible realizar esta estimación mediante el cálculo del caso base, donde todo se completa en tiempo y forma, el caso menos favorable, donde se estima que se puede tardar hasta un 20% y el caso más favorable, donde se estima que se puede tardar un 20% menos en realizar la totalidad del proyecto.

Para realizar esta estimación, se utilizará la estimación mediante los casos base, menos y más favorable. Lo primero es calcular el número total de puntos historia en el backlog, en este caso, **147**.

Hecho esto, se debe estimar el tiempo acumulado en horas que se invertirán, en total, en el desarrollo del proyecto. A continuación, se muestra la Tabla 2 que refleja estos datos.

Tabla 2. Estimación de horas semanales que se invertirán en el desarrollo del proyecto

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
1h	1h	1h	2h	3h	6h	3h

Esto hace un total de **17 horas** semanales, por cuatro semanas al mes, **68 horas** semanales, multiplicadas por 6 meses de duración del proyecto, hace un total de **408 horas**.

$$\text{Velocidad de trabajo} = \frac{147 \text{ puntos historia}}{408 \text{ horas}} = 0{,}36\ puntos\ historia\ por\ hora$$

Al aplicar la fórmula de la velocidad del trabajo se obtiene una media de **0,36 puntos historia por hora**, que también se puede expresar como **2,78 horas por punto historia.**

Conociendo este dato, ahora es posible aplicar las reglas de los tres posibles casos (Véase Tabla 3).

Tabla 3. Comparación de alternativas en la estimación.

Caso	Velocidad de Trabajo (ph/h)	Horas estimadas
Más favorable (-20%)	0,45	327
Base	0,36	408
Menos favorable (-20%)	0,3	490

Puesto que se tiene experiencia en proyectos similares, se cogerá el caso base como estimación válida facturable. A continuación, se muestra en la Tabla 4 la subdivisión de horas por sprint

Tabla 4. Estimación de horas por Sprint

Velocidad de Trabajo (ph/h)	Horas estimadas	Sprint	Puntos historia	Horas
0,36	408	0	8	23
		1	15	40
		2	11	31
		3	22	60
		4	13	36
		5	11	31
		6	10	27
		7	10	28
		8	8	23
		9	7	20
		10	13	36
		11	10	28
		12	9	26

Ahora, ya se puede estimar el precio utilizando la técnica de estimación por tarifas de recursos, donde existirá un único recurso que dedicará el 100% de su tiempo efectivo a este proyecto.

Tabla 5. Estimación de horas laborables al año en jornada de 2,5h diarias

Días laborables al año	249 días
Vacaciones	22 días
Fiestas locales (laborables)	2 días
Total días laborables	225 días
Horas de jornada laboral:	2,5 horas
Total horas laborables	563 horas
Duración del proyecto	408 horas

Habiendo calculado el total de horas laborales durante el año y con una jornada aproximada de **2,5h** diarias, es posible calcular el precio del desarrollador por hora.

Tabla 6. Estimación del coste/hora del desarrollador.

	Programador
Neto mes (En jornada de 8h)	1300
Neto mes (En jornada de 2,5h)	407
Número de pagas	12
Total neto año	4884
Retenciones y SS	32%
Coste autónomo/empresa anual	6447
Margen	30%
Coste total	8382
Coste €/hora	15

Tabla 7. Precio total de recursos para el desarrollo del proyecto

	Programador
Personas del perfil en el proyecto	1
Dedicación	100%
Horas desarrollo	408
Horas demostraciones y despliegues	40
Coste total proyecto	6720 €

Por último, quedaría incluir los gastos administrativos, de gestión e infraestructura descritos en la Tabla 8.

Tabla 8. Coste final del proyecto

Concepto	**Coste**
Luz, agua y otros gastos mensuales (proporción)	250 €
Coste del proyecto	6720 €
Total	**6970 € SIN IVA**

6. DESARROLLO DEL CONTENIDO DEL PROYECTO

6.1 Sprint 1

A continuación, se muestra en la Tabla 9 las historias seleccionadas del backlog a implementar en este Sprint.

Tabla 9. Historias extraídas del backlog por implementar en el Sprint 1.

ID	Sprint	Descripción	Prioridad	P. Historia
5	1	Creación de paleta de colores	2	1
6	1	Creación estructura de documentos de la base de datos.	1	6
7	1	Creación estructura de clases Java.	1	6
8	1	Creación logo.	4	2

6.1.1 Explicación del Sprint

Para la paleta se han barajado distintos colores relacionados con la medicina y la piel. En el cruce de estos dos conceptos se encuentra el lila, que recuerda a ambos. Junto a este color se han escogido dos complementarios para mensajes de éxito o error, y el resto de la paleta se compondrá de variaciones de estos.

Por último, se han incluido una serie de grises para distintos componentes de la interfaz puesto que es el color que suele dominar en la mayoría de las interfaces (Refactoring UI.).

Puesto que la base de datos está orientada a documentos y es NoSQL, la estructura se basará en aquella que defina las clases de Android y sus atributos, permitiendo, de esta manera, una mayor versatilidad a la hora de añadir o eliminar campos. La única colección que destacar sería la encargada de almacenar las estadísticas generales, la cual contendrá un único documento con el id 0 para una fácil identificación.

De esta forma, tanto la estructura de clases como la estructura de colecciones de la base de datos quedarían de la forma mostrada en el siguiente diagrama de clases (Ilustración 11).

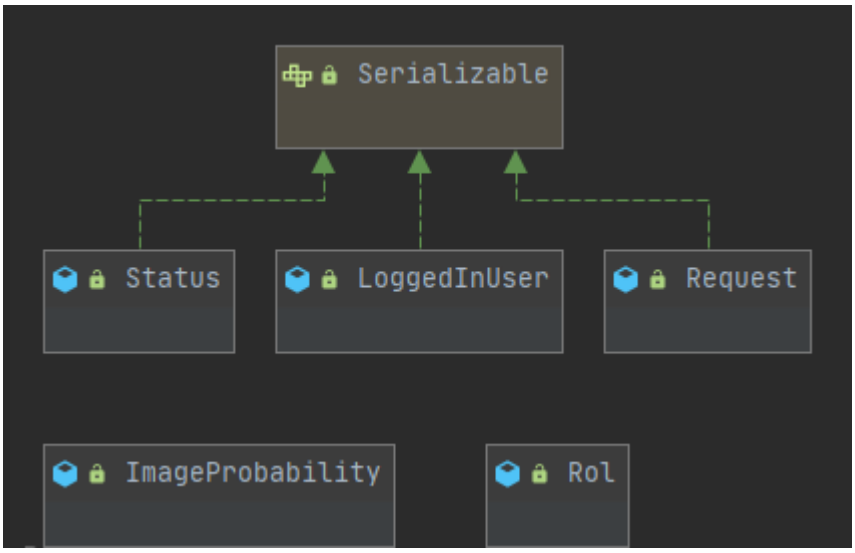

Ilustración 11. Diagrama de clases del modelo de datos utilizadas por la aplicación Android.

Como se puede observar, el modelo es muy simple. Las clases no se enlazan para utilizar al máximo las ventajas de las bases de datos orientadas a documentos y el SDK de Firebase para la transmisión de datos. Se puede destacar la implementación de la interfaz Serializable para poder transferir la información como extras en algunas transiciones entre Activities.

Por último, se ha decidido utilizar un logo sencillo que sea fácilmente identificable y se adapte perfectamente a los distintos tamaños y formas en web y móvil. Junto al logo se le ha otorgado un nombre a la aplicación: «DermalCheck».

6.1.2 Retrospectiva del sprint

Durante este sprint no ha surgido ningún contratiempo si bien en cuanto a colores y logo corresponden a un perfil más de diseñador. Gracias a todos los recursos disponibles en la web ha sido posible conseguir un buen equilibrio. El logo se mostrará en distintas pantallas de la aplicación (Véase un ejemplo en la Ilustración 12).

En lo que respecta a la estructura de clases y base de datos, se ha quedado un diseño muy simple pero igualmente potente. Haciendo uso de las distintas clases de Java no será necesaria ninguna clase de modelo más.

Se puede apreciar que las clases no están relacionadas entre ellas, esto es debido a que son un fiel reflejo de la base de datos Firestore. Al ser construida sin JOINs como cualquier base de datos NoSQL, algunos datos se almacenan de forma redundante para acelerar la velocidad de lectura, que es el propósito general de las bases de datos orientadas a documentos, una lectura mucho más rápida incluso con grandes cantidades de información.

6.2 Sprint 2

En este sprint, se empieza a desarrollar las primeras pantallas completas y las comunicaciones con los sistemas de Backend como se especifica en las historias de la Tabla 10.

Tabla 10. Historias extraídas del backlog por implementar en el Sprint 2.

ID	Sprint	Descripción	Prioridad	P. historia
9	2	Creación pantalla de registro.	2	2
10	2	Creación pantalla de acceso aplicación Android.	1	2
11	2	Creación de la barra inferior de navegación	1	2
12	2	Creación de la pantalla de perfil.	3	1

6.2.1 Explicación del Sprint

En este caso las tareas a realizar son bastante simples: interfaces básicas con entradas de texto y uno o dos botones. Para ambas pantallas se ha utilizado Activities en lugar de Fragments debido a que son pantallas que no requieren de ningún tipo de navegación contextual ni van a ser embebidas dentro de otras interfaces.

Para la realización de la barra inferior de navegación se han utilizado las secciones de Menú y Navegación del gestor de recursos del proyecto, permitiendo diseñar todo desde una perspectiva sencilla y visual.

Para la pantalla de identificación, se ha incluido un checkbox que permite que se almacene el email del usuario y, de esta forma, dejar la aplicación más preparada para futuras implementaciones en sprint venideros.

Toda la arquitectura se ha implementado siguiendo la guía de buenas prácticas de Android, con una arquitectura Modelo Vista VistaModelo (MVVM) con repositorios y datasources para acceder y persistir la información. Esta arquitectura permite que la lógica de la aplicación «sobreviva» a los cambios en el ciclo de vida de la aplicación, como rotaciones de pantalla o puestas en segundo plano.

6.2.2 Retrospectiva del sprint

La creación de las interfaces no ha presentado ninguna dificultad y se ha podido lograr en tiempo y forma, quedando de la manera mostrada en la Ilustración 12.

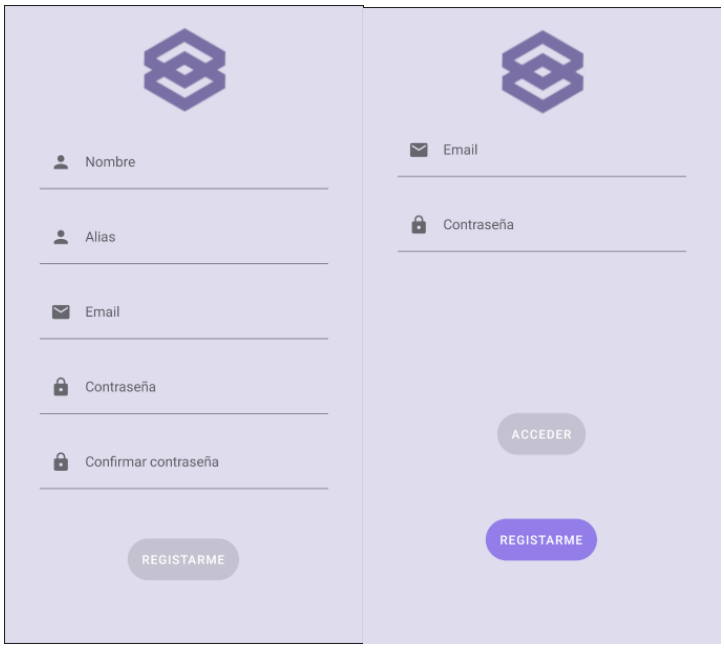

Ilustración 12. Interfaces de registro e identificación de la aplicación móvil DermalCheck.

Si bien el desarrollo de las interfaces ha sido sencillo, también lo ha sido la comunicación con Firebase Firestore y Firebase Authentication debido a los SDK que proporcionan.

La única complicación se ha presentado en mantener la información del usuario dentro de los SharedPreferences sin violar la arquitectura diseñada.

Finalmente, se ha podido realizar, sin mayor problema, gracias al uso de una clase que implementa los métodos necesarios y el uso de «AndroidViewModel». Una clase que extiende de «ViewModel» pero permite trabajar con el contexto de la «Activity».

6.3 Sprint 3

Para este sprint, como se observa en la Tabla 11, ya se inicia el desarrollo de una de las principales pantallas de la aplicación: la creación de consultas, así como la implementación del modelo entrenado y su estimación de los diagnósticos.

Tabla 11. Historias extraídas del backlog por implementar en el Sprint 3.

ID	Sprint	Descripción	Prioridad	P. historia
14	3	Creación interfaz pantalla de alta de consultas.	1	2
15	3	Implementación captura de imágenes desde galería o cámara.	1	4
16	3	Implementación del modelo pre entrenado para estimación de diagnósticos	1	10
17	3	Implementación del guardado de nueva consulta en Firebase Firestore.	1	6

6.3.1 Explicación del Sprint

Como se introducía, para este sprint se ha de ejecutar una de las partes más complejas del proyecto, la integración del modelo pre entrenado.

En primer lugar, el cliente solicita un diseño de interfaz muy sencillo para que la creación de una consulta sea lo más rápido posible. Únicamente se solicita un identificador de paciente como forma de identificar la consulta más tarde, mediante un número auto incremental sugerido pero editable y con la posibilidad de introducir texto. También incluye un seleccionador de imágenes,

un desplegable para seleccionar el diagnóstico y una serie de estrellas a modo de indicador de la seguridad con la que se dicta el diagnóstico.

El seleccionador de imágenes debe admitir tanto la selección de imagen desde la galería como desde la cámara, por lo que es necesario implementar la solicitud de permisos para tomar fotografías.

A continuación, llega la parte compleja. Se diseña una clase específica para gestionar todas las predicciones que recibe el nombre de «Classifier», y se encargará de recibir la imagen y obtener una predicción, compuesta de array con un número entre el 0 y el 1 por cada posible diagnóstico. Cuando se obtiene, se recorre para conseguir el valor más alto, que será el diagnóstico final estimado por el algoritmo.

Puesto que dependiendo del dispositivo en el que se ejecute, la aplicación, el tiempo de estimación puede variar todo el procesamiento se realiza en otro hilo de forma que la interfaz de usuario no se vea alterada. También se ha optimizado lo máximo posible, de forma que la complejidad total de ejecución de todo el código dentro de este hilo sea de O(n).

Por último, la implementación del guardado de la consulta en Firestore ha requerido de un método para convertir los campos que se desean guardar en un «Map». También se realiza una llamada extra para modificar las estadísticas totales de la aplicación.

6.3.2 Retrospectiva del sprint

Este sprint ha presentado una alta complejidad a la hora de su implementación, justo como se esperaba en la planificación, aunque, si bien la integración del modelo de TensorFlow Lite se esperaba difícil, también han surgido complicaciones a la hora de implementar la toma de imágenes desde la cámara. Esto se debe a que, con la última versión de Android, la forma de acceder al sistema de archivos ha cambiado y se ha necesitado un tiempo de investigación extra para aprender cómo funciona este nuevo método de acceso.

El resultado final de este sprint se puede observar en la Ilustración 13

Ilustración 13. Pantalla de creación de consultas disponible para los usuarios con rol de médico especialista en la aplicación Android.

El resto de la implementación se ha hecho lo más intuitiva posible y dando la máxima retroalimentación al usuario, con el propósito de conseguir que en el menor tiempo posible sea capaz de utilizar esta pantalla con gran velocidad para crear múltiples consultas seguidas.

6.4 Sprint 4

A continuación, se muestra la lista de tareas a implementar en este sprint en la Tabla 12.

Tabla 12. Historias extraídas del backlog por implementar en el Sprint 4.

ID	Sprint	Descripción	Prioridad	P. historia
18	4	Listado de consultas por rol	1	3
19	4	Pantalla de consulta por rol.	1	7
20	4	Edición de datos de usuario.	3	2
21	4	Implementación del cierre de sesión.	4	1

6.4.1 Explicación del Sprint

La obtención del listado de consultas debe ser distinto por cada rol. Para el médico especialista, deberá poder visualizar todas las consultas que ha

creado mientras que un médico general solo verá las que tiene pendiente de diagnosticar en ese momento.

De igual forma que el listado, se tratará la pantalla de consulta. Para el médico especialista será una pantalla de edición de información, donde no podrá modificar la foto ni su diagnóstico, pero sí podrá añadir o editar la siguiente información:

- Fototipo del paciente.
- Edad del paciente.
- Localización de la mancha.
- Antecedentes personales y familiares del paciente.
- Sexo del paciente.
- Diagnóstico del patólogo si se conoce.
- Notas u observaciones.

Para el médico general, únicamente se verá la imagen y una ruleta estilo Cupertino con todos los diagnósticos posibles para seleccionar el que desee.

La pantalla de edición de datos de usuario sí será igual para ambos roles puesto que comparten información. Ídem para el botón del cierre de sesión.

6.4.2 Retrospectiva del Sprint

Para las distinciones por roles se podrían haber utilizado muchas técnicas, pero debido a la sencillez y modularización del código, se ha conseguido implementar las distinciones en una sola «Activity», haciendo uso de distintos métodos en los «ViewModel» y «Datasources». Quedando, el resultado final, como se muestra en la Ilustración 14.

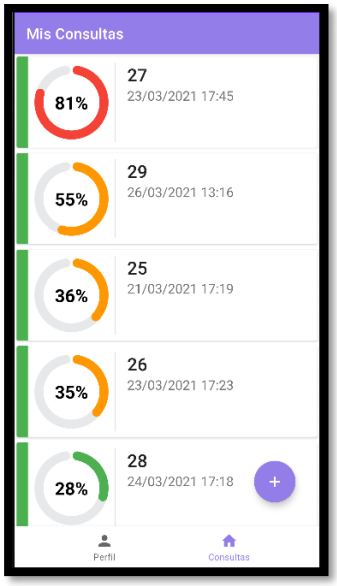

Ilustración 14. Listado de consultas para el rol especialista. Un médico general vería el mismo listado sin el icono de añadir.

La mayor dificultad en este sprint ha sido, sin lugar a duda, la selección de la localización de la macha de forma que fuese intuitiva para el usuario. No se ha encontrado ninguna librería o componente que permitiese realizar esto a pesar de que es un componente bastante común en aplicaciones orientadas a la salud corporal. Finalmente se han creado todas las imágenes y se ha implementado el desarrollo de forma manual, quedando como se observa en la Ilustración 15.

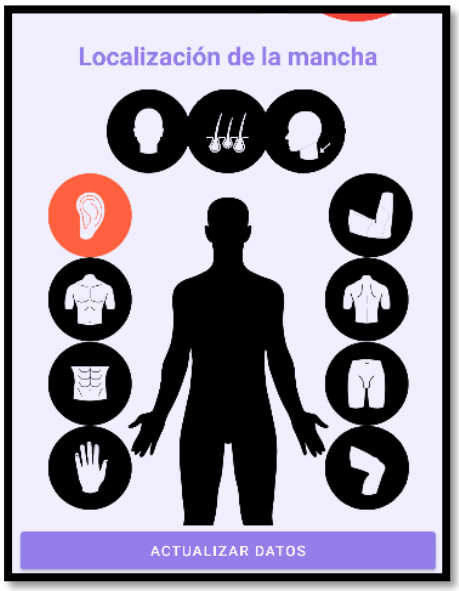

Ilustración 15. Selector de localización de la mancha implementado mediante una serie de imágenes donde queda destacada, en otro color, la seleccionada.

La edición de usuarios y el cierre de sesión no supusieron ninguna dificultad y se pudo implementar antes del tiempo estimado (Véase Ilustración 16), por lo que finalmente el sprint se ha nivelado al tardar más en una historia y menos en otra.

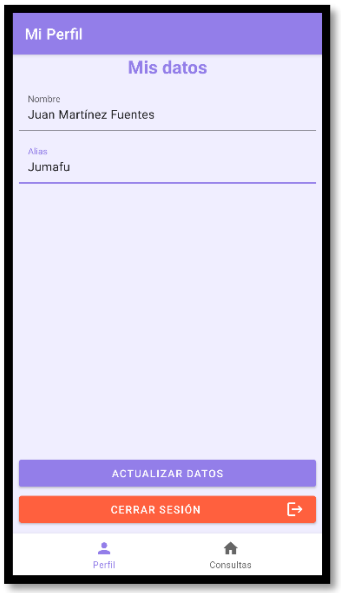

Ilustración 16. Pantalla de perfil visible para un usuario con rol especialista, donde puede cerrar sesión o actualizar su información.

6.5 Sprint 5

Este sprint contiene un menor número de historias, pero su complejidad conjunta es mayor. Esto es debido a que se incluye la edición de la consulta y el diagnóstico por parte del médico general con comunicación a la base de datos de Firestore.

Tabla 13. Historias extraídas del backlog por implementar en el Sprint 5.

ID	Sprint	Descripción	Prioridad	P. historia
22	5	Implementación de edición de consulta para rol especialista.	1	4
23	5	Implementación de diagnóstico de consulta para rol general	1	7

6.5.1 Explicación del Sprint

La edición de una consulta conllevará la implementación de la gestión de cada uno de los campos existentes descritos en el sprint anterior, incluyendo su validación y posterior transformación a un valor almacenable en la base de datos.

La mayor parte de los campos se han dispuesto en la interfaz de forma que el usuario tenga que interactuar lo menos posible con el teclado, reduciendo al mínimo los campos donde hay que introducir texto, esto conlleva que la mayoría de los campos se tengan que transformar en los controladores de la selección de un icono a un valor booleano, entero o cadena de texto.

Para la implementación del diagnóstico se dispondrá de la ruleta creada en el sprint anterior y de un botón con el texto «Diagnosticar». El usuario, al entrar, no verá más información que la imagen. Una vez seleccione su diagnóstico, se mostrará el diagnóstico estimado, el del patólogo si lo hubiera y el diagnóstico del especialista con la seguridad que indicó en un primer momento.

Hecho esto, saldrá una alerta, durante unos segundos, indicando si acertó o falló y al ocultarse se abrirá la siguiente consulta que el médico de cabecera tuviera pendiente o se le asignará una nueva.

6.5.2 Retrospectiva del Sprint

La edición de los campos por parte del médico especialista se terminó antes de tiempo, de esta manera se ha conseguido invertir más tiempo en la implementación del diagnóstico del médico de cabecera y todo lo que eso conlleva, quedando la interfaz final como se muestra en la Ilustración 17.

Ilustración 17. Pantalla de diagnóstico después de seleccionar el diagnóstico donde ya se pueden observar las distintas estimaciones, así como la alerta temporal inferior.

Cuando el usuario da su diagnóstico se almacena la fecha en la consulta, que servirá como indicador de la última vez que se diagnosticó dicha consulta y, automáticamente, se le retirará al médico de cabecera de sus consultas asignadas para que no le siga apareciendo.

6.6 Sprint 6

Para este sprint se desarrollarán las tareas expuestas en la Tabla 14, que incluyen el acceso automático a partir de las credenciales almacenadas en el sprint 2 y se llevará a cabo todo el proceso del almacenamiento de las estadísticas generales y específicas de la aplicación Android.

Tabla 14. Historias extraídas del backlog por implementar en el Sprint 6.

ID	Sprint	Descripción	Prioridad	P. historia
24	6	Implementación de guardado de credenciales y acceso automático.	3	3
25	6	Implementación de cálculo y almacenamiento de estadísticas.	1	7

6.6.1 Explicación del Sprint

Para la implementación del guardado de credenciales y acceso automático, se ha reutilizado el almacenamiento del email creado en el Sprint 2

pero retirando el checkbox. Ahora se almacena el email y la contraseña, automáticamente, en los «SharedPreferences» y al acceder a la aplicación, esta comprueba si tiene credenciales almacenadas, si las tiene, accede.

Para hacer esto también se ha tenido que modificar la función de cerrado de sesión, haciendo que cuando se pulse, la contraseña almacenada se elimine, pero el email se mantenga. Esto provoca que, al acceder de nuevo a la aplicación, la contraseña sea obligatoria pero el email se rellene automáticamente.

La implementación de las estadísticas se ha realizado añadiendo funciones en los puntos de creación, asignación y diagnóstico de consultas, de esta forma se sabe cuántas consultas se han creado y diagnosticado en total, cuantas consultas ha creado un especialista, cuantas ha diagnosticado un médico general y cuántas consultas por tipo de diagnóstico existen en la base de datos actualmente.

6.6.2 Retrospectiva del Sprint

Para la implementación de la eliminación de la contraseña en el cierre de sesión se ha tenido que modificar el «ViewModel» del «Fragment» del perfil para poder trabajar con el contexto sin violar la arquitectura MVVM. Esto se ha realizado sin mayor complicación gracias a la versatilidad que ofrece la metodología SCRUM para adaptarse a nuevos cambios.

La implementación de estadísticas habría sido más sencilla de implementar en una base de datos relacional, ya que se podrían consultar al vuelo, pero debido a la arquitectura de las bases de datos NoSQL, se deben de calcular según se producen cambios para así ganar muchísima velocidad a la hora de obtener la información y mostrarla. No obstante, el cálculo se ha simplificado al máximo en cada uno de los casos de forma que no afecte al usuario, en cuanto a experimentación de lentitudes se refiere, al realizar ciertas acciones.

Puesto que no todas las estadísticas son a nivel de usuario, existe, en la base de datos, una colección exclusiva con el nombre de «statistics» que

almacenará tanto las estadísticas generales como las de consultas por tipo de diagnóstico, de forma que estén agrupadas todas en un mismo lugar y la consulta sea lo más sencilla posible.

6.7 Sprint 7

A continuación, se muestra la lista de tareas a implementar en este sprint en la Tabla 15.

Tabla 15. Historias extraídas del backlog por implementar en el Sprint 7.

ID	Sprint	Descripción	Prioridad	P. historia
26	7	Implementación de estadísticas en el perfil del rol general.	4	2
27	7	Implementación de algoritmo de obtención de nueva consulta para rol general.	1	5
28	7	Rotación aleatoria en la pantalla de diagnóstico de consultas para rol general.	1	3

6.7.1 Explicación del Sprint

Para el médico general, se mostrará en su página de perfil las estadísticas de consultas que ha diagnosticado en total, correcta, e incorrectamente.

Para que ninguna consulta esté indefinidamente sin diagnosticar y las asignaciones se hagan de la forma más efectiva posible, se ha diseñado un algoritmo pseudoaleatorio que permitirá una asignación óptima de las consultas a los médicos de cabecera.

Este algoritmo consta de tres factores:

- Un número aleatorio que es generado en la creación de la consulta y regenerado con cada edición o diagnóstico ordenado de forma ascendiente o descendiente según otro número aleatorio generado en el momento de la asignación.
- La fecha de creación de la consulta.
- El número de diagnósticos totales que ha recibido una consulta.

Si bien será efectivo en el momento que haya múltiples consultas y múltiples usuarios, durante los inicios del rodaje de la aplicación es muy posible que a un médico de cabecera se le repita una misma consulta, para sobrellevar esto entra en juego la última historia de este Sprint.

Al realizarse la asignación de una consulta, a la imagen se le aplicará una rotación aleatoria, de forma que el médico de cabecera no vea una consulta exactamente igual que otra que ya diagnosticó hace unas horas.

6.7.2 Explicación del Sprint

Este sprint se ha logrado realizar en el tiempo estimado a pesar de haber necesitado una modificación en varios puntos de la aplicación.

Para la pantalla de estadísticas se ha utilizado un «Fragment» que será embebido dentro del «Fragment» del perfil de usuario siempre que este posea el rol de médico general. Se ha reutilizado el gráfico circular de los diagnósticos estimados, pero modificando el estilo ya que aporta la visualización necesaria para este tipo de estadística.

Ilustración 18. Pantalla de perfil de un usuario con rol de médico general donde, además de editar sus datos, puede consultar sus estadísticas generales.

Debido a la generación del número aleatorio para la asignación de consultas, se han tenido que modificar las funciones de creación, edición y diagnóstico de consultas, pero gracias a la abstracción de las llamadas a la

base de datos y separación de las distintas casuísticas en varios métodos, esta implementación no ha requerido de una gran refactorización del código.

6.8 Sprint 8

En este Sprint se inicia el desarrollo de la parte web para gestión y visualización de la información que contiene la aplicación Android (Véase Tabla 16).

Tabla 16. Historias extraídas del backlog por implementar en el Sprint 8.

ID	Sprint	Descripción	Prioridad	P. historia
29	8	Estructura de ficheros aplicación web.	1	3
30	8	Integración Firebase en la aplicación web.	1	2
31	8	Creación interfaz cuadro de mando.	2	2
32	8	Creación interfaz descarga bulk.	1	1

6.8.1 Explicación del Sprint

Para el caso de la aplicación web se utilizará una arquitectura basada en Modelo Vista Controlador con «Stores».

Los «Stores» son la parte que se encarga de almacenar múltiples instancias de un modelo y ejecutar las comunicaciones con las APIs pertinentes, en este caso, como se usará Firestore utilizando su SDK, el modelo quedará únicamente para rellenar tablas, desplegables, etc.

La aplicación web se desarrollará como SPA (Single Page Application) de forma que siempre estén a la vista el botón de ayuda y cerrar sesión y los distintos módulos sean cargados dentro de un contenedor.

En cuanto al desarrollo de las interfaces, la de cuadro de mando quedará dividida en dos partes, una para gráficas y otra para la información en bruto en formato tabla. La de descarga Bulk será mucho más sencilla puesto que a nivel visual únicamente requerirá de dos filtros de fecha y un botón de filtro.

6.8.2 Retrospectiva del Sprint

El desarrollo de este Sprint ha sido bastante fluido y no ha presentado parón alguno. La creación del proyecto se realiza con un comando y a partir de

ahí solo quedaría crear los ficheros necesarios y eliminar los que no hagan falta de los que vienen de ejemplo.

Algo parecido ocurre con la integración de Firebase. Siguiendo los pasos que se exponen en su web la implementación se realiza sin complicaciones. No obstante, y puesto que algunas claves son privadas, el index.html de la aplicación se ha separado en uno de ejemplo, con las propiedades necesarias, pero sin sus valores, y otra real que no está controlada por git para evitar exponer las claves a otros desarrolladores al subirlas al repositorio remoto.

Las interfaces se han desarrollado con información de prueba para asegurar que no existan fallos que puedan retrasar la implementación de los próximos sprints.

6.9 Sprint 9

Durante este sprint se desarrolla todo lo relativo a la gestión de nuevos usuarios como se muestra en la Tabla 17, esto incluye la modificación y eliminación de los usuarios actuales, así como la creación de nuevos usuarios en Firebase Firestore y Firebase Authentication.

Tabla 17. Historias extraídas del backlog por implementar en el Sprint 9.

ID	Sprint	Descripción	Prioridad	P. Historia
33	9	Creación interfaz gestión de usuarios.	3	3
34	9	Desarrollo pantalla gestión de usuarios	1	4

6.9.1 Explicación del Sprint

La interfaz de la gestión de usuarios se pretende crear de forma que todo quede fácilmente accesible y con unos controles claros e intuitivos. La información de todos los usuarios cargará automáticamente, lo que no supondrá ninguna lentitud debido a la optimización de los componentes de ExtJS para representar grandes cantidades de información.

La interfaz de creación y edición de los usuarios será compartida, incluyendo el campo contraseña en el controlador cuando la acción sea crear un nuevo usuario.

Al crear un usuario, no sólo se debe dar de alta en la base de datos para poder almacenar sus estadísticas e información personal, también se debe dar de alta en Firebase Authentication para que sea capaz de iniciar sesión en la aplicación Android.

Lo último para tener en cuenta, es la selección de rol puesto que, aunque los usuarios con el rol de médico general se pueden registrar directamente desde la aplicación Android, tener un selector de rol permitiría aligerar algunos procesos que impliquen la creación de múltiples usuarios por ejemplo para una clase o congreso.

Con un desplegable de selección de rol también se incluye de forma intrínseca un punto de extensibilidad al sistema ya que queda preparado para implementar nuevos roles en caso de crecimiento de la aplicación.

6.9.2 Retrospectiva del Sprint

Las historias de usuario incluidas en este sprint se han desarrollado utilizando un tiempo superior al estimado. Si bien la interfaz no ha representado ninguna dificultad, al implementar la funcionalidad de creación de usuario el sistema fallaba.

Esto es debido a que Firebase Authentication no está orientado a una administración de usuarios por terceros, si no al registro de estos. Al crear un usuario lo que provocaba es que automáticamente iniciase sesión, echando al usuario real que estaba realizando la administración.

La solución ha sido crear una segunda instancia de la clase firebase en el momento de la creación del usuario, de esta forma el usuario es creado con esa instancia que luego se desecha por lo que el nuevo usuario no inicia una sesión y el creador mantiene la suya propia.

6.10 Sprint 10

En este sprint se desarrolla la descarga bulk de la información y la parte del cuadro de mando correspondiente a la información en bruto (Véase Tabla 18).

Tabla 18. Historias extraídas del backlog por implementar en el Sprint 10.

ID	Sprint	Descripción	Prioridad	P. Historia
35	10	Desarrollo pantalla descarga bulk.	1	8
36	10	Desarrollo cuadro de mando en versión tabla.	1	5

6.10.1 Explicación del Sprint

Para la descarga bulk se deberán descargar únicamente las imágenes de las consultas con la siguiente nomenclatura: «UCAM_diagnostico_hash.extension». Estas imágenes serán descargadas dentro de un archivo comprimido.

El diagnóstico que se mostrará variará dependiendo de la información que contenga la consulta, de forma que se seleccionará el diagnóstico del patólogo si lo tiene y, si no, seleccionará el del especialista.

Puesto que debido a esto pueden coincidir múltiples nombres de imágenes, se añade un hash generado de forma aleatoria al final del nombre del archivo.

La historia para el desarrollo de la descarga bulk se estimó costosa en un primer momento debido al número de promesas anidadas que supone la descarga de las imágenes. Los pasos para descargar las imágenes son los siguientes:

1. Se realiza una petición para obtener todas las consultas diagnosticadas en el periodo de tiempo introducido.
2. Si devuelve resultados, se recorre cada consulta obteniendo su id y el diagnóstico correspondiente.
3. Se busca en Firebase Storage una **carpeta** con el id de la consulta y se listan sus ficheros.
4. Por cada fichero se genera la url de descarga para obtener la imagen a máxima calidad.

5. Se descarga cada imagen a través del enlace obtenido.

Esto genera iteraciones y promesas anidadas que son difíciles de gestionar debido a que hasta que no se descarguen todas las imágenes no se puede generar el archivo comprimido para entregarlas al usuario.

En cuanto al cuadro de mando es necesario crear una clase «Indicator» que almacene el nombre del indicador, su valor total y sus mediciones estadísticas si las tiene (media, varianza, etc). Estos indicadores serán almacenados en un store de ExtJs cuando se descarguen de Firebase Firestore y serán cargados como filas de la tabla.

6.10.2 Retrospectiva del Sprint

Este sprint ha presentado una dificultad considerable a la hora de obtener las imágenes debido a los puntos expuestos anteriormente. No obstante, esta dificultad inicial ha permitido dejar el sistema preparado para, en un futuro, poder aceptar múltiples imágenes por consulta al crear una carpeta por cada una en lugar de un archivo.

Finalmente se ha conseguido descargar el archivo comprimido con la nomenclatura indicada anteriormente. Se puede observar el resultado en la Ilustración 19.

Ilustración 19. Archivo comprimido con todas las imágenes diagnosticadas en cierto intervalo de tiempo.

Se han introducido validaciones en todo el proceso que permiten avisar al usuario si las fechas introducidas no son correctas o no se encuentran consultas en dichas fechas. Además, el usuario será retroalimentado mientras se ejecuta el proceso de descarga mediante la inclusión de mensajes que indican la parte en la que se encuentra el procesamiento.

Para el cuadro de mando se ha dejado un área menor de la pantalla (Véase Ilustración 20) para mostrar la información en bruto y centrarl en la visualización de las gráficas que se implementarán en futuros sprints.

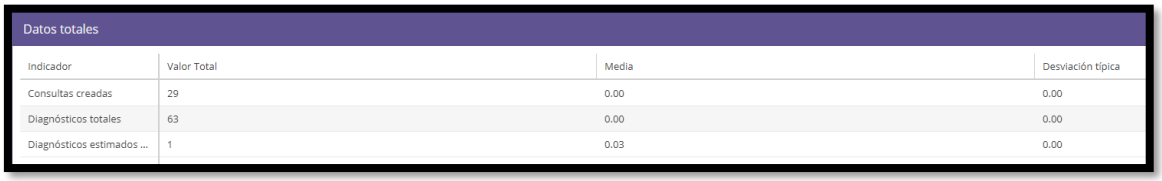

Ilustración 20. Cuadro de mando DermalCheck Web donde se muestran los datos estadísticos en bruto.

6.11 Sprint 11

En este Sprint se desarrollarán las gráficas que muestra el sistema, así como la implementación de inicio y cierre de sesión. Últimas funcionalidades esperadas para la aplicación web, descritas en la Tabla 19.

Tabla 19. Historias extraídas del backlog por implementar en el Sprint 11.

ID	Sprint	Descripción	Prioridad	P. Historia
37	11	Implementación de gráficas en cuadro de mando.	1	7
38	11	Implementación de inicio y cierre de sesión.	3	3

6.11.1 Explicación del Sprint

ExtJS contiene un paquete para gráficas que consiste en una jerarquía de clases que proporcionan funcionalidad de visualización de datos, incluyendo series, ejes, interacciones y mecanismos para implementar marcadores y leyendas (Sencha).

Esta librería se utilizará para presentar dos gráficas a partir de las estadísticas recogidas por la aplicación. La primera será una gráfica de columnas mostrando los datos totales de las estadísticas, esto permitirá tener una visualización rápida de la información en bruto que se muestra actualmente en la tabla. La segunda será un gráfico circular para visualizar el número de consultas por cada tipo de diagnóstico, esto permitirá saber si hay un exceso o escasez de consultas con un diagnóstico concreto que pudiese alterar el futuro entreno de nuevos modelos convolucionales.

En cuanto al inicio y cierre de sesión, se utilizará, de igual forma, Firebase Authenticator, almacenando en la caché de Firebase el usuario identificado para que no se cierre la sesión al cambiar o cerrar la pestaña.

6.11.2 Retrospectiva del Sprint

La implementación de las gráficas ha requerido la inclusión de un paquete nuevo que no viene directamente importado en los nuevos proyectos de ExtJs debido a que añade algo de tiempo de carga a las webs, no obstante, como la aplicación se ha desarrollado como SPA (Single-Page Application), el usuario solo deberá esperar a que cargue la página la primera vez que accede, antes de ver unas gráficas como las que se muestran en la Ilustración 21.

Ilustración 21. Gráficas mostradas en la aplicación web de DermalCheck bajo la pestaña de Cuadro de Mando.

Una vez implementadas, las gráficas se cargan automáticamente al acceder y representan la información de forma clara, permitiendo, además, la interacción del usuario con efectos o movimiento.

El acceso por usuario y contraseña se ha diseñado de forma que se muestre como única ventana al acceder al sistema. Como se puede contemplar en la Ilustración 22, una vez identificado, se mantendrá la sesión abierta hasta que el usuario la cierre, independientemente de que cierre la pestaña, navegador o reinicie su equipo.

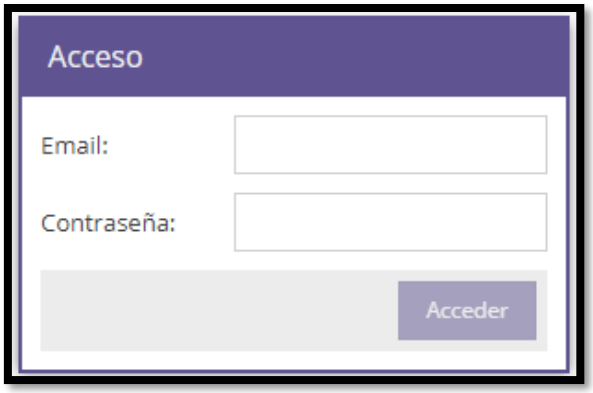

Ilustración 22. Ventana de identificación mediante email y contraseña de la aplicación web de DermalCheck.

6.12 Sprint 12

Este último Sprint, compuesto por las historias mostradas en la Tabla 20, queda reservado a la realización de múltiples pruebas, tanto de la aplicación Android como de la Web, y la solución a cualquier posible error que se haya podido pasar por alto.

Tabla 20. Historias extraídas del backlog por implementar en el Sprint 12.

ID	Sprint	Descripción	Prioridad	P. Historia
39	12	Pruebas aplicación Android y solución de posibles incidencias	1	5
40	12	Pruebas aplicación Web y solución de posibles incidencias	1	4

6.12.1 Explicación del Sprint

La revisión de la aplicación Android se basa en pruebas de la aplicación completa en varios dispositivos físicos para comprobar como funcionaría en un entorno real. Esto permite asegurar que la aplicación desarrollada funciona de igual forma independientemente del tamaño o capacidades de los dispositivos.

En cuanto a la revisión de la aplicación web, se hace algo más sencilla puesto que únicamente requiere la revisión en distintos navegadores con tamaños de pantalla de diferentes resoluciones, sin necesidad de tener que adaptarse a dispositivos móviles pues no es un requisito de la aplicación.

6.12.2 Retrospectiva del Sprint

En la revisión de la aplicación de Android se encontró un error en ciertos dispositivos físicos que impedía la captura de imágenes a través de la cámara, provocando el cierre de la aplicación. Tras la revisión se observó que era un tema de permisos de acceso al almacenamiento.

También se localizaron ciertos puntos donde la aplicación parecía no hacer nada cuando estaba procesando en dispositivos con menos recursos. Para solventar esto se han implementado imágenes de carga donde se ha requerido.

En cuanto a la aplicación web, también se encontró una pequeña incidencia al cerrar sesión que provocaba que la ventana de identificación se mostrara dos veces. La implementación de la solución fue mucho más sencilla que en la aplicación Android.

7. DESPLIEGUE DE LA SOLUCIÓN

7.1 Aplicación Android

Para el despliegue de la aplicación Android será necesario crear una cuenta de desarrollador en Google Play desde su web[1] abonando la tasa correspondiente de 25€ a fecha de este documento. Una vez ahí será posible crear una aplicación en Play Store mediante el botón «Crear aplicación» (Véase Ilustración 23).

Ilustración 23. Botón de creación de nueva aplicación Android en el panel de usuario de Google Play Console.

Una vez pulsado se mostrarán distintas pantallas de formularios secuenciales que habrá que ir rellenando con imágenes y datos de la aplicación, paso que únicamente habrá que realizar una vez.

Al terminar de rellenar la información, se abrirá un panel de la aplicación en el que se mostrará un botón con el título «Crear versión» (Véase Ilustración 24).

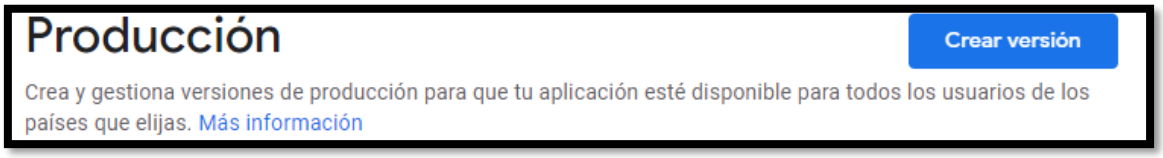

Ilustración 24. Botón de creación de versión en Google Play dentro del panel de administración de una aplicación existente.

Tras pulsar este botón, se mostrará una ventana para rellenar los cambios de la versión y se solicitará la subida de un archivo APK. Este archivo

[1] https://play.google.com/apps/publish

se generará desde Android Studio, bajo la sección «Build->Generate Signed Bundled / APK» como se puede ver en la Ilustración 25.

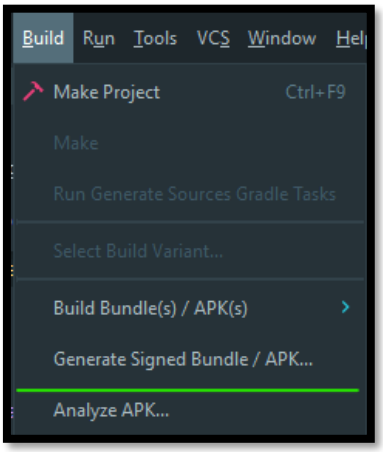

Ilustración 25. Generación de APK firmada en Android Studio.

Una vez aquí, si es la primera vez que se genera este archivo, se solicitará una ruta y contraseña para el almacén de claves (Véase Ilustración 26). El almacén de claves es un archivo que contiene las claves con las que se firma la aplicación. Si se pierden no se podrán generar las mismas claves de nuevo y Google Play no permitirá subir nuevas actualizaciones de la aplicación al no reconocer la firma.

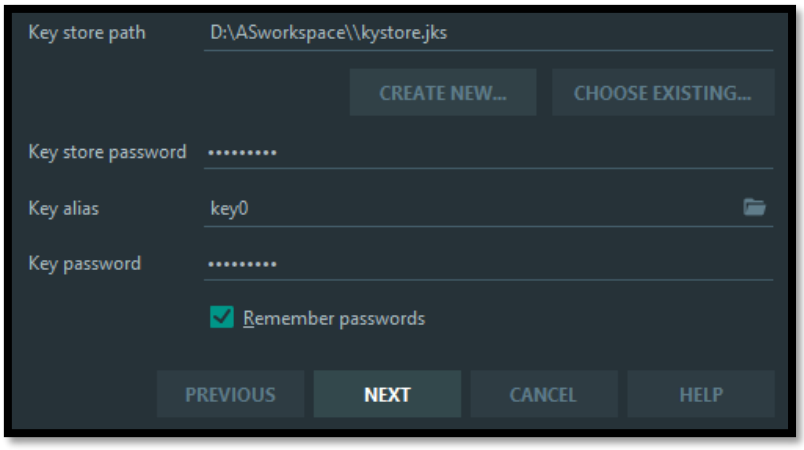

Ilustración 26. Firma de APK desde Android Studio, el archivo de almacén de firma y la contraseña se deben almacenar de forma que no se pierda, o la aplicación no podrá recibir actualizaciones en Google Play.

Por último, se arrastra el APK generado a la web de Play Console y se pulsa sobre el botón «Iniciar lanzamiento a Producción».

A excepción de los pasos de creación de la aplicación en la Play Store, todos los pasos descritos han de ser realizados cada vez que se quiera lanzar una nueva versión de la aplicación, además, hay que incrementar el número de versión y la etiqueta que se encuentran en el archivo «build.gradle», como se puede observar en la Ilustración 27.

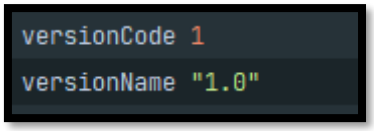

Ilustración 27. Códigos de versión aplicación Android localizados en el archivo «build.gradle» de toda aplicación Android.

7.2 Aplicación Web

El despliegue de la aplicación web se hará a Firebase Hosting y el proceso es algo más sencillo que el de la aplicación Android. En primer lugar. será necesario acceder al apartado Hosting en el proyecto de Firebase y pulsar sobre el botón «Get Started».

A continuación, se mostrará una ventana como la de la Ilustración 28, con tres pasos a seguir para configurar el proyecto con Firebase Hosting. Hay que seguirlos al pie de la letra, seleccionando la carpeta del proyecto «build/production» cuando se pregunte por la carpeta que contiene los archivos a desplegar.

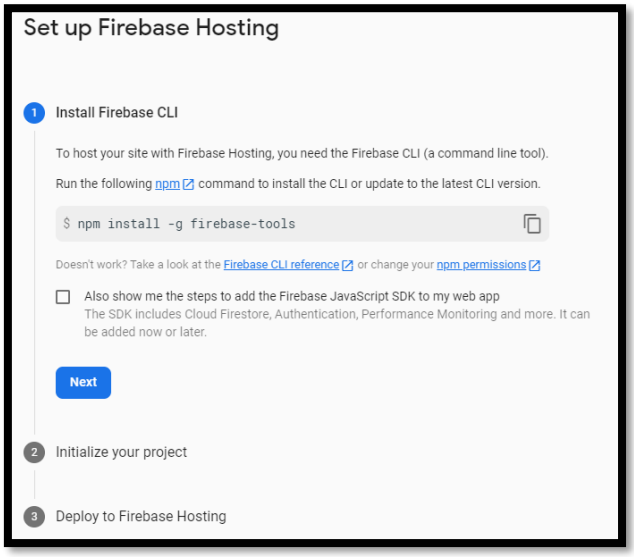

Ilustración 28. Puesta en marcha Firebase Hosting en un nuevo proyecto.

Cuando los pasos se hayan realizado, se puede asegurar que el proceso ha sido correcto si al ejecutar el comando «firebase --version» se muestra un mensaje con un código de versión similar al de la Ilustración 29.

Ilustración 29. Comprobación instalación firebase-cli mediante un comando simple que devuelve la versión que se está ejecutando de firebase en ese momento.

Por último, solo habría que ejecutar el comando «sencha app build», para construir la versión minimizada y ofuscada de la aplicación y, el comando «firebase deploy» para hacer el despliegue en sí mismo. Estos dos comandos serán los únicos que necesiten ejecutarse una vez puesto en marcha todo lo anterior, cada vez que se quiera publicar una actualización.

Si todo ha salido bien, se mostrará una salida similar a la de la Ilustración 30.

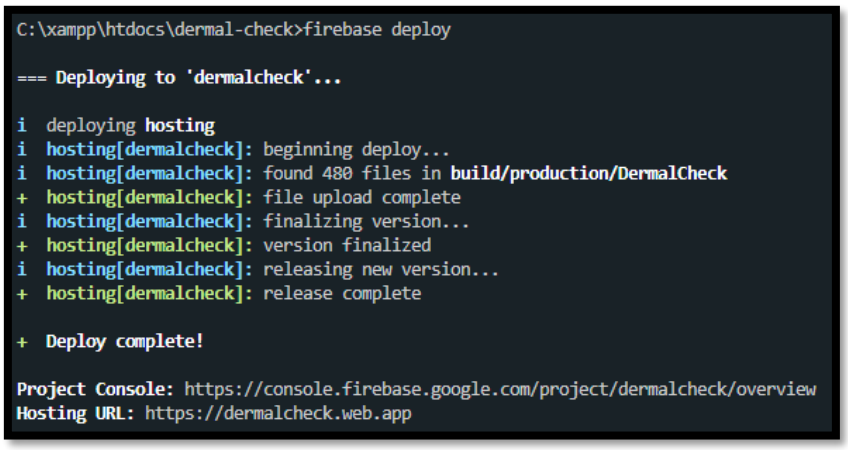

Ilustración 30. Despliegue exitoso de la aplicación web Dermalcheck mediante el uso del comando «firebase deploy».

7.3 Escalabilidad

7.3.1 Aplicación Android

La aplicación Android se ha desarrollado utilizando una arquitectura MVVM, cuyo esquema se expone en la Ilustración 31, lo que ha permitido

separar cada componente de la aplicación en clases separadas, de forma que cada componente depende única y exclusivamente de aquel que tiene debajo.

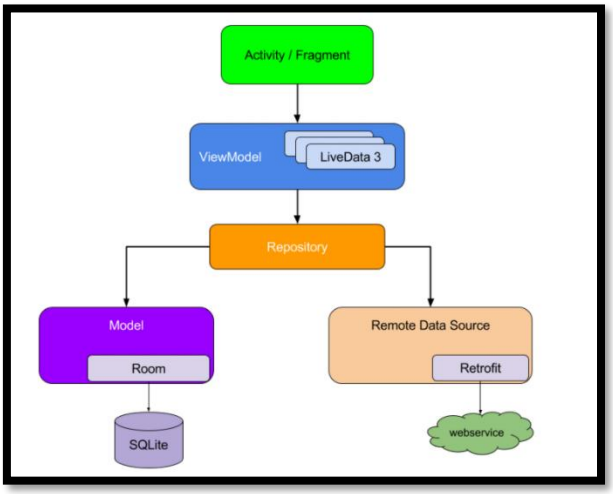

Ilustración 31. Arquitectura recomendada de aplicaciones Android (Google).

En este caso el repositorio sería sustituido directamente por el «datasource» al no permitir que la información se almacene en el dispositivo. Si en el futuro fuera un requisito, no habría que hacer más que implementar una clase intermedia entre «Datasource» y «ViewModel».

Además, la obtención de datos se ha modularizado de forma que sería posible cambiar la fuente de datos en un momento dado para que, en lugar de Firebase, la información se obtuviese de una API REST, o incluso de información almacenada en la base de datos del dispositivo.

En cuanto a la escalabilidad del modelo de datos, al utilizar Firebase Firestore se tiene la posibilidad de añadir nuevos campos o colecciones sin necesidad de alterar previamente la estructura de la base de datos lo que, además de permitir una alta escalabilidad, reduce los puntos de falla originados por este tipo de modificaciones.

7.3.2 Aplicación Web

En este caso la arquitectura utilizada ha sido MVC con el objetivo de ser igualmente escalable y mantenible.

Todas las interfaces han sido desarrolladas con independencia unas de otras, lo que permitiría su ocultación, movimiento, combinación con cualquier

otra interfaz sin tener que modificar ninguno de sus componentes, al igual que ocurre con cada uno de sus controladores.

En caso de incluir nuevos componentes, toda la definición de archivos se hace desde un único fichero por lo que no habría que recorrer cada uno haciendo su importación y, una vez creada una nueva interfaz, será cargada automáticamente junto con todas sus dependencias.

7.4 Plan de formación de usuarios

Para el plan de formación de usuarios se ha de diferenciar entre un usuario típico de las aplicaciones y un usuario desarrollador que desee instalarse las herramientas necesarias para poder contribuir al proyecto.

Para este último caso, todo lo necesario para poner en marcha los entornos de desarrollo locales están disponibles en los anexos A, B, C, D y E de este documento.

En cuanto a los usuarios convencionales cuyo objetivo es hacer un uso normal de la aplicación existen manuales de uso separados por aplicación y rol en los anexos E, F y G de este documento.

8. CONCLUSIONES

8.1 Objetivos alcanzados

Para poder valorar si los objetivos propuestos se corresponden con los alcanzados será necesario referenciar a apartados anteriores y validar en líneas generales si los desarrollos realizados cubren las necesidades impuestas en un primer momento.

Los objetivos principales del proyecto en lo referente a la aplicación Android han sido ampliamente cubiertos pues se ha desarrollado un ciclo de vida completo de creación, edición y diagnóstico de consultas gestionadas mediante roles con las estimaciones del modelo de inteligencia artificial pre entrenado provisto. Además, la aplicación es capaz de calcular y almacenar información estadística en distintos puntos de los procesos. Por último, toda la información es recogida, enviada, obtenida y almacenada de los distintos servicios que ofrece la plataforma Firebase.

Por la parte de la aplicación web, los objetivos también han sido abordados con éxito puesto que se comunica y comparte la información con la aplicación Android, siendo capaz de descargar las imágenes diagnosticadas y permite una visualización de los datos estadísticos y la gestión completa de los usuarios.

Por último, las aplicaciones han sido controladas por Git en cuanto a versiones se refiere y la web ha sido desplegada en Firebase Hosting con éxito por lo que, en la parte técnica, también se han cumplido los objetivos previstos.

Finalmente, el objetivo a largo plazo queda validado con la consecución de todos los anteriores. Esta aplicación permitirá recopilar múltiples imágenes dermatoscópicas diagnosticadas que podrán formar un dataset para el entrenamiento de futuras redes neuronales que permitan acelerar el proceso de detección de posibles lesiones, además de ayudar a los médicos de cabecera a atender mejor y con más celeridad a sus pacientes.

8.2 Conclusiones del trabajo y personales

En lo personal, este proyecto me ha ayudado a implicarme más con una amplia gama de tecnologías y plataformas que no había utilizado o lo había hecho muy por encima y me ha dado una visión superficial pero interesante sobre el ámbito que envuelve.

El desarrollo Android con Java lo había utilizado con anterioridad durante muchos años, no obstante, el proyecto me ha ayudado a ponerme al día con las nuevas arquitecturas, cambios en las versiones más recientes de los dispositivos y comunicaciones con los distintos servicios de Firebase.

Y hablando de Firebase, sólo lo había utilizado en algunas aplicaciones muy concretas. Con este proyecto he podido conocer a fondo la mayoría de sus módulos, bien porque al inicio quería informarme de qué debía usar o bien porque empecé a usarlas en un primer momento, aunque no llegasen a la versión final. Me ha permitido ver que se puede desarrollar una aplicación Android centrándose en la propia aplicación y no en todo el desarrollo que conlleva una API REST, base de datos, infraestructura, etc.

Toda la parte de integración con la inteligencia artificial me ha ayudado a profundizar, ya no solo en TensorFlow y tecnologías concretas, si no en todos los conceptos que envuelven el aprendizaje automático, supervisado, no supervisado, etc.

Además, con toda la ayuda que me han brindado mis tutores Andrés Bueno y Francisco Arcas, he sido capaz de solventar cualquier obstáculo técnico y conocer cómo debe ser una aplicación técnicamente para que sea efectiva y acogida por el usuario final. Por último, también tengo que agradecer a Fernando Alarcón, dermatólogo adjunto del Complejo Hospitalario Santa Lucía – Rosell, su actuación como cliente, y usuario final, que me ha permitido comprender mejor cuáles son las funcionalidades que necesita una aplicación de este tipo y cómo funciona el proceso de detección de lesiones de piel.

8.3 Vías futuras

A continuación, se enumeran una serie de puntos de extensión que ofrecen las aplicaciones en distintos ámbitos:

- Tabla de clasificaciones para incentivar a los usuarios con rol de médico general a diagnosticar más consultas para mejorar así sus capacidades.

- División de las consultas a diagnosticar en niveles en base a la seguridad del diagnóstico del especialista y el porcentaje de predicción de la inteligencia artificial.

- Nuevas estadísticas: aciertos y fallos por tipo de diagnóstico, usuarios más activos, etc.

- Posibilidad, un médico general, de solicitar una explicación detallada al especialista en caso de fallo en un diagnóstico del que estaba convencido.

- Para la aplicación web, exportación de las tablas a Excel y de las gráficas a PDF o imagen.

- Para la aplicación web, nuevos filtros en la pantalla de descarga Bulk o envío automatizado de nuevas consultas diagnosticadas por email o subida a un servidor FTP.

9. BIBLIOGRAFÍA

Bagnato, j. I. (29 de Noviembre de 2018). *¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador.* Obtenido de Aprende Machine Learning: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Bengio, Y., Courville, A., & Pasval, V. (13 de Marzo de 2013). Representation Learning: A Review and New Perspectives. *IEEE Trans*, págs. 1798-1828.

Brownlee, J. (2 de Enero de 2019). *Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates.* Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/

de Areba, J. B. (2001). *Metodología del análisis estructurado de sistemas (Vol. 20).* Univ. Pontifica Comillas.

Dongping, D., & Wanli, X. (2018). Dropout Prediction in MOOCs: Using Deep Learning for Personalized Intervention. *Journal of Educational Computing Research*, 073563311875701.

Facebook. (29 de Mayo de 2013). *React*. Obtenido de React: react.org

Figueroa, R. G., Solís, C. J., & Cabrera, A. A. (2008). *Metodologías tradicionales vs. metodologías ágiles.* Loja: Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación.

Garzas, J. (8 de Jului de 2013). *El Sprint cero y el Sprint de release. Javier Garzas*. Obtenido de Javier Garzas: https://www.javiergarzas.com/2013/07/el-sprint-cero-y-el-sprint-de-release.html

Garzas, J. (13 de Marzo de 2019). *Buscando desperdicios Ágiles: los Puntos Historia. Javier Garzas*. Obtenido de Javier Garzas: https://www.javiergarzas.com/2019/03/buscando-desperdicios-agiles-los-puntos-historia.html

Git. (s.f.). *Git.* Obtenido de Git: https://git-scm.com/

Google. (14 de Septiembre de 2016). *Angular*. Obtenido de Angular: angular.io

Google. (Mayo de 2017). *Flutter*. Obtenido de Flutter: https://flutter.dev/

Google. (s.f.). *Guía de arquitectura de apps. Android Developers.* Obtenido de Android Developers: https://developer.android.com/jetpack/guide?hl=es-419#recommended-app-arch

JetBrains. (2016). *Sobre Kotlin. Kotlin*. Obtenido de Kotlin: https://kotlin.es/sobre-kotlin/

Joskowicz, J. (2008). *Reglas y prácticas en eXtreme Programming.* Vigo: Universidad de Vigo.

Kaur Grill, J. (27 de Agosto de 2017). *Automatic Log Analysis using Deep Learning and AI.* Obtenido de XenonStack: https://www.xenonstack.com/blog/log-analytics-deep-machine-learning/

L, J. R. (2015). *Desarrollo de software ágil: Extreme Programming y Scrum.* IT Campus Academy.

Larranaga, P., & Moujahid, A. (2021). *Tema 8. Redes Neuronales ResearchGate.* Obtenido de ResearchGate: https://www.researchgate.net/publication/268291232_Tema_8_Redes_Neuronales

Mahnic, V. (2011). A Case Study on Agile Estimating and Planning using Scrum. *Elektronika ir Elektrotechnika*, 123-128.

MathWorks. (s.f.). *Convolutional Neural Network: MathWorks.* Obtenido de MathWorks: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html

MetaOptima. (Abril de 2013). What is DermEngine? Naarden, Naarden, Netherlands.

MongoDB, Inc. (2009). *What Is MongoDB?* Obtenido de MongoDB: https://www.mongodb.com/what-is-mongodb

Montero, B. M., Cevallos, H. V., & Cuesta, J. D. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales revista multidisciplinaria de investigación*, 113-121.

Open JS Foundation. (27 de Mayo de 2010). *NodeJs*. Obtenido de NodeJS: https://nodejs.org/

Oracle. (2010). *MySQL*. Obtenido de MySQL: https://www.mysql.com/

P. Guy, G., R. Machlin, S., U. Ekwueme, D., & Robin Yabroff, K. (2015). Prevalence and Costs of Skin Cancer Treatment in the U.S., 2002−2006 and 2007−2011. *American Journal of Preventive Medicine*, 183-187.

Proagilist . (22 de Agosto de 2016). *La Velocidad en el Sprint de Scrum.* Obtenido de Proagilist : https://proagilist.es/blog/agilidad-y-gestion-agil/la-velocidad-sprint-scrum/

Qi, H. (2016). *Derivation of Backpropagation in Convolutional Neural Network (CNN).*

Refactoring UI. (s.f.). *Building Your Color Palette. Refactoring UI.* Obtenido de Refactoring UI.: https://refactoringui.com/previews/building-your-color-palette/

Sencha. (s.f.). *ExtJs Sencha*. Obtenido de Sencha: https://www.sencha.com/products/extjs/

Sencha. (s.f.). *Introduction to Charting. ExtJS 7.0.0.* Obtenido de ExtJS 7.0.0: https://docs.sencha.com/extjs/7.0.0/guides/components/introduction_to_charting.html

Shanhong, L. (Diciembre de 2020). *Ranking of the most popular database management systems worldwide, as of December 2020*. Obtenido de Statista: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/#:~:text=As%20of%20December%202020%2C%20the,rounded%20out%20the%20top%20three.

TensorFlow. (s.f.). *Save and Load models. TensorFlow Tutorials*. Obtenido de TensorFlow: https://www.tensorflow.org/tutorials/keras/save_and_load

The PHP Group. (2001). *What Is PHP? PHP*. Obtenido de PHP: https://www.php.net/manual/en/intro-whatis.php

Tschandl, P. (2018). *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Hardvard Dataverse*. Obtenido de Hardvard Dataverse: https://doi.org/10.7910/DVN/DBW86T

W. Grenning, J. (Abril de 2002). *Planning Poker or How to avoid analysis paralysis while release planning.* Obtenido de Wingman Software: https://wingman-sw.com/papers/PlanningPoker-v1.1.pdf

Warden, P. (14 de Diciembre de 2017). *How many images do you need to train a neural network?* Obtenido de Pete Warden's blog: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/

10. ANEXOS

10.1 Manuales de instalación de herramientas de desarrollo

Anexo A. Instalación de Android Studio

En primer lugar, hay que descargar el paquete de instalación desde la web oficial[2].

Una vez descargado el fichero el proceso de instalación es muy sencillo puesto que no requiere de ninguna opción especial en lo que a este proyecto se refiere, por lo que bastará con pulsar el botón de siguiente hasta que comience el proceso de instalación.

Al completarse la instalación se abrirá una ventana como la mostrada en la Ilustración 32 para iniciar un nuevo proyecto o abrir uno ya existente. Se podrá seleccionar la opción «Get from version control», para el caso de que el proyecto haya sido transmitido mediante el enlace a un repositorio remoto o la opción «Open an Existing Project» para abrir una carpeta como proyecto existente.

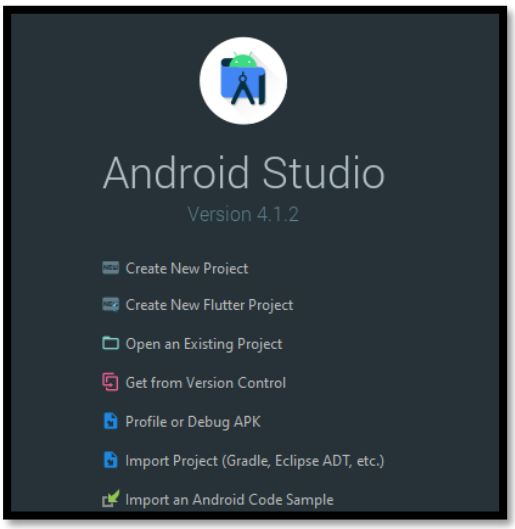

Ilustración 32. Ventana de bienvenida de Android Studio en la que se pueden importar proyectos desde múltiples orígenes.

[2] https://developer.android.com/studio

Al pulsar en «Open an Existing Project» se abrirá una nueva ventana para seleccionar la carpeta del proyecto. Una vez seleccionada Android Studio iniciará la indexación del proyecto y, cuando finalice, la importación del proyecto se habrá realizado de forma exitosa.

Por último, si no existe el archivo PrivateConstants.java hay que renombrar el archivo en la ruta donde se encuentra[3] para quitarle el «.example» y rellenar las constantes con los valores indicados en los comentarios.

Anexo B. Instalación de VSCode

La instalación de VSCode es todavía más sencilla que la de Android Studio, simplemente habrá que ir a la web oficial[4] y descargar la versión correspondiente a la del sistema operativo utilizado.

Durante la instalación, habrá un paso, mostrado en la Ilustración 33, en el que aparecerán distintas opciones. Una de ellas permitirá abrir una carpeta como proyecto simplemente pulsando click derecho sobre ella, para una importación todavía más sencilla se procederá a seleccionar dicha opción.

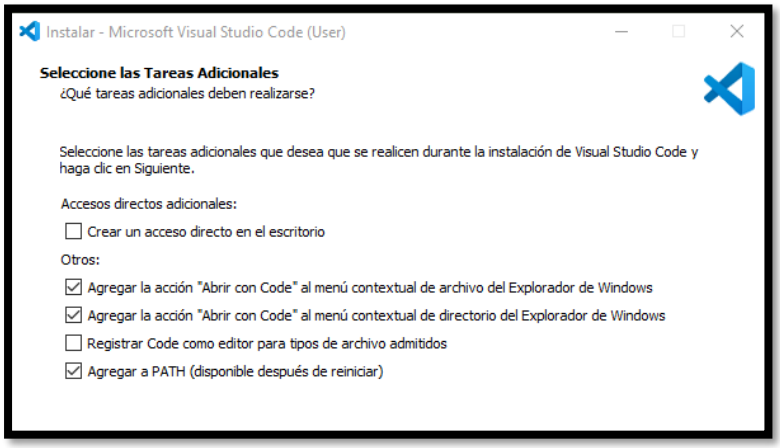

Ilustración 33. La opción «Agregar la acción "Abrir con Code" al menú contextual de archivo del Explorador de Windows» debe estar marcada durante la instalación.

[3] app\src\main\java\com\brugui\dermalcheck\utils\PrivateConstants.java.example

[4] https://code.visualstudio.com/

Al finalizar la instalación solo habrá que dirigirse a la carpeta de Windows donde esté el proyecto, hacer clic derecho y pulsar sobre «Abrir con Code» para importar el proyecto (Véase Ilustración 34).

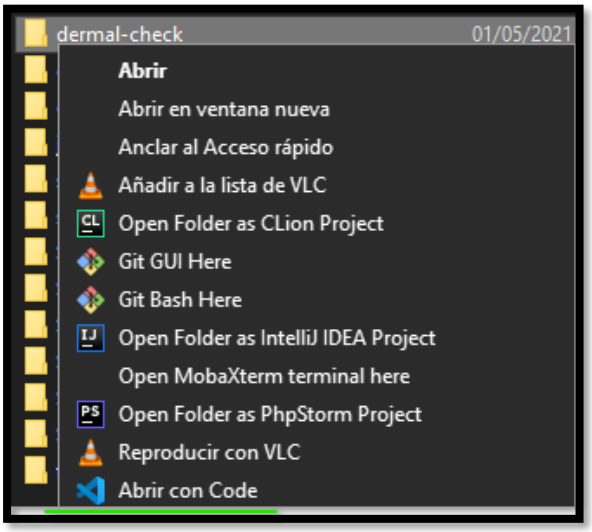

Ilustración 34. En el explorador se podrá abrir directamente el proyecto con VSCode.

Por último, si no existe el archivo «index.html» hay que renombrar el archivo en la ruta «index.html.example» para quitarle el «.example» y rellenar las constantes con los valores indicados en los comentarios.

Anexo C. Instalación de ExtJS + Sencha cmd

Para modificar el proyecto de la aplicación web será necesario instalar el SDK de ExtJS y la utilidad CLI de Sencha, Sencha Cmd para poder desplegar en local y en producción.

Para la instalación de Sencha cmd simplemente habrá que ir a la página oficial[5], descargar la versión correspondiente al sistema operativo y versión de ExtJS a utilizar, en el caso de este proyecto es la 7.0.0.40. Una vez descargado el archivo se extraerá en la carpeta que el usuario desee y, para mayor facilidad, se añadirá la ruta al PATH para poder acceder al comando desde cualquier lugar (Véase Ilustración 35).

[5] https://www.sencha.com/products/extjs/cmd-download/

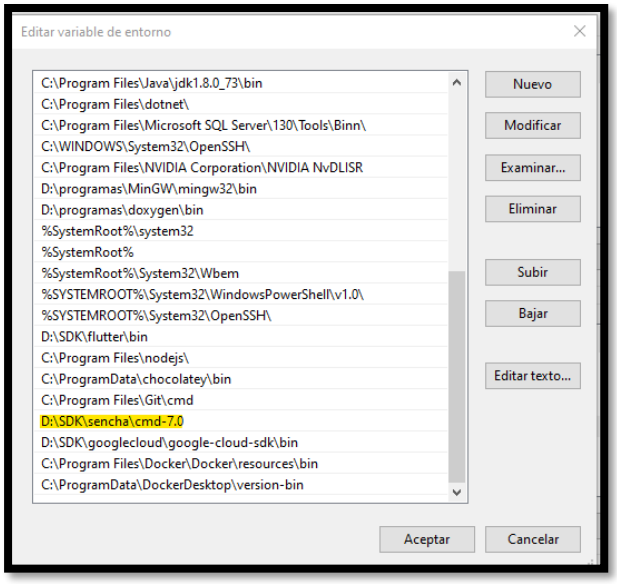

Ilustración 35. Se añade el ejecutable de Sencha cmd al PATH para que el comando esté disponible en cualquier contexto.

El SDK de ExtJS se puede descargar desde la web oficial en el apartado «Community Edition»[6].

La Community Edition es una versión que permite obtener los componentes de ExtJs al completo sin necesidad de liberar el código ni pagar licencia para estudiantes o personas que quieran aprender sobre este Framework. Permite, incluso, utilizar las aplicaciones desarrolladas con fines comerciales siempre que los ingresos totales que genere no superen los 10.000$ anuales. Una vez descargado el archivo, bastará con descomprimirlo en la misma carpeta donde se instaló Sencha cmd.

También es posible instalar Sencha cmd y ExJs en directorios distintos puesto que es posible indicar una ruta al SDK de forma manual con el comando «sencha -sdk ruta/al/sdk».

[6] https://www.sencha.com/products/extjs/communityedition/

Si la instalación ha funcionado correctamente, el comando «sencha» debería mostrar la versión y otra información relevante como se observa en la Ilustración 36.

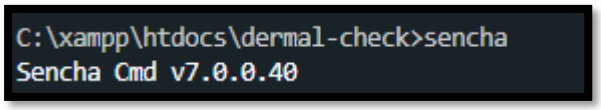

Ilustración 36. Verificación de la instalación de extjs y sencha cmd.

Por último, para lanzar la aplicación web en modo desarrollador se ejecutará el comando «sencha app watch», la salida mostrará en qué dirección se ha desplegado la aplicación en su versión de desarrollo (Véase Ilustración 37).

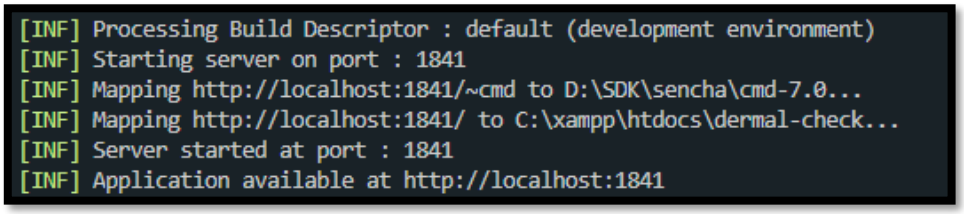

Ilustración 37. La salida del comando «sencha app watch» indicará en qué url y puerto está disponible la versión de desarrollo de la aplicación.

Anexo D. Creación de proyecto en Firebase

Para todas las comunicaciones con base de datos, hosting y almacenamiento será necesario disponer de un proyecto creado en Firebase. Para crear uno nuevo habrá que acceder a la web de Firebase[7] y pulsar sobre «Add project». Esto hará que aparezca una pantalla para establecer el nombre del proyecto.

Al crearlo, el usuario será redirigido a la pantalla de métricas del proyecto recién creado, será aquí donde habrá que importar las aplicaciones

[7] https://console.firebase.google.com/u/0/

Android y web mediante el botón que se muestra en la parte superior «Add app» y seleccionando la plataforma deseada como se muestra en la Ilustración 38.

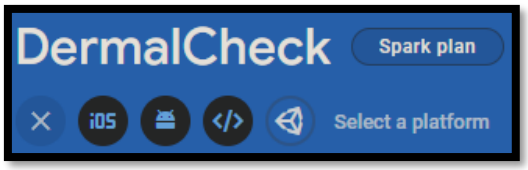

Ilustración 38. Selección de plataformas a añadir en un proyecto de Firebase.

Al seleccionar la plataforma deseada se mostrará una serie de pasos para ponerlo todo a punto. Durante el proceso, será necesario modificar los archivos que contienen las claves y otras configuraciones de Firebase indicados en apartados anteriores:

- **PrivateConstants.java:** Para la aplicación Android.
- **Index.html:** Para la aplicación Web.

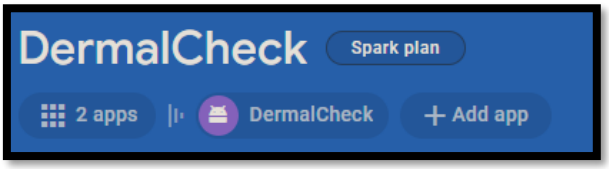

Ilustración 39. Aplicaciones que usan el proyecto Firebase actual, servirá para comprobar si las integraciones se han realizado correctamente.

Si todos los pasos mencionados se realizan correctamente, en la consola de Firebase se podrá observar un indicador que muestra que en el proyecto hay dos aplicaciones integradas (Véase Ilustración 39).

10.2 Manuales de uso

Anexo E. Aplicación Android en rol de especialista

El primer paso que realizará un especialista es identificarse en la aplicación mediante el usuario y contraseña que le han sido provistos, puesto que no posee la potestad de registrarse por sí mismo. Una vez dentro, se mostrará una ventana con todas las consultas que ha creado y un botón para añadir una nueva como se puede observar en la Ilustración 40.

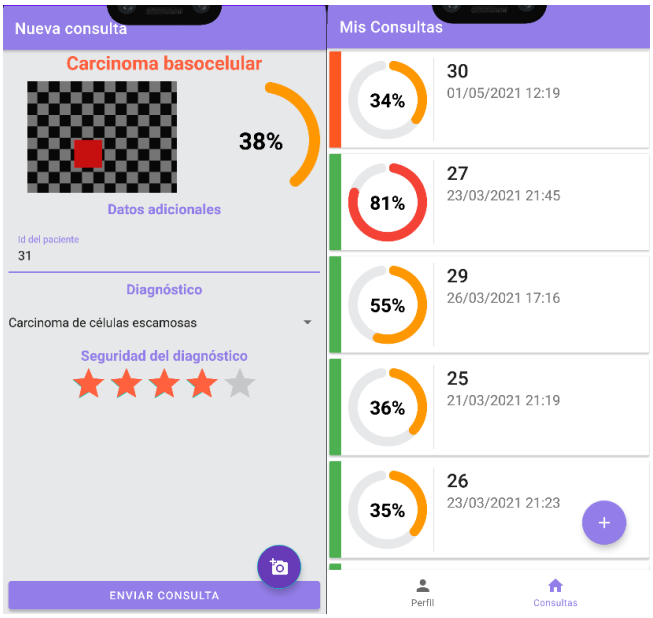

Ilustración 40. Listado y creación de consultas en la aplicación DermalCheck para Android como un usuario con rol especialista.

Al pulsar sobre este botón se abrirá una nueva pantalla en la que introducir foto, mediante subida desde galería o foto, diagnóstico y seguridad con la que se diagnostica. Al pulsar sobre «Enviar consulta», se creará la consulta y el usuario será redirigido a la pestaña principal.

Una vez creada la consulta, será posible acceder a su detalle pulsando sobre ella y añadir información más detallada (Véase Ilustración 41).

Ilustración 41. Pantalla de edición de consultas para el rol especialista.

Por último, desde la pantalla de «Perfil», el médico especialista podrá editar sus datos personales y cerrar sesión como se muestra en la Ilustración 42.

Ilustración 42. Pantalla de Perfil en el rol especialista.

Anexo F. Aplicación Android en rol de médico general

Para acceder a la aplicación, el médico, deberá identificarse con email y contraseña que, si no lo ha hecho con anterioridad, puede establecer registrándose (Véase Ilustración 43).

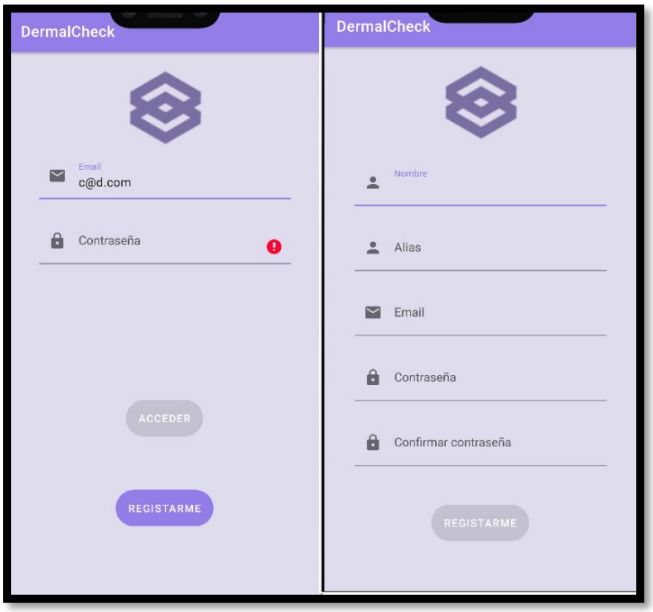

Ilustración 43. Pantallas de identificación y registro de la aplicación Android.

Al acceder, el médico general podrá ver un listado de las consultas que tiene pendientes de diagnosticar, si no tuviese ninguna, el sistema intentaría asignarle una automáticamente. Al pulsar, sobre ella, verá la imagen dermatoscópica y un selector con todos los diagnósticos posibles como se puede ver en la Ilustración 44.

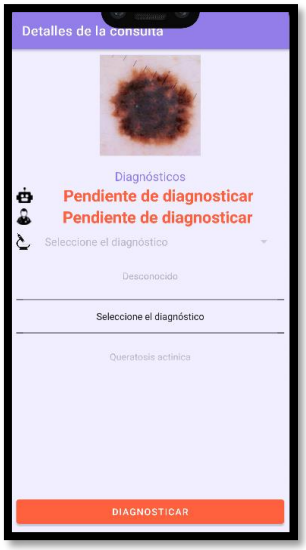

Ilustración 44. Imagen de una consulta sin diagnosticar asignada a un médico general.

Al pulsar el botón diagnosticar, se mostrarán los diagnósticos del especialista, patólogo (si lo hubiese) y el estimado por la inteligencia artificial, junto con una pequeña alerta temporal indicando si se ha acertado o no (Véase Ilustración 45). Tras unos instantes, la aplicación asignará de nuevo otra consulta y se accederá a su pantalla automáticamente.

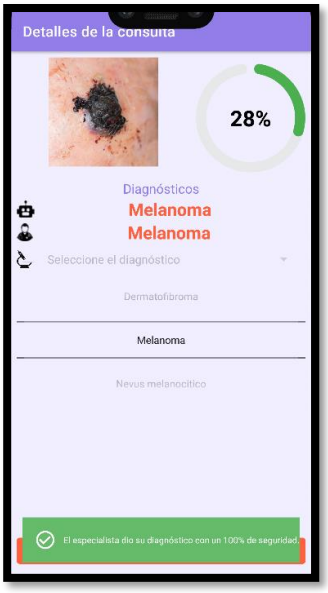

Ilustración 45. Pantalla de consulta ya diagnosticada por parte de un usuario con rol de médico general.

En la pestaña de perfil, el usuario será capaz de modificar su nombre y alias, cerrar sesión y consultar sus estadísticas totales, como se puede apreciar en la Ilustración 46.

Ilustración 46. Pantalla de Perfil de la aplicación Android para un usuario con rol de médico general, donde puede consultar sus estadísticas de forma rápida y visual.

Anexo G. Aplicación web

Para acceder a la web, en primer lugar será necesario identificarse utilizando el usuario y contraseña provistos. Una vez dentro, automáticamente

se mostrará la pestaña de cuadro de mando con las estadísticas generales de la aplicación en formato tabla y gráficas como se muestra en la Ilustración 47.

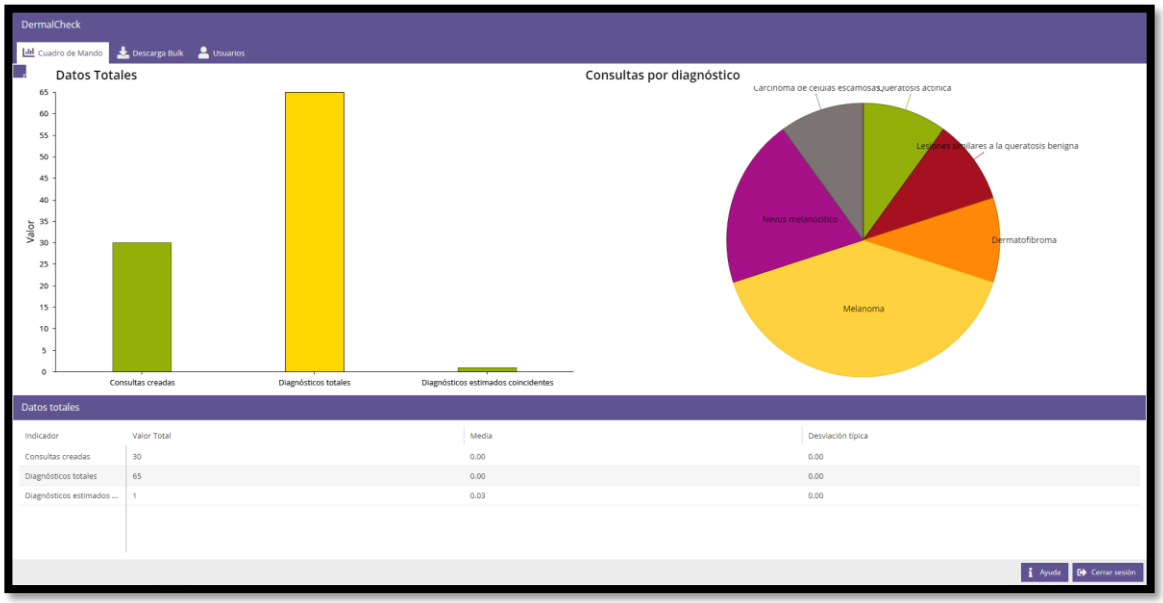

Ilustración 47. Pantalla principal de la página web, la cual cargará la información automáticamente al acceder.

En la parte superior, existen dos pestañas más, una accederá a la pantalla de descarga de las imágenes diagnosticadas mientras que la otra mostrará los usuarios existentes en el sistema a la vez que permitirá su gestión. También se mostrarán siempre en pantalla los botones de cierre de sesión y ayuda, este último mostrará una breve descripción indicando qué función cumple cada pestaña.

En la pestaña de descarga bulk se podrá realizar un filtro por intervalo de fechas para descargar todas las imágenes diagnosticadas en ese intervalo de tiempo. El sistema mostrará una alerta indicando si se están descargando o si, por el contrario, no existen consultas diagnosticadas para las fechas introducidas.

Por último, al acceder a la pestaña de gestión de usuarios, el sistema cargará automáticamente en una tabla todos los usuarios existentes, otorgando

la posibilidad de editarlos, mediante el botón que se muestra en cada fila, crearlos o eliminarlos, todo desde la misma interfaz (Véase Ilustración 48).

Ilustración 48. Interfaz de gestión de usuarios de la aplicación web. Los nombres y correos electrónicos mostrados son ficticios para no incumplir la LOPD.