_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DESARROLLO DE UNA APLICACIÓN ANDROID: ALLERGYHELP

Álvaro Domínguez Castillo

Tutor: José Enrique López Patiño

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 1 de julio de 2016

Resumen

El presente trabajo de fin de grado consiste en el desarrollo de una aplicación cliente-servidor para smartphones con plataforma Android. Se trata de una aplicación con una base de datos que contiene información de los principales productos alimenticios del mercado Español y que identifica qué alimentos puede consumir un determinado usuario, dependiendo de las alergias configuradas. Además la aplicación es capaz de identificar los productos escaneando el código de barras del mismo y de mostrar, mediante un algoritmo de búsqueda, productos relacionados al buscado en caso de no poder ser consumido por el usuario.

Para el desarrollo del servidor se ha hecho uso de un servidor Apache junto con un servidor de base de datos MySQL. Además para la implementación de la aplicación se hace uso de tecnologías como Java, XML, JSON, PHP, Google Maps API, protocolo http, etc.

En último lugar, destacar que la base de datos está pensada para ser alimentada por los usuarios de la aplicación y que estos la hagan crecer a modo de Wikipedia de los alimentos.

Palabras clave: Aplicación móvil, Android, Google Maps API, JSON, PHP.

Resum

El present treball de fi de grau consistix en el desenvolupament d'una aplicació client-servidor per a smartphones amb plataforma Android. Es tracta d'una aplicació amb una base de dades que conte l'informació dels principals productes alimentaris del mercat Espanyol i que identifica quins aliments poden ser consumits per un determinat usuari, depenent de l'al·lèrgia configurada. A més l'aplicació es capaç d'identificar els productes llegint el seu codi de barres i de mostrar, mitjançant un algoritme de recerca, productes relacionats al buscat en cas de no poder ser consumit per l'usuari.

Per al desenvolupament del servidor s'ha fe ús d'un servidor Apache junt amb un servidor de base de dades MySQL. A més per a la implementació de l'aplicació s'han utilitzat tecnologies com Java, XML, JSON, PHP, Google Maps API, protocol http, etc.

Per a acavar, destacar que la base de dades està pensada per a ser alimentada per els usuaris de l'aplicación i que siguen els usuaris els que la fagen creixer a mode de Viquipèdia dels aliments.

Paraules clau: Aplicació mòbil, Android, Google Maps API, JSON, PHP.

Abstract

This final project consists on the development of a client-server application for smartphones with Android platform. It is an application with a database which contains information about the main Spanish food market and identifies which food products could be eatten by an specific user, depending on the configured allergy. Furtermore the application is able to identify the food products scanning the barcode and show, whith a reserch algorithm, food products related to the searched.

For de server development it has been used an Apache server with a MySQL database server. Furthermore, for the application implementation it has been used technologies like Java, XML, JSON, PHP, Google Maps API, http protocol.

Finally, it should be noted that the database has been designed to be fed by the application users.

Keywords: Mobile application, Android, Google Maps API, JSON, PHP.

Índice

1. INTRODUCCIÓN

Desarrollo de una aplicación Android. Allergy Help, una ayuda para el día a día de los alérgicos.

1.1 Contexto socio-económico

En este apartado se pretende realizar un breve análisis del contexto socio-económico de la sociedad, para de esta forma mostrar la viabilidad de la aplicación *AllergyHelp*, así como la justificación del porqué de la aplicación .

1.2 AllergyHelp, una necesidad real

Cuando una persona alérgica o intolerante acude a un supermercado a realizar una compra, debe fijarse con atención en el etiquetado del producto y de este modo evitar aquellos productos que no puede consumir. Esto puede hacer que el simple hecho de realizar la compra resulte complejo, se limite la variedad de productos comprados, además de resultar una gran pérdida de tiempo con respecto a las personas que no padecen ningún tipo de alergia.

Con la voluntad de hacer la vida más sencilla a estas personas mejorar estas dos problemáticas(seguridad y tiempo), nace la idea de crear una aplicación para smartphone que ayude a las personas alérgicas e intolerantes en su día a día. Haciendo más eficiente el tiempo invertido al realizar la compra y ayudando a aumentar la seguridad a la hora de consumir los productos alimenticios.

1.2.1 Análisis del entorno

1.2.1.1 Factores económicos

La crisis mundial de 2008 afectó a la mayor parte de países del mundo, entre ellos España.
El comienzo de esta crisis mundial supuso para España la explosión de otros problemas; el final de la burbuja inmobiliaria, la crisis bancaria de 2010 y finalmente el aumento del desempleo en España.

A pesar de todos estos factores, la venta y penetración de smartphones en España se ha visto escasamente afectada por la crisis*: "España es el país líder europeo en penetración de los smartphones. Y es que el 81% de los móviles que se utilizan en España son teléfonos inteligentes, según se desprende del Informe de la Sociedad de la Información 2014 de Telefónica"*(RTVE.es).

1.2.1.2 Factores socioculturales

Uso de smartphones e internet en nuevas generaciones

Las futuras generaciones están viviendo en una sociedad completamente digital, y cuyos usuarios se inician cada vez mas temprano en el uso de smartphones, *" la disposición de teléfono móvil se incrementa significativamente a partir de los diez años hasta alcanzar el 90,3% en la población de 15 años"*(elEconomista.es).

En el caso de AllergyhHelp, esta situación es muy favorable, debido a que todas y cada una de las generaciones futuras van a saber manejar un Smartphone, lo cuál abre todo un abanico de posibilidades en cuanto a mercado de las aplicaciones se refiere.

Aumento del número de personas alérgicas/intolerantes

"*Las alergias alimentarias en España se han duplicado en diez años, pasando del 3,6% al 6,05% entre 1992 y 2005, especialmente a los frutos secos, la fruta y el marisco en adultos, y a la leche, el huevo y el pescado en niños, aunque sólo hay un millar de especialistas en la sanidad pública"(larazon.es)*

El aumento de las alergias en los últimos años se ha disparado. Esto es debido a factores genéticos, factores climáticos, factores relacionados con el aumento de la higiene personal y alimentaria(algunos expertos señalan que en un ambiente de exceso de higiene, se entra en contacto con más tarde y con menos microorganismos, así que nuestro sistema inmunológico puede desarrollar respuestas inadecuadas ante sustancias a las que no tendría por qué reaccionar).
También se ha sugerido por parte de expertos que las alergias a los alimentos podrían estar aumentando debido al aumento de consumo de alimentos procesados. En esos alimentos (procesados) hay más presencia de ingredientes con potencial alergénico.

El último factor a destacar es el aumento de la contaminación en las ciudades, qué ha sido relacionado por los expertos en el aumento de problemas de funcionalidad pulmonar, aumento en el riesgo de asma y sensibilidad a alergias en la población(*Effect of Traffic-Related Air Pollution on Allergic Disease: Results of the Children's Health and Environmental Research*, pubmed).

2. Objetivos

Se definen a continuación los objetivos del presente Trabajo de Fin de Grado:

- Estudio de la plataforma Android. Posibilidades y cuota de mercado en la sociedad Española.
- Análisis del marco español de las alergias y posibilidades de mercado.
- Estructura Cliente Servidor
- Formato JSON para el intercambio de datos entre cliente y servidor.
- Diseño de una aplicación android que permita comunicarse con una base de datos mediante un servidor web.
- Definición de un análisis funcional de una aplicación.

3. Android y su Historia

La idea de Android se creó bajo el nombre de una compañía, Android Inc.
Corría el año 2003 cuando los fundadores se decidieron a poner en marcha un proyecto cuyas bases eran desarrollar un sistema operativo abierto, basado en Linux, que fuese capaz de generar una experiencia de usuario innovadora al tiempo que libre.
Entre **2003 y 2004 el desarrollo del sistema operativo** continúo con los mismos creadores, pero 2005 fue el año del vuelco. El año en que Google entró en acción y compró la compañía por unos 50 millones de dólares.

En 2006, justo después de la compra de Android por parte de Google, comenzaron los desarrollos más avanzados, y sobre todo, más orientados a cómo Google veía la evolución del sistema operativo. Durante este año, se trabajó en un desarrollo sobre un kernel linux poniendo una interfaz accesible y sencilla de manejar como principal objetivo de este desarrollo que partía de la base de los años anteriores.
En este año veíamos también como el sistema operativo en realidad crecía a una buena velocidad, ya que llegaron asociaciones con otras compañías que permitieron una evolución más efectiva, eficaz y rápida.

3.1 Android 1.0: Apple Pie

La primera versión oficial de Android la encontramos en 2007. Las premisas con las que se presentó oficialmente a Android 1.0 Apple Pie eran precisamente que se trataba de un sistema totalmente gratuito y Open Source, lo que ya marcaba las notables diferencias con iOS.

Las principales características que reunía esta versión eran:

- Notificaciones en un menú desplegable
- Se incluían widgets de escritorio
- Traía de serie la que a día de hoy es la Google Play, en aquel entonces el Android Market.
- Integración con las aplicaciones de Google; Google Mail, Contacts y Calendar.
- Otras funciones relacionadas con productos Google como eran por ejemplo el navegador, los mapas, Google Talk, el reproductor de YouTube.
- Soporte para cámaras.

El año 2008 sería el año del primer teléfono con el SO, el HTC Dream

3.2 Android 1.1: Banana Bread

Android 1.1 Banana Bread fue una pequeña actualización publicada el 9 de febrero de 2009, además, al existir en ese momento únicamente el HTC Dream, esta actualización estaba orientada exclusivamente a este terminal resolviendo pequeños errores detectados, mejorando y cambiando la API y añadiendo una serie de nuevas características.

3.3 Android 1.5: Cupcake

Con esta versión llegaron algunos cambios relevantes. En este caso, la versión Android 1.5 Cupcake se basaba en el kernel de linux 2.6.27. Entre los cambios que supuso, se puede destacar:

- Grabación y reproducción de vídeos con camcorder.
- Subida de vídeos desde el terminal a Youtube y Picassa.
- Nuevo teclado con predicción textual.
- Soporte Bluetooth A2DP y AVRCP.
- Conexión automática de Bluetooth en determinada distancia.
- Nuevos widgets y carpetas que podían formar parte de la pantalla de inicio.
- Transiciones de pantalla animadas.

3.4 Android 1.6: Donut

Esta versión fue en realidad una pequeña actualización, pero vino empaquetaba con un cuadro de búsqueda mejorado, cámara y aplicación de galería, y una renovada Android Market.

A continuación un resumen de las nuevas mejoras que trajo consigo:

- Mejor experiencia en el Android Market.
- Cámara, grabación y galería integrados entre sí.
- Selección múltiple de fotos en la galería para poder eliminarlas.
- Búsqueda por voz actualizada.
- Mejora de la experiencia de búsqueda.
- Soporte CDMA/EVDO, 802.1x, VPN y text-to-speech.
- Introducción de soporte de pantallas WVGA.
- Mejoras de rendimiento en búsqueda y cámara.
- GestureBuilder.
- Navegación gratuita turn-by-turn.

3.5 Android 2.0: Eclair

Lanzada el 26 de octubre del 2009, la actualización de Android 2.0 Eclair debuté en noviembre de ese mismo año en los Motorola Droid y se trató de un hito muy importante para la plataforma que dio paso al crecimiento exponencial y la atención de las masas.

Android Eclair sorprendió con su integración social permitiendo sincronizar los contactos de Facebook, y más tarde, Twitter, que le permitió a sus usuarios tener todos sus contactos de todas las redes sociales en un solo lugar.

Eclair también trajo el menú de contacto rápido, permitiendo que al tocar la foto de un contacto se deslizara un menú mostrando todas las formas de comunicación con el mismo.

Además de esto se introdujeron las siguientes novedades:

- Mejora de la velocidad de hardware.
- Soporte de varios tamaños de pantalla y resoluciones.
- Interfaz de usuario renovada.
- Soporte para HTML5.
- Introducción de novedades en las listas de contactos.
- Actualización de Google Maps 3.1.2.
- Soporte para Microsoft Exchange.
- Soporte de flash integrado en la cámara.
- Zoom digital.
- Posibilidad de captura multitáctil con MotionEvent mejorada.
- Mejoras en el teclado virtual.
- Bluetooth 2.1.
- Fondos de pantalla animados.

3.6 Android 2.1: Segunda etapa de Eclair

Android 2.1 representa la segunda etapa en la evolución de Eclair con su introducción en el Nexus One. No fue una gran actualización por parte de Google.

El Nexus One fue también el primer teléfono que extendiera las capacidades de voz existente encontrados en versiones anteriores de Android, dando al usuario la opción de traducir la voz en texto en cualquier campo de texto, así Android comenzaba a dar soporte a la búsqueda a través del reconocimiento de voz. Con esto se incorporo un botón del micrófono en el teclado.

Android 2.1 también introdujo algunos efectos 3D en el sistema operativo entre los que podemos encontrar el icono para lanzar las aplicaciones.

En enero de 2010, Google lanzó una actualización para el dispositivo en la que se añadía la funcionalidad *multitouch* en todos los ámbitos del Nexus One.

3.7 Android 2.2: Froyo

Lanzada el 20 de mayo de 2010, Android 2.2 Froyo fue una de las actualizaciones que consagró al sistema operativo como la competencia de iOS 4 de Apple, dotando a los terminales Android con un notable incremento de la velocidad de todo el sistema, tanto en sus aplicaciones como en la navegación de Internet.

Las principales novedades y características que Froyo trajo consigo fueron:

- Se optimiza de forma general el rendimiento del SO. Se consigue una mejora de la velocidad, de la memoria y de las aplicaciones.
- La implementación de JIT fue un paso clave en aumentar la respuesta de las aplicaciones.
- Se integra JavaScript V8 del Chrome en la aplicación navegador.
- Se añade el soporte mejorado de Microsoft Exchange.
- Se mejora el lanzador de apps y se crean accesos directos para el teléfono y el navegador.

- Aparece la función de hotspot con WiFi y la de tethering por USB.
- Se añade la función de deshabilitación del tráfico de datos del operador.
- Llegan las actualizaciones automáticas a las apps descargadas desde la tienda oficial.
- Cambio de idiomas rápido en el teclado.
- Marcación por voz y posibilidad de compartir contactos en Bluetooth.
- Se añade soporte para contraseñas con números y letras.
- Soporte para Adobe Flash 10.1.
- Se añade soporte a pantallas con alta resolución de píxeles por pulgada como las de 4 pulgadas a 720p

3.8 Android 2.3: Gingerbread

El 6 de diciembre de 2010 Google presentó de forma oficial Android 2.3 Gingerbread, una actualización que se materializaría con el lanzamiento del Nexus S.

Gingerbread incorporó una gran cantidad de novedades tanto a estético con una renovada interfaz de usuario con incrementos de velocidad y simpleza, y se preparó para la llegada de los smartphones de doble núcleo al cambiar al sistema de archivos EXT4 y de pantallas más grandes con el soporte para resoluciones WXGA y mayores.

Del lado del usuario, una de las características más notables fue el nuevo teclado virtual que simplificó la entrada de texto y permitió una edición más rápida gracias a la nueva disposición de las teclas y la función para corregir palabras ya ingresadas con sugerencias del diccionario o la opción de cambiarlas mediante voz.

Algunos otros cambios relevantes fueron los siguientes:

- Se rediseña la interfaz de usuario general de Android.
- Se añade soporte para las pantallas de mayores dimensiones y resolución WXGA y más grandes.
- Se introduce el soporte nativo en Android para llamadas VoIP SIP.
- Se añade reproducción de vídeos WebM/VP8 y decodificación de audio AAC.
- Se introducen mejoras en el campo de audio con efectos como reverberación, ecualización, virtualización y refuerzo de graves.
- Se añade el soporte para Near Field Communication.
- Se introducen las clásicas funciones de cortar, copiar y pegar en el sistema.
- Se rediseña el teclado multitáctil nativo.
- Se añade soporte mejorado para desarrollar código nativo.
- Se añade soporte nativo para sensores como giroscopios y barómetros.
- Se introduce el administrador de descargas para archivos grandes.
- Se mejora el apartado de administrador de energía, y el control de las apps con el administrador de tareas.
- Se puede utilizar de manera nativa más cámaras.

3.9 Android 3.0: HoneyComb

El 22 de febrero de 2011 Google comenzó a desdoblar el sistema operativo con la actualización de Android 3.0 Honeycomb y su correspondiente SDK, algo que tendría poca vida debido al alto costo que supone mantener dos plataformas separadas.

Basado en el kernel 2.6.36.50 de linux, Honeycomb llego por primera vez en las tablets Motorola Xoom el 24 y su principal característica fue una renovada interfaz de usuario con una nueva barra de sistema en la parte inferior de la pantalla que permitía el acceso rápido a notificaciones, estados y botones de navegación suavizados y el Action Bar que permitía el acceso a opciones contextuales, navegación, widgets y otros tipos de contenido desde la parte superior.

Otras de las principales novedades de esta actualización de android fueron:

- Escritorio 3D con rediseño de widgets
- Se mejora el sistema multitarea
- Se añaden múltiples avances en el navegador web preestablecido. Entre ellos: navegación por pestañas, relleno en automático de formularios, sincronización de los favoritos con Google Chrome y pestañas de navegación privada.
- Se añade el soporte a videochat con Talk
- Se mejoran los soportes para redes WiFi
- Se añade compatibilidad con periféricos que usan conexión USB que se ejecutan en automático desde la aplicación que les da la orden.
- Personalización de los widgets ofreciendo redimensiones sin importar el tamaño del escritorio.
- Se permite la redimensión de aplicaciones creadas para móviles, para su adaptación a las mayores pantallas de las tabletas.

3.10 Android 4.0: Ice Cream Sandwich

La llegada de Android 4.0 Ice Cream Sandwich el 19 de octubre de 2011 significó un importante paso en la evolución de Android que no solo vio renovada casi por completo su interfaz de usuario con el nuevo diseño Holo, sino que volvió a integrar el sistema operativo en sus versiones para Tablets y Smartphones.

Las novedades más importantes fueron las siguientes:

- Se unifica el uso de Android en cualquier dispositivo, ya sean tabletas, smartphones o netbooks.
- Se mejora la interfaz y se ofrece un diseño limpio. Se añade la fuente Roboto como la propia del sistema.
- Se añade la posibilidad de usar botones virtuales en lugar de los táctiles capacitivos anteriores.
- Se añade la aceleración por hardware lo que mejora considerablemente la rapidez, pero también la experiencia de usuario.
- Multitarea mejorada.
- Se añade gestor de consumo de datos de tráfico móvil dando el control de éste al usuario.
- Los widgets pasan a una lista similar a la del menú principal.

- Mejoras en el corrector de texto.
- Mejoras en las notificaciones con descartes y visualización en la pantalla de bloqueo.
- Se añaden teclas de acceso rápido para hacer capturas de pantalla.
- Se mejora la aplicación cámara añadiendo la posibilidad de las panorámicas.
- Se añade la función Android Beam para compartir contenidos.
- Reconocimiento de voz del usuario.
- Reconocimiento facial.
- Se pueden eliminar las aplicaciones personalizadas por la operadora del usuario.
- Soporte nativo MKV.
- Se añade soporte nativo para lápices táctiles.

3.11 Android 4.1, 4.2 y 4.3: Jelly Bean

Se trató de una de las grandes actualizaciones lanzadas por Google. Fue presentada el 27 de junio de 2012 y llegó al mercado el 13 de julio con el Nexus 7, el primer tablet de Google.
El objetivo primordial de Android Jelly Bean fue mejorar la estabilidad, funcionalidad y rendimiento de la interfaz de usuario, para lo cual se implementó el núcleo de linux 3.0.31 y una serie de mejoras en lo que se llamó Project Butter que permitió aumentar hasta 60 FPS las transiciones en la interfaz de usuario, dando una experiencia realmente fluida.

Otras mejoras que trajo consigo esta versión fue, una mejora de las notificaciones, ofreciendo mayor integración que en versiones anteriores. También se introdujo Google Now, que junto al Knowledge Graph y la búsqueda por voz mejorada permitió superar ampliamente a Siri, el asistente de Apple, ya que fue capaz de reconocer y predecir nuestros intereses en función del historial de búsquedas.

Otras de las características destacables son:

- Se añaden mejoras relacionadas con los ajustes de tamaño de los widgets.
- Entrada por voz mejorada, no requería conexión a Internet puesto que el intérprete se encuentra dentro del dispositivo.
- Se añade el navegador propio Google Chrome.
- Poto Sphere, permite tomar imágenes panorámicas.
- Gesture Typing, funcionalidad de escritura deslizando el dedo sobre el teclado.
- Predicción de escritura en el teclado.
- Soporte para múltiples usuarios.
- Posibilidad de incluir widgets en la pantalla de bloqueo.
- Soporte para OpenGL ES 3.0, Bluetooth Smart (o Bluetooth LE) y optimizaciones en vsync timing y el triple buffering.
- Soporte para perfiles restringidos que permite crear ambientes separados para cada usuario en el mismo dispositivo.
- Nuevo marco de DRM modular.
- Soporte para codificación VP8 integrado.
- Mejoras en el soporte RTL.
- Mejoras en seguridad gracias a SELinux

3.12 Android 4.4: Kit Kat

Lanzado oficialmente el 31 de Octubre de 2013 junto con el LG Nexus 5, Android 4.4 KitKat introdujo una reducción en el tamaño del sistema operativo junto con algunos cambios estéticos menores.

El principal avance se produjo en el apartado del rendimiento, Android 4.4 KitKat trajo la implementación de RAM, en donde se optimizaba el rendimiento para dispositivos con memoria de 512MB de RAM y de esta manera incluir los gama baja de los mercados emergentes. Cabe recordar que por aquel entonces GingerBread (Android 2.3) era la versión más utilizada de Android y esto se traducía en serias complicaciones para Google que trataba de impulsar a los fabricantes a incorporar sus nuevas funcionalidades y servicios.
En esta versión se implementó la funcionalidad para que Hangouts sirviera como el servicio de mensajería SMS oficial de Google.

En esta versión también se introdujo la implementación de manera opcional la máquina virtual ART que luego suplantaría al Dalvik en versiones posteriores.

3.13 Android 5.0: Lollipop

Lanzado el 12 de Noviembre de 2014 junto con el Nexus 6 y el Nexus 9, Android 5.0 Lollipop presentó, dentro de sus cambios, una interfaz de usuario renovada con una serie de innovaciones y nuevas funcionalidades pero lo que se destacó fue su nuevo diseño, el Material Desing que hasta el día de hoy sigue vistiendo el sistema operativo de Google con el que buscó renovar la estética con colores llamativos y un diseño atrevido.

Pero no todo fueron renovaciones estéticas en Android 5.0 Lollipop ya que dentro de sus características de funcionamiento también se destaca las nuevas formas de controlar el consumo de batería(Project Volta) y las notificaciones.

En cuanto a las notificaciones, Lollipop trae nuevas formas de controlar la forma en la que nos aparecen y cómo las recibimos gracias a los perfiles que se configuran por hora y por aplicaciones. En una determinada franja horaria, es posible establecer la prioridad de las notificaciones que se quieren recibir, incluso las llamadas entrantes.

Otra modificación importante fue la sustitución del Dalvik por parte de ART (Android Runtime), la nueva máquina virtual de Google diseñada para entregar un mejor rendimiento de las aplicaciones. Sumado a esto existen los perfiles que permite establecer el contenido del dispositivo al cual tienen acceso cada uno.

La lista de adiciones se completa con varias otras funcionalidades como soporte para procesadores de 64 bit, vectoriales dibujables, soporte para vistas previas de impresión, una nueva pantalla de desbloqueo que ahora ya no soporta widgets para una visual más limpia y sencilla, entrada y salida de audio vía USB.

3.14 EL futuro de Android: Android 6.0 Marshmallow

Lanzado el 29 de Septiembre de 2015 junto con nuevos dispositivos de Google.

En esta nueva actualización se incluyen cambios enfocados en el rendimiento y en nuevas características manteniendo la interfaz de usuario y el Material Design.

Algunas de las novedades que trae consigo esta nueva versión de Android afecta de manera directa a los permisos de las aplicaciones. Ahora las aplicaciones ya no conceden automáticamente todos los permisos al momento de la instalación cambiando a un sistema "opt-in", donde los usuarios aceptarán o no que la aplicación acceda a diversas partes de nuestro dispositivo tales como la cámara o los contactos en el mismo momento en el que el programa lo requiera.

Sumada a esta importante novedad, Marshmallow viene con una de las más novedosas y buscadas características añadidas, es el soporte nativo para el reconocimiento de las huellas digitales.
Mediante nuestras huellas dactilares, es posible autenticar la cuenta personal en el Play Store o Android Play con sólo deslizar un dedo por el dispositivo.

También trae consigo "Doze", un nuevo sistema de administración de la energía que se complementará con el anterior "Project Volta" para una mejora en la duración de la batería del dispositivo.

Android6.0 Marshmallow también será compatible con el ya famoso y muy bienvenido puerto USB tipo C, el cual es simétrico y se puede conectar de ambos lados. Pero no sólo traerá comodidad a la hora de conectar el dispositivo sino que permitirá una carga hasta 5 veces más rápida.

4. Fundamentos de Android

En la actualidad existen multitud de plataformas para móviles (iOS, Symbian, Windows Phone, BlackBerry, Palm, Java Mobile Edition, Linux Mobile (LiMo), Firefox OS, etc.). Sin embargo, algunas de las características de Android siguen haciéndolo único y distinto a los demás.
A continuación se describirán las características más importantes y las principales diferencias con el resto de sus competidores, intentando así justificar la elección de Android como plataforma para el desarrollo del producto *AllergyHelp.*

4.1 Plataforma "Abierta"

En primer lugar destacar que Android es una plataforma móvil, no un sistema operativo propiamente dicho, tal y como se cita en la web de desarrolladores oficial ***Android, the world's most popular mobile platform(https://developer.android.com/about/android.html)***

Esta plataforma esta compuesta por una base que tiene como pilar el Android Open Source Project (AOSP), a la que se sumarían los Google Mobile Services(GSM) y las Google Apps.

Android Open Source Project es como su nombre indica, el proyecto de código abierto de Android liderado por Google, con la tarea de mantener y continuar el desarrollo futuro de Android.

A pesar de que el AOSP de android es libre, Google realiza los desarrollos a puerta cerrada, y cuando está lista una nueva versión libera el código fuente para que cualquier fabricante pueda incorporarlo a sus dispositivos añadiendo sus propias capas de personalización. La personalización y libertad en Android no es del todo "real", ya que para acceder a los servicios de Google, se tendrá que asumir los términos de Google y sus requisitos (ciertas especificaciones hardware, por ejemplo), además de pasar una validación técnica de Google. Es por esto que Android siempre termina siendo una combinación de partes abiertas con partes cerradas, puesto que Google hace que utilizar Android bajo sus condiciones sea más fácil y seductor.

A pesar de todo esto, es posible trabajar con alternativas de Android completamente abiertas tal y como hizo Amazon con sus tablets kindle fire y proyectos como ***Replicant*** un fork de Android completamente libre, donde todos sus componentes lo son.

4.2 Adaptable a cualquier tipo de hardware

Uno de los mayores retos que se encuentra un programador Android cuando se dispone a desarrollar una aplicación, y una de las principales diferencias con IOS, es la inmensa variedad de dispositivos que pueden incorporar Android.

Se puede encontrar todo tipo de dispositivos con Android, relojes, cámaras, electrodomésticos y gran variedad de sistemas empotrados que se basan en este sistema operativo, además de por supuesto teléfonos móviles y tablets.

El desarrollador tendrá que pensar y tener en cuenta qué tipo de dispositivos y tamaños de pantalla va a querer que puedan utilizar su aplicación de forma optima además de la diferente capacidad de memoria que pueden ofrecer los distintos dispositivos.

4.3 Portabilidad

Cuando Google creó Android, deseaba que el entorno de desarrollo para los programadores fuera cómodo y que portar aplicaciones ya existentes fuera lo más rápido y fácil posible. Es por ello que eligieron Java, un lenguaje muy popular y del cual existían ya muchas aplicaciones escritas, además de varios entornos gráficos de desarrollo integrados.

Además el concepto de portabilidad toma fuerza en Java y Android gracias a una programa especial, la máquina virtual(VM). Un programa escrito en C o C++, suele ser dependiente de la plataforma (procesador, sistema operativo, etc.). Por el contrario, los programas Java se compilan en un formato intermedio denominado bytecode. Los bytecodes no son normalmente ejecutables directamente por ninguna plataforma. La VM, puede traducirlos a código máquina interpretable por el dispositivo sobre el que se esté ejecutando.

A pesar de basarse en código java, Google, con el argumento que Java ME no estaba preparada para la siguiente generación de teléfonos inteligentes y evitar también la compra de licencias a *Sun Mycrosystems* desarrolló la ***máquina virtual Dalvik,*** de la que se hablará más detenidamente en el presente trabajo.

4.4 Arquitectura de Interfaz

El diseño de la interfaz de usuario en Android se hace mediante xml, lo que permite que una misma aplicación se ejecute en un móvil de pantalla reducida o en un TV.

Al desarrollar para Android se anima a declarar las actividades, con sus correspondientes elementos, en ficheros XML independientes del código. De esta manera se gana en flexibilidad y claridad, ahorrando bastante trabajo el hacer cambios para probar diferentes disposiciones gráficas.
Cuando se utiliza un fichero externo para definir una interfaz se vuelve todo mucho más cómodo que escribiendo el código que construye la interfaz gráfica instanciando cada uno de los objetos y agrupándolos de una manera específica.

El programador simplemente describe cómo se agrupan los elementos de la actividad utilizando etiquetas XML de un modo muy similar al utilizado en el desarrollo web. Entonces, las herramientas de desarrollo interpretan este fichero y generan el código necesario.

4.5 Rendimiento

Es un hecho conocido a pesar del avance en los últimos años y actualizaciones que Android continua un peldaño por detrás respecto a IOS en cuanto al rendimiento nos referimos.

Tanto iOS como Android tienen una base similar basado en un núcleo de sistema operativo Unix.
En el caso de iOS se basa en el desarrollo Darwin que da núcleo también al propio OS X y en el caso de Android se basa en el núcleo Linux que usan sistemas como Ubuntu o cualquier otro.

Donde radica la principal diferencia es que iOS está diseñado para dar servicio a una decena de terminales. Partiendo de esa base, y que Apple controla el hardware que fabrica, el sistema operativo puede llevar a niveles más cercanos al núcleo porque ellos siempre controlan las configuraciones de hardware.

A diferencia de esto, existen cientos de terminales que utilizan Android. Cada terminal Android es como quiere el fabricante, lo que implica que no hay una configuración cerrada o certificación de componentes. Lo que esto implica es obvio: hacer un sistema operativo que garantice la total compatibilidad con configuraciones tan variadas no es tarea fácil. Para ello, Android ha tenido que incorporar a su sistema una capa de ejecución en software, que se coloca por encima del hardware, y que garantiza la compatibilidad. Una capa creada en Java.

Por lo tanto, la diferencia está en que iOS ejecuta el sistema operativo directamente sobre el hardware del dispositivo (ya que dicho hardware está controlado) y el código máquina que se genera, se ejecuta contra el dispositivo sin ninguna capa intermedia. Pero Android, lo que ejecuta es un código intermedio que es generado por una máquina virtual software (un programa) que se está ejecutando contra la capa del sistema.

4.6 Estabilidad y Fiabilidad

En cuanto al apartado de la estabilidad, parece que Android ha ganado terreno a sus competidores en las últimas versiones de la plataforma. A pesar de la tendencia de anteriores versiones en las que IOS era ligeramente más estable que Android, parece que con la llegada de Android Lollipop la tendencia ha dado un giro.

Según un estudio producido por la consultora Crittercism, expertos especialistas en el rendimiento de aplicaciones móviles, Android 5.0 Lollipop es algo más estable que iOS 8. Según el estudio elaborado el ratio de problemas con las aplicaciones es un 0,2% menor que en dispositivos corriendo iOS 8. El porcentaje de errores que dan las aplicaciones en el sistema de Google es de un 2%, mientras que en iOS 8 es de un 2,2%. El estudio también señala que iOS 7 era más estable que la versión actual, ya que tan sólo tenía un 1,9%.

4.7 Seguridad

4.7.1 Introducción a la seguridad de Android 5.0

Uno de los aspectos más importantes en la actualidad y que afecta directamente a las plataformas móviles es el tema de la seguridad.

Con lo que respecta al nivel de seguridad, la plataforma móvil de Google se ha convertido, gracias a su éxito global, en el blanco preferido para los desarrolladores de malware.

En el año 2014, la plataforma Android tuvo numerosos avances en seguridad. Permitió el despliegue de la encriptación total del disco, expandió el uso de criptografía de protección de Hardware y la mejora del sistema de control de acceso de la aplicación Android Sandbox, que aísla los datos y ejecución de código de una aplicación del resto de aplicaciones del dispositivo. A los desarrolladores se les proporcionaron nuevas herramientas para detectar y reaccionar a las vulnerabilidades de seguridad como los proyectos *nogotofail* y *SecurityProvider*.

También se proporcionó soporte continuo a los fabricantes de dispositivos haciendo hincapié en la vulnerabilidad de los dispositivos. Según la documentación oficial presentada por Google, se desarrollaron 79 parches de seguridad, mejorando así la capacidad de respuesta a vulnerabilidades en áreas clave, como el WebView en Android 5.0(visor de páginas web nativo) que se actualiza a través de Google Play, permitiendo así que se puedan corregir los fallos de seguridad en menor tiempo y para el mayor número de dispositivos.

Como se ha comentado, Android Sanbox ha sido uno de los focos donde se ha centrado Google en dar un avance en cuanto a seguridad. En las últimas versiones de Android se requiere SELinux en modo obligatorio para todos los dominios. SELinux es una característica que ya lleva un tiempo implementada en la mayoría de las distribuciones Linux actuales. Se trata de una serie de modificaciones al kernel (núcleo del sistema) que dividen las distintas acciones en reglas.
Por ejemplo, si una aplicación quiere acceder a la cámara web, el sistema comprobará primero cuál es la política al respecto, y comprobará que el programa tiene en efecto permiso para usar ese hardware. SELinux tiene mucho potencial, ya que permite limitar el acceso del código que se ejecuta en el sistema, sin afectar al resto y sin que su funcionamiento se impida. Por lo tanto aquellas aplicaciones creadas para tomar el control no podrán hacer nada a nuestro sistema, tampoco podrán acceder a las partes importantes del sistema operativo, como los ajustes o los archivos de configuración, que no son usados por una aplicación normal y por tanto queda fuera de las reglas implementadas.

Las principales mejoras con SELinux han sido comentadas anteriormente, a continuación se proporciona una descripción más detallada de estas mejoras:

4.7.2 Encriptación completa de Disco

El cifrado de disco completo se incorporó en Android 3.0(FDE). El FDE de Android utiliza dm-crypt del framework de Linux para implementar una encriptación de disco transparente para la partición de datos de usuario. Tan pronto como la encriptación esté habilitada, todas las escrituras que van al disco son automáticamente encriptadas antes de llevar a cabo el proceso, así como todas las lecturas desencriptadas al leerlas.

La clave de encriptación de disco (master key) es de 128 bits y se genera aleatoriamente, protegida por la contraseña de la pantalla de bloqueo. Para la encriptación se utiliza AES en modo CBC.

Todo este método de encriptación es bastante seguro, y un ataque por fuerza bruta podría tardar años si la contraseña es suficientemente larga y compleja. Sin embargo Android 3.0 utilizó el PIN o la contraseña de desbloqueo. Esto nos lleva a que un ataque por fuerza bruta podría llegar a romper el FDE 1.0 de Android.

Con la llegada de Android 4.4, FDE evolucionó. El sistema de encriptación parecía también a priori mucho más fuerte. Lo más importante vino de la mano de reemplazar PBKDF2 con scrypt, especialmente duro en GPUs pues requiere una gran cantidad de memoria. A pesar de esto, éste todavía basa su protección en una contraseña establecida por el usuario. Por lo tanto, todavía es posible realizar ataques de fuerza bruta para vulnerar los PINs más débiles.

Con Android Lollipop se da un significativo avance en materia de seguridad.
La primera característica que se introduce con la nueva versión de Android es la versión del cifrado, la cual utiliza un nuevo KDF (en lugar de PBKDF2 o scrypt), pero el modo de cifrado de disco y el tamaño de la clave no han cambiado.

La principal novedad y diferencia con las versiones anteriores es que la clave maestra a no depende exclusivamente de la seguridad de una contraseña creada por el usuario. Android se ha apoyado de QSEE (Qualcomm Secure Execution Environment), el cual ofrece un almacén de credenciales hardware compatible con la mayoría de los dispositivos que utilizan los últimos SoCs de Qualcomm. De esta forma se almacena la clave KEK en hardware.

En este sentido, si bien los ataques de fuerza bruta seguirán existiendo, ya no servirán para descifrar la información almacenada en el disco de los smartphones y tablets que ejecuten la versión “Lollipop”.

4.7.3 Multiusuario, perfil restringido, y perfil invitado

Con Android 4.2 se introdujo la posibilidad de añadir múltiples cuentas de usuario en las tablets, dejando de lado los smartphones.
Con Android 5.0 esta posibilidad llega también a los usuarios de terminales móvil, que podrán crear diferentes cuentas de usuario, con perfiles diferentes y además la posibilidad de entrar en modo invitado.

4.7.4 Autenticación mejorada

Con Android 5.0 se introdujo Smart Lock tanto en teléfonos móviles como en tablets. Esta tecnología permite más opciones en lo que se refiere al desbloqueo de dispositivos. Se puede permitir, por ejemplo, un desbloqueo automático al estar cerca de otro dispositivo de confianza, a través de NFC o Bluetooth, o cerca de una persona cuyo rostro sea considerado como de confianza.

4.7.5 Respuesta a vulnerabilidades

Como se ha comentado, en 2014 el equipo de seguridad de Android publicó numerosos parches que solucionaron 41 vulnerabilidades de impacto Moderado, 30 de impacto Alto, y 8 de impacto Bajo. Durante ese mismo año no se registraron vulnerabilidades críticas.

4.7.6 Vulnerabilidades SSL

En 2014, se produjeron diversas vulnerabilidades de perfil alto que afectaron a la implementación del protocolo de conexión segura *SSL*.

La más significativa que afectó a Android fue Heratbleed, vulnerabilidad basada en una librería OpenSSL, puede ser utilizada para extraer datos de móviles con versiones antiguas de Android. Esta permite inyectar código malicioso en el navegador Android, y utilizarlo para robar datos guardados en memoria. El fallo no es general, ni mucho menos, las herramientas de seguridad de las últimas versiones de Android protegen los datos de la memoria del terminal contra peticiones por aplicaciones no autorizadas. Los terminales susceptibles de sufrir este tipo de ataque son los que llevan la versión de Android 4.1, que según fuentes de Google alcanza el 34,4% de usuarios Android.

Para remediar esta grave vulnerabilidad, Google ha expandido la capacidad de monitorización del servicio Safety Net. Este servicio, revisa alrededor de 400 millones de conexiones cada día en busca de potenciales problemas con conexiones SSL. Esta tecnología, parte de Verify Apps, detecta y protege contra amenazas no relacionadas con las aplicaciones, como ataques a través de la red o explotación de vulnerabilidades.

4.7.7 Vulnerabilidades Android

Recientemente, expertos de la empresa de seguridad BlueBox han descubierto una vulnerabilidad Android, que se ha denominado como *FakeID*.
FakeID, permite insertar código malicioso en aplicaciones, acceder a datos de la tarjeta de crédito del usuario y tomar el control de los ajustes del dispositivo.

Google, solucionó esta vulnerabilidad rápidamente lanzando una parche para las versiones más nuevas de Android, sin embargo los miles de dispositivos que aún tienen las versiones del sistema operativo que van desde Android 2.1 a Android 4.3 y a los que los operadores o fabricantes no les han enviado el parche siguen siendo vulnerables si descargan aplicaciones desde fuera de la tienda de Google Play.
Además de proporcionar un parche para esta vulnerabilidad, Google ha explotado el potencial de monitorización vía *Google Play y Verify App*.

4.7.8 Vulnerabilidades OEM/SOC

Los dispositivos Android están generalmente implementados por el equipamiento original de fábrica, *OEM(Original Equipment Manufacture)* más la asociación con la tecnología de fabricación de componentes SoC(*System on a Chip*)

Con estos dos ingredientes se implementa el Kernel(se detallará más adelante), junto con los drivers del dispositivo hacen posible el funcionamiento de la plataforma Android.

Como se ha comentado al principio con la inclusión en modo obligatorio de SELinux, Goolgle espera reducir significativamente las oportunidades de explotar las vulnerabilidades relacionadas con el Kernel.

4.7.9 Comparativa con otras plataformas

Después de haber realizado, en apartados anteriores, un análisis detallado de las principales características de Android, a continuación se va a describir a modo de resumen, las principales diferencias y similitudes entre las distintas plataformas móviles.

	Android	iOS	Windows Phone	Blackberry
Núcleo del SO	Linux	XNU(Mac OS X)	Windows	QNX(Unix)
Licencia del software	Libre	Propietaria	Propietaria	Propietaria
Año de lanzamiento	2008	2007	2010	1999
Fabricante único	No	Sí	No	Sí
Soporte memoria externa	Sí	No	Sí	Sí
Motor del navegador	WebKit	WebKit	Trident	WebKit
Tienda	Google Play	Apple Store	Windows Phone Store	Blackberry World
Número App	1.5 millones	1.4 millones	340.000	130.000
Coste de publicación	25$	99$/años	19$/año	Gratis
Familia CPU	ARM/MIPS x86	ARM	ARM	ARM
Soporte64bits	Sí	Sí	Sí	Sí
VM	Dalvik/ART	No	.net	No
Lenguaje de Programación	Java, C++	Objective-C,C++	C#, Visual Basic, C++	C,C++,Java
Multiusuario	Sí	Sí	Sí	No

Tabla 1. Comparativa entre plataformas móviles.

Por último, compararemos la cuota de mercado entre las principales plataformas móviles. Un aspecto clave a la hora de elegir en que plataforma desarrollar nuestro producto, ya que aumentará las posibilidades de llegar a más usuarios.

Destacar que el 96,3% de los teléfonos vendidos en 2014 tenían corazón Android, o lo tenían iOS. De forma individual, la plataforma de Google es la que se queda con la gran parte del mercado, llegando a un 81,5% de cuota de mercado, sumando las ventas de todo el año. Si nos vamos a 2013, el porcentaje era del 78,7%, así que ha habido crecimiento. Será complicado que el porcentaje aumente así que los esfuerzos de Google deberán centrarse en mantener lo conseguido.

La plataforma de Apple se queda en un 14,8%, que es algo menos del 15,1% conseguido el año pasado. La situación es bastante favorable para los de Cupertino, teniendo en cuenta que luchan con una competencia que vende productos más baratos, con menos márgenes.

El espacio que le dejan a la competencia es muy pequeño, con BlackBerry casi desaparecida y con una hemorragia en la cuota, el tercer lugar es para Windows Phone, que ha registrado un crecimiento del 4,2% con respecto al año pasado.

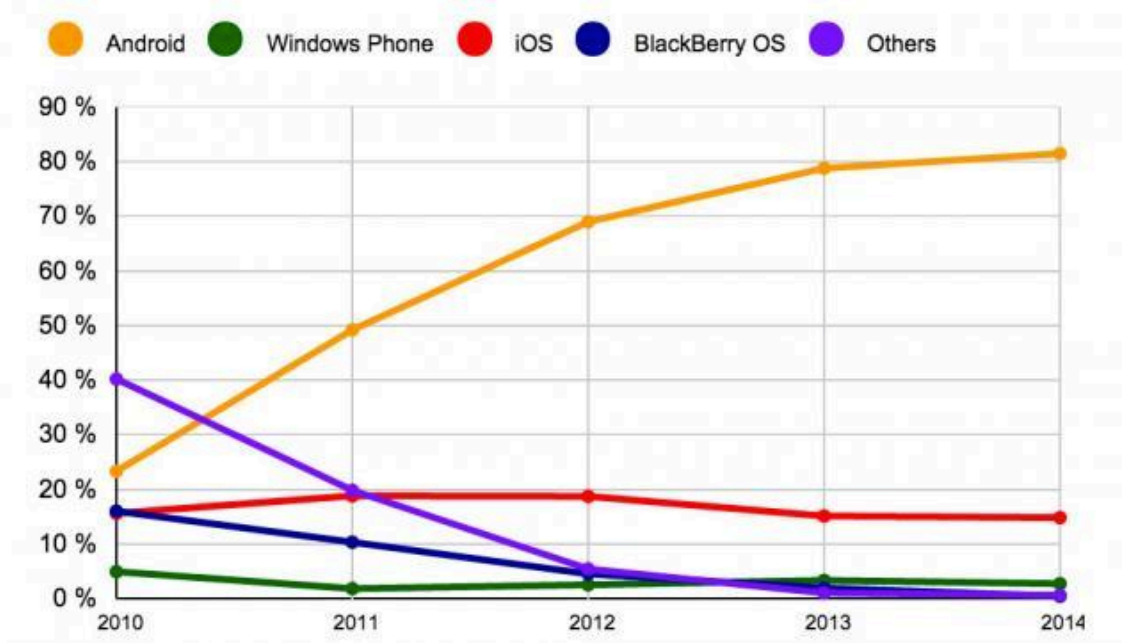

Figura 1. Cuota de mercado.

5. Estructura Android

5.1 Arquitectura Android

Android es un sistema operativo creado para ser independiente de cualquier tipo de arquitectura de hardware en los dispositivos móviles.
Es importante conocer como está estructurado Android antes de empezar a desarrollar una aplicación, a esto se le llama arquitectura y en el caso de Android está formada por varias capas que facilitan al desarrollador la creación de aplicaciones. Además, esta distribución permite acceder a las capas más bajas mediante el uso de librerías para que así el desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware de los teléfonos.

A este tipo de estructura se la conoce también como *pila*, ya que cada una de las capas utiliza elementos de la capa anterior para realizar correctamente sus funciones.

En la siguiente figura podemos observar de manera gráfica la *pila* de Android.

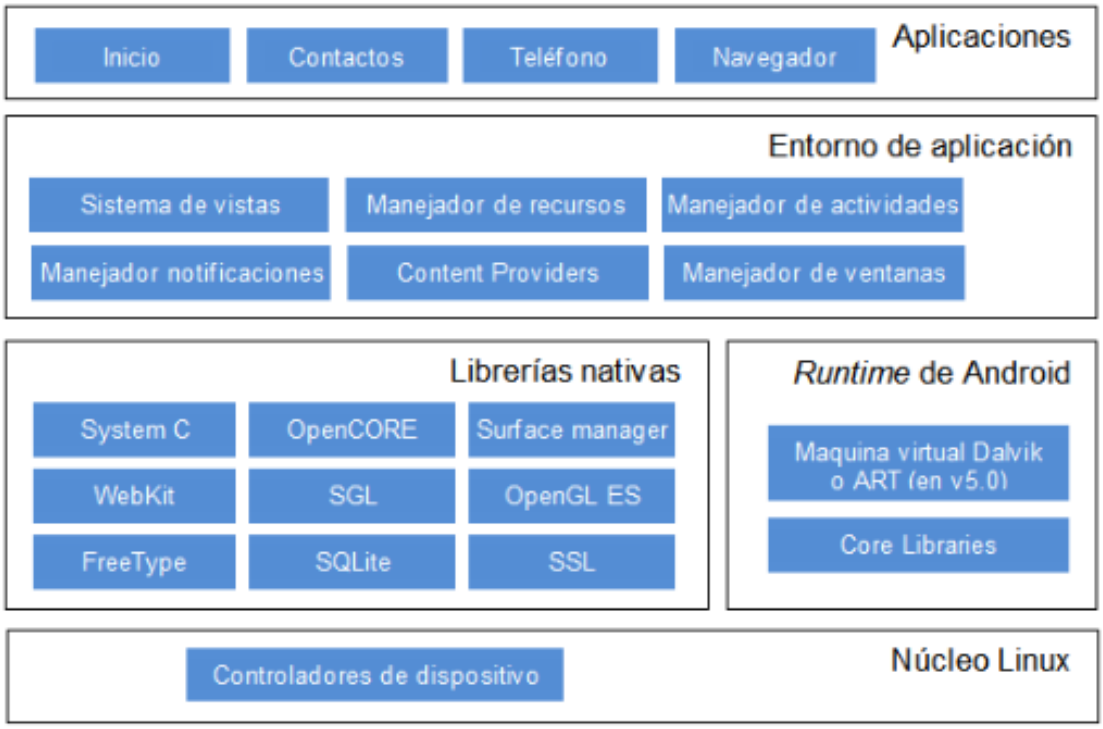

Figura 2. Arquitectura Android.

5.1.1 Núcleo Linux

El núcleo del sistema operativo Android está basado en el kernel de Linux. El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores.
Si se necesita hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cuál sea. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del kernel que permite utilizarlo desde el software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (networking), etc.

5.1.2 Librerías nativas

La siguiente capa que se sitúa justo sobre el kernel, la componen las bibliotecas nativas de Android también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma "más eficiente".

Algunas de estas librerías son:

- ***System C library***: una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- ***Media Framework:*** librería basada en PacketVideo's OpenCORE; soporta codecs de reproducción y grabación de multitud de formatos de audio vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- ***Surface Manager:*** maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- ***WebKit:*** soporta un moderno navegador Web utilizado en el navegador Android y en la vista Webview. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple.
- ***SGL***: motor de gráficos 2D.
- ***Librerías 3D***: implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador harware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- ***FreeType:*** fuentes en bitmap y renderizado vectorial.
- ***SQLite***: potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
- ***SSL***: proporciona servicios de encriptación Secure Socket Layer (capa de conexión segura).

5.1.3 Runtime de Android

Al mismo nivel que la capa de las librerías se encuentra el Runtime de Android. Esta basado en el concepto de máquina virtual utilizado en Java. Dadas las limitaciones de los dispositivos donde ha de correr Android (poca memoria y procesador limitado) no fue posible utilizar una

máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.
Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión .dex que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.
A partir de Android 5.0 se reemplaza Dalvik por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%.

También se incluye en el Runtine de Android el "core libraries" con la mayoría de las librerías disponibles en el lenguaje Java.

5.1.4 Entorno de aplicación

La siguiente capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik.
Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

Los servicios más importantes que incluye son:

- ***Views***: extenso conjunto de vistas, (parte visual de los componentes).
- ***Resource Manager***: proporciona acceso a recursos que no son en código.
- ***Activity Manager***: maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- ***Notification Manager***: permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- ***Content Providers***: mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

5.1.5 Aplicaciones

Por último, se encuentra la capa de aplicaciones Android. Aquí se incluirán todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

5.2 Componentes de las aplicaciones

Las aplicaciones Android se componen de una serie de elementos clave que resultan imprescindibles para desarrollar aplicaciones Android. Cada uno de estos componentes existe como una entidad propia y desempeña un papel específico, cada elemento es una pieza única que ayuda a definir el comportamiento general de la aplicación.

A continuación se va a realizar una descripción de los más destacados e importantes:

5.2.1 Vistas(Views)

Componen la interfaz de usuario de una aplicación., ejemplos de *views* puede ser un botón o una entrada de texto. Todas las vistas serán objetos descendientes de la clase *view*. Aunque pueden ser declaradas mediante código Java la forma habitual será utilizar el lenguaje XML, siendo el sistema el que se encargue de crear los objetos a partir de un fichero XML.

5.2.2 Layout

Un layout es un conjunto de vistas agrupadas de una determinada forma. Vamos a disponer de diferentes tipos de layouts para organizar las vistas de forma lineal, en cuadrícula o indicando la posición absoluta de cada vista. Los layouts también son objetos descendientes de la clase View. Igual que las vistas, los layouts pueden ser definidos en código, aunque la forma habitual de definirlos es utilizando código XML.

5.2.3 Actividad(Activity)

Una actividad o activity es el componente básico, que permite crear componentes que interaccionan con el usuario. Cada actividad tiene asociada una vista o view.
Una aplicación suelen necesitar varias actividades para crear el interfaz de usuario. Las diferentes actividades creadas serán independientes entre sí, aunque todas trabajarán para un objetivo común.

Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema va a mantener una pila con las actividades previamente visualizadas, de forma que el usuario va a poder regresar a la actividad anterior pulsando la tecla «retorno».

Para implementar una activity es necesario crear una clase que derive de la clase Activity. La clase Activity proporciona un conjunto de métodos que se ejecutan como consecuencia de la generación de diversos eventos.

Los ciclos de vida por los que puede pasar una actividad son los siguientes:

- ***Activa (Running):*** La actividad está encima de la pila, lo que quiere decir que es visible y tiene el foco.

- ***Visible (Paused):*** La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la

pantalla. Cuando una actividad está tapada por completo, pasa a estar parada.

- ***Parada (Stopped):*** Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.

- ***Destruida (Destroyed):*** Cuando la actividad termina al invocarse el método finish(), o es matada por el sistema.

Con el cambio de estado de una actividad, se generarán distintos métodos asociados a estos eventos y que podrán capturar el estado de la actividad.
En la figura que se muestra a continuación se observa el esquema de los métodos del ciclo de vida de una actividad.

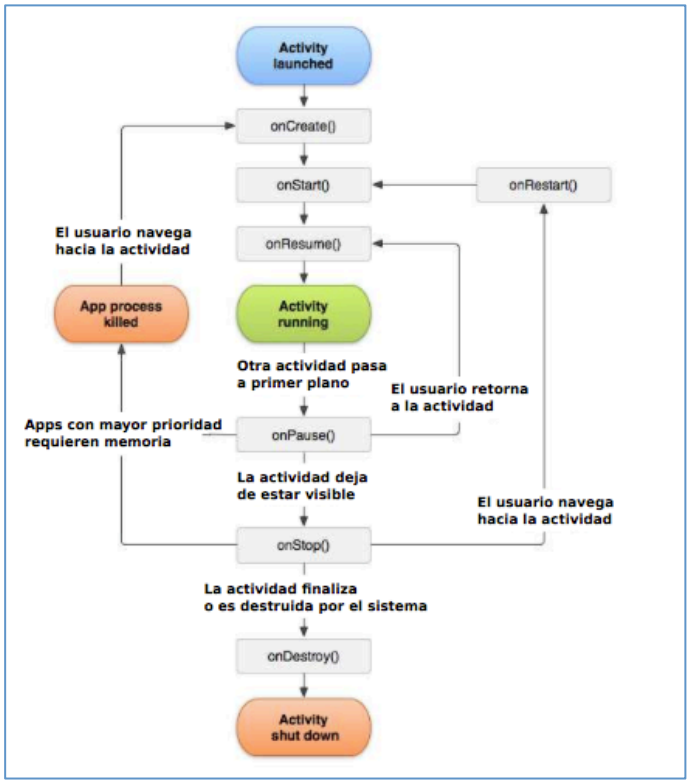

Figura 3. Ciclo de vida de una actividad.

Se pueden definir cada uno de estos métodos:

- ***onCreate():*** Se dispara cuando la Activity es llamada por primera vez. Aquí es donde se debe crear la inicialización normal de la aplicación, crear vistas, hacer los bind de los datos, etc. Este método da acceso al estado de la aplicación cuando se cerró. Después de esta llamada siempre se llama al onStart().
- ***onStart():*** Se ejecuta cuando la Activity se está mostrando en la pantalla del dispositivo del usuario.
- ***onRestart():*** Se ejecuta cuando tu Activity ha sido parada, y se quiere volver a utilizar. En el diagrama se puede ver que después de un onStop() se ejecuta el onRestart() e inmediatamente llama a un onStart().
- ***onResume():*** Se ejecuta una vez que la Activity ha terminado de cargarse en el

dispositivo y el usuario empieza a interactuar con la aplicación. Cuando el usuario ha terminado de utilizarla es cuando se llama al método onPause().

- ***onPause():*** Se ejecuta cuando el sistema arranca una nueva Activity que necesitará los recursos del sistema centrados en ella. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva activity. Después de esta llamada puede venir un onResume() si la Activity que haya ejecutado el onPause() vuelve a aparecer en primer plano o un onStop() si se hace invisible para el usuario.

- ***onStop():*** Se ejecuta cuando la Activity ya no es visible para el usuario porque otra Activity ha pasado a primer plano. Después de que se haya ejecutado este método quedan tres opciones: ejecutar el onRestart() para que la Activity vuelva a aparecer en primer plano, que el sistema elimine este proceso porque otros procesos requieran memoria o ejecutar el onDestroy() para apagar la aplicación.
- ***onDestroy():*** Esta es la llamada final de la Activity, después de ésta, es totalmente destruida. Esto pasa por los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Para volver a ejecutar la Activity se arrancaría un nuevo ciclo de vida.

5.2.3.1 Intents

Son los mensajes del sistema que se encuentran corriendo en el interior del dispositivo. Se encargan de notificar a las aplicaciones de varios eventos: cambios de estado en el hardware, notificaciones de datos entrantes y eventos en las aplicaciones.
Es posible crear Intents que lancen actividades o usarlos para detonar eventos ante algunas situaciones específicas.

5.2.3.2 Content providers

En muchas ocasiones las aplicaciones instaladas en un terminal Android necesitan compartir información. Android define un mecanismo estándar para que las aplicaciones puedan compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros.
Un ejemplo sencillo de esto es el proveedor de contenido que tiene Android para gestionar la información de contactos (agenda) del teléfono. Cualquier aplicación con los permisos adecuados puede realizar una consulta a través de un proveedor de contenido como ContactsContract.Data para leer y escribir información sobre una persona en particular.

5.2.3.3 Servicios

Las actividades, las intenciones y los proveedores de contenido explicados arriba son de corta duración y pueden ser detenidos en cualquier momento. Por el contrario, los servicios están diseñados para seguir corriendo, y si es necesario, de manera independiente de cualquier actividad. El ejemplo más simple para aterrizar este concepto es el del reproductor de música, que es un servicio que puede mantenerse corriendo mientras mandamos un SMS o realizamos alguna otra función en nuestro teléfono.

5.2.3.4 Receptor de anuncios (Broadcast receiver)

Un receptor de anuncios recibe y reacciona ante anuncios de tipo broadcast. Los anuncios broadcast pueden ser originados por el sistema o por las aplicaciones. Algunos tipos de anuncios originados por el sistema son: batería baja, llamada entrante, etc.

Aunque los broadcaster receivers no muestran una interfaz de usuario, pueden crear notificaciones en la barra de estado del teléfono para avisarle al usuario cuando un evento de este tipo se genera. Podemos pensar en estos elementos como el medio por el cual otros componentes pueden ser iniciados en respuesta a algún evento. Cabe mencionar que cada una de las emisiones de los broadcaster receivers se representa como una intención.

5.2.3.5 Fragment

Con la llegada de las tablets surgió el problema al diseñar una aplicación pensando en el tamaño de pantalla de un dispositivo móvil e intentar posteriormente, reproducirla en la tablet.
Para ayudar al diseñador a resolver este problema, en la versión 3.0 de Android aparecen los fragments. Un fragment está formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario. Una vez creados los fragments, podemos combinar uno o varios fragments dentro de una actividad, según el tamaño de pantalla disponible.

6. Entorno y recursos técnicos

En este apartado se van a describir y detallar los distintos recursos utilizados. Herramientas software, y tecnologías utilizadas para el desarrollo del proyecto.

6.1 SDK

Cuando se decidió iniciarse en el presente proyecto, el primer paso como siempre es preparar el entorno de desarrollo básico. En primer lugar se encuentra el SDK.

El Android SDK(Software Development Kit) es el conjunta de librerías y herramientas desarrolladas por Google para desarrollar, compilar y depurar aplicaciones para el sistema operativo Android. Para utilizar estas herramientas se han instalado estos dos programas:

- Java JDK
- Android SDK

6.1.1 Java JDK

El JDK(*Java Development Kit*) es necesario el para que pueda ejecutarse emulador de Android y algunas herramientas de depuración que están basados en JAVA.
JDK incluye Java Runtime Environment, el compilador Java y las API de Java.

6.1.2 1.2. Android SDK

El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen GNU/Linux, Mac OS X 10.5.8 o posterior, y Windows XP o posterior. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente, y además la utilizada para el desarrollo del proyecto es Android Studio.

Las actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

6.1.3 Android Studio

En el año 2013 Google creó Android Studio con la idea de sustituir a Eclipse a la hora de crear aplicaciones.
Está basado en el software IntelliJ IDEA de JetBrains, esta programado en Java y es publicado

de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.

Fue presentado con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android, proporcionando a Google un mayor control sobre el proceso de producción.

Android Studio se ha mantenido durante todo este tiempo en versión beta, pero desde el 8 de diciembre de 2014, en que se liberó la versión estable de Android Studio 1.0, Google ha pasado a recomendarlo como el IDE para desarrollar aplicaciones para su sistema operativo, dejando el plugin ADT para Eclipse de estar en desarrollo activo.

A continuación se describen las principales características de Android Studio:

- Utiliza ProGuard para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).
- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
- Nuevo diseño del editor con soporte para la edición de temas.
- Nueva interfaz específica para el desarrollo en Android.
- Permite la importación de proyectos realizados en el entorno Eclipse, que a diferencia de Android Studio (Gradle) utiliza ANT.
- Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar Mercurial, Git, Github o SubVersion.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Integración con Google Cloud Platform, para el acceso a los diferentes servicios que proporciona Google en la nube.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.

Dado que el entorno que venía siendo más utilizado para desarrollar aplicaciones Android era Eclipse junto con el ADT de Android, a continuación se establece una tabla comparativa entre los dos IDE, y en la que se podrá ver el porqué se ha elegido desarrollar con Android Studio.

Características	Android Studio	ADT Eclipse
Sistema de Construcción	Gradle	ANT
Construcción y gestión de proyectos basado en Maven	Si	Necesario instalar un plugin
Refactorización y completado avanzado de código Android	Si	No
Diseño del editor gráfico	Si	Si
Firma APK y gestión de almacén de claves	Si	Si
Soporte para NDK	A partir de la versión 1.3	Si
Soporte para Google Cloud Platform	Si	No
Vista en tiempo real de renderizado de layouts	Si	No

Generador de assets	Si	No
Nuevos módulos en proyecto	Si	No
Visualización de recursos desde editor de código	Si	No

Tabla 2. Android Studio vs Eclipse.

Comparando ambos entornos de desarrollo y añadiendo a esto que Android Studio ha pasado a ser el entorno recomendado para el desarrollo de aplicaciones en Android (al tratarse de un IDE oficial de Google) en el presente proyecto se ha optado por utilizar Android Studio. Además Android Studio permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos. Con la simple descarga de Android Studio se disponen de todas las herramientas necesarias para el desarrollo de aplicaciones para la plataforma Android.

6.1.4 Gradle

Gradle es una herramienta de automatización de construcción de código basada en las aportaciones que han realizado herramientas como ant y maven pero intenta llevarlo todo un paso mas allá. Para empezar se apoya en Groovy y en un DSL (Domain Specific Language) para trabajar con un lenguaje sencillo y claro a la hora de construir el build comparado con Maven. Por otro lado dispone de una gran flexibilidad que permite trabajar con ella utilizando otros lenguajes y no solo Java. Dispone por otro lado de un sistema de gestión de dependencias sólido.

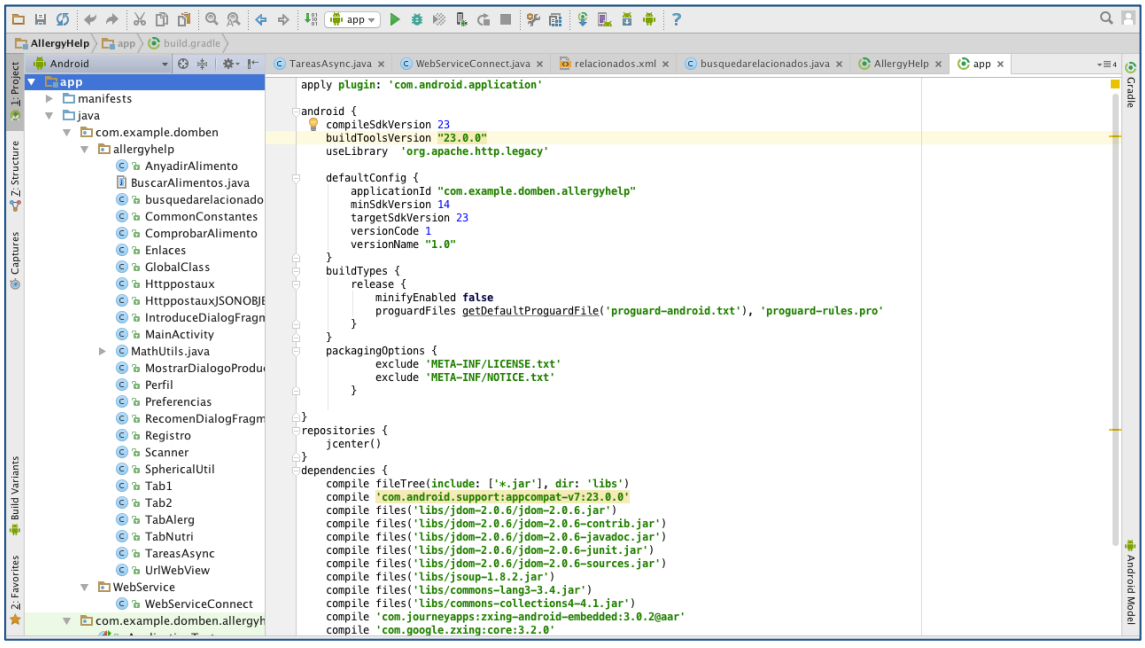

Figura 4. Ejemplo del Gradle.

6.2 MAMP

Se trata de un programa gratuito que instala en un solo paso el servidor Apache, junto con el módulo para programación en PHP y la base de datos MySQL.
Para el trabajo se ha elegido este programa puesto que es muy interesante para que en el menor tiempo sea posible ponerse a trabajar con PHP sobre Mac OS.

6.2.1 Instalación y configuración

La instalación es muy sencilla e inmediata, una vez realizada se arrancará el programa. Al ejecutarlo se abrirá una ventana donde se podrá ver el estado de los servidores PHP y MySQL, si están iniciados correctamente o detenidos. Desde esta ventana se podrá arrancar o parar los servidores.

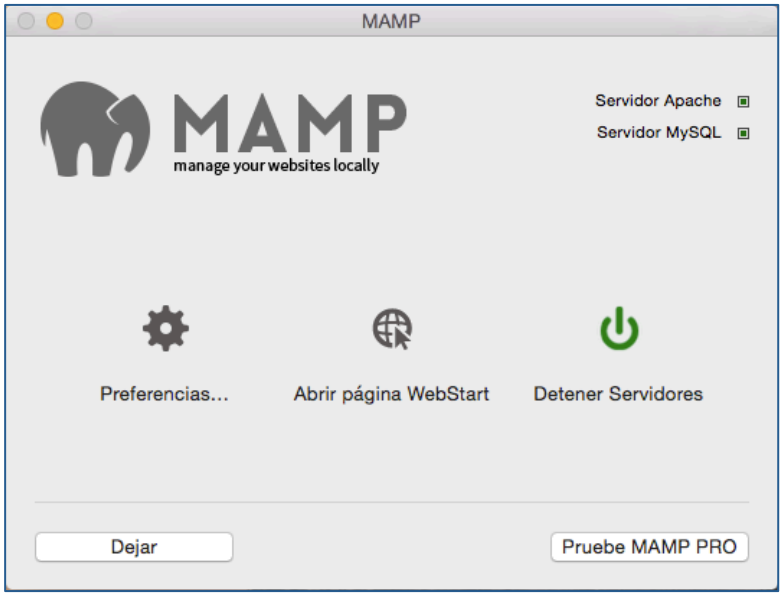

Figura 5. Pantalla de Inicio de MAMP.

La configuración del servidor es muy sencilla y aunque para nuestro caso de uso se ha dejado la establecida por defecto, Mamp permite desde la interfaz gráfica su configuración.

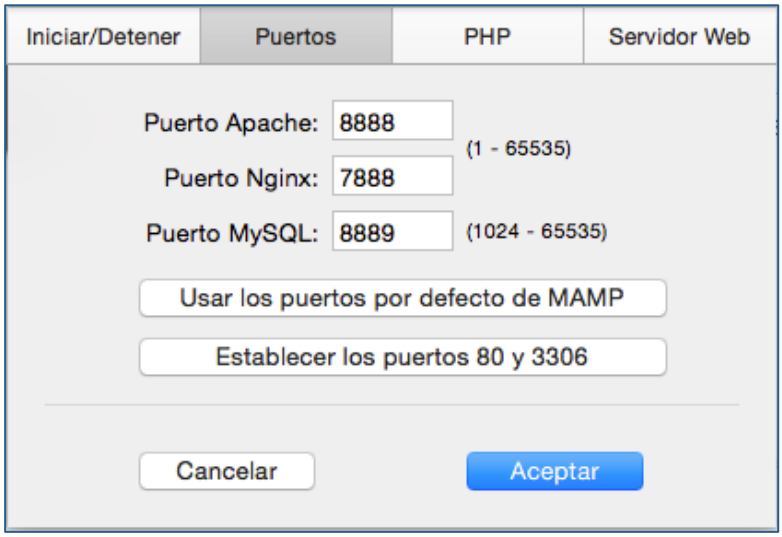

Figura 6. Pantalla de configuración.

6.2.2 Lenguaje PHP

En el presente proyecto se ha utilizado la implementación de WebServices, que serán detallados más adelante.
Para la programación de los mismos se ha utilizado el lenguaje PHP. Las principales características son:

- ***Libre*:** Gratuito, y con documentación muy amplia en internet.
- ***Sencillez y Versatilidad*:** PHP es un lenguaje de una sintaxis muy simple, y fácil de aprender. Además posee una gran variedad de funciones que pueden ser utilizadas para mejorar el rendimiento de nuestros programas.
- ***Seguridad*:** PHP es un lenguaje de uso muy común en la web, además de ser libre, esto significa que una inmensa comunidad de programadores que utilizan este lenguaje están cooperando para la mejora del motor de PHP, por lo cual es cada vez mas seguro y estable a medida que pasa el tiempo y aumenta su versión. Otra ventaja es que en internet se pueden encontrar muchos tips para evitar errores que puedan convertirse en bugs peligrosos en nuestros sitios web, y con ello puedes aprender mas fácilmente a evitar que exploiteen tus scripts php.

6.2.3 Servidor Web Apache

Uno de los componentes que viene con la instalación de Mamp es un servidor Apache. Se trata de un servidor de código abierto para la creación de páginas y servicios web. Es multiplataforma, gratuito, muy robusto y que destaca por su seguridad y rendimiento. Su misión es crítica, ya que es el encargado de aceptar las peticiones de nuestra aplicación, comunicarse con la base de datos y posteriormente devolver el resultado en un formato conocido para la aplicación.

Algunas de las ventajas de utilizar este tipo de servidor son las siguientes:

- ***Instalación/Configuración***. Software de código abierto.
- ***Coste***. El servidor web Apache es completamente gratuito.
- ***Funcional y Soporte***. Alta aceptación en la red y muy popular, esto hace que muchos programadores de todo el mundo contribuyen constantemente con mejoras, que están disponibles para cualquier persona que use el servidor web y que Apache se actualice constantemente.
- ***Multi-plataforma***. Se puede instalar en muchos sistemas operativos, es compatible con Windows, Linux y MacOS.
- **Rendimiento.** Capacidad de manejar más de un millón de visitas/día.
- ***Soporte de seguridad SSL y TLS.***

6.2.4 Base de datos MySQL

Junto con el servidor Apache, comentado en el apartado anterior, se instala el servidor de base de datos MySQL. Se trata de un sistema de gestión de base de datos relacional, multihilo y multiusuario. MySQL es como Apache *open source*.

Al contrario que en los proyectos Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es

patrocinado por una empresa privada, que posee los derechos de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado.

6.2.4.1 Características

Inicialmente MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones.
A pesar de ello, atrajo a los desarrolladores de páginas web, justamente por su simplicidad.

Poco a poco se han ido incorporando algunas de las características que no incorporaba; tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características más actuales cabe destacar:

- Lenguaje SQL
- Disponibilidad en gran cantidad de sistemas y plataformas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, etc..
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL es un sistema de administración relacional. Los tipos de base de datos relacionales archivan los datos en tablas separadas en vez de colocar todos los archivos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas.

6.2.5 *PHPMyAdmin*

Herramienta web, creada con el leguaje de programación PHP que es *open source,* para la gestión de bases de datos MySQL.

Mediante esta herramienta la gestión se puede desarrollar de manera visual por lo que es más intuitiva. Su acceso es mediante vía web.

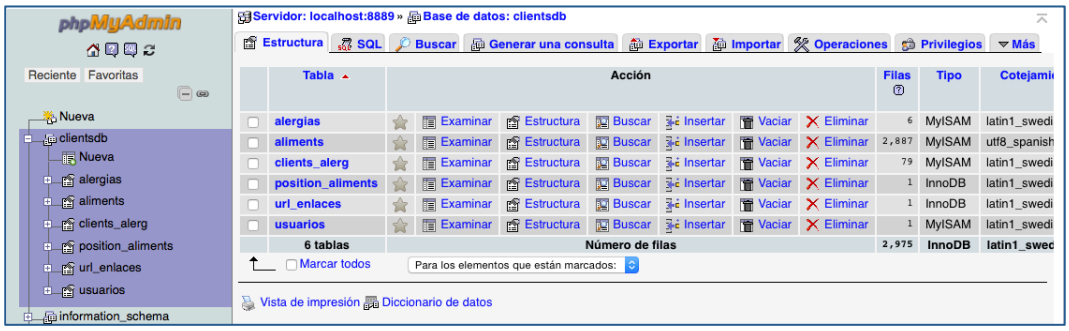

Figura 6. Imagen de PHPMyAdmin.

7. Aplicación AllergyHelp

7.1 Requisitos

7.1.1 Plataforma Android

El mínimo requisito que deberá cumplir el dispositivo en el que se desea instalar esta aplicación sea disponer de la versión 4.0 de Android(API 14).
Utilizando esta versión aseguramos que la práctica totalidad de dispositivos serán compatibles son la aplicación, y que no será necesario disponer de un Smartphone de última generación para poder beneficiarse del uso de la App.

7.1.2 Servidor Web

Como se ha comentado anteriormente, el presente proyecto dispone de una base de datos MySQL, por lo que la comunicación entre Android y la base de datos será uno de los aspectos más importantes y fundamentales para que el funcionamiento sea correcto y fluido.
A pesar de la posibilidad que ofrece un lenguaje como Java de realizar una conexión directa a una base de datos, mediante el uso de librerías. En Android y mas concretamente para nuestra aplicación, se ha estimado oportuno crear un servidor web que + sirva de puente entre la aplicación y el servidor MySQL.

Dentro de la carpeta local web, alojada dentro del programa EasyPHP, se crearán las carpetas referentes al servidor web; y en el interior de éstas, los archivos PHP, que permiten acceder a la base de dados.

7.1.3 Base de Datos

La base de datos que viene instalada con la herramienta Mamp es MySQL 5.5.42. Para el desarrollo del proyecto se accederá a ella de manera local a través de localhost.
Se ha creado una base de datos llamada "clientsdb" que contiene tanto la información de los clientes como la de los productos alimentarios.

7.2 Arquitectura de AllergyHelp

La aplicación tiene una estructura de modelo cliente-servidor. Se trata de un modelo de aplicación en el que las tareas se reparten entre los proveedores de recursos(servidor web y servidor de base de datos) y los demandantes(la propia aplicación Android).

En este apartado se va a describir el esqueleto o estructura de la aplicación, así como los recursos utilizados para el desarrollo del proyecto.

7.3 Servidor

Tal y como se ha comentado existe la posibilidad de conectarse a la base de datos MySQL remota desde la aplicación. Pero tanto por temas de eficiencia de la aplicación como por seguridad, se ha creado un sencillo WebService Apache basado en el lenguaje PHP que recibe peticiones de nuestra app y se las envía al servidor, posteriormente el servidor responde al Webservice, y este, a su vez, responde a la app.

Un WebService no tiene porque ser un archivo PHP, también se podría realizar en el servidor un programa en Java que escuche por socket a la app y pasarle parámetros para que se comunique con el servidor, o se podría crear un archivo JavaScript, o cualquier otra cosa que valga para comunicar la App con el servidor. Pero como ya se ha comentado anteriormente, PHP ofrece numerosas ventajas como versatilidad, libre y sobretodo sencillez.

Por ejemplo, si se quiere crear un nuevo usuario en la aplicación desde el dispositivo Android , habrá que introducir algunos valores de entrada, como el nombre de usuario y la contraseña. Cuando el usuario rellena esta información, se pasa la información al servicio web PHP. El servicio web se conectará a la base de datos MySQL, y buscará la tabla "Usuarios", por ejemplo, e insertará una nueva fila en la base de datos MySQL con la información que se envió desde el dispositivo Android.

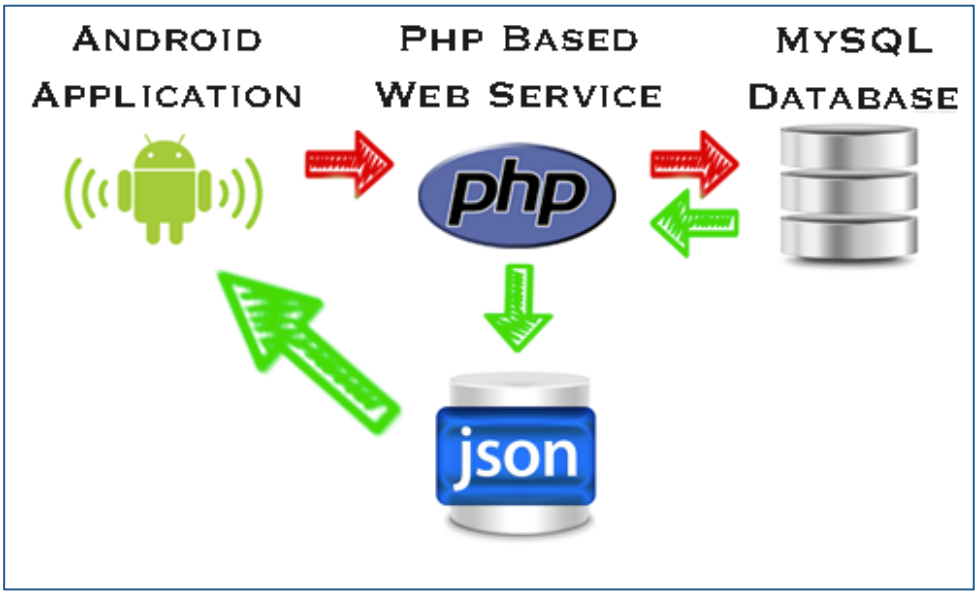

Figura 7. Comunicación entre App y Base de Datos.

7.3.1 Servidor de Base de datos

7.3.1.1 Desarrollo de la base de datos

Para el funcionamiento de la aplicación se ha diseñado un sencillo modelo de datos, en el que se estructura y organiza toda la información necesaria, como la información de los usuarios, tipos de alergias, posicionamiento y la información acerca de los productos alimenticios existentes en el mercado Español.

A continuación se describen las principales tablas usadas para la aplicación:

- ***Tabla usuarios***
  Contiene la información personal de los usuarios.
  - *id_user*: Clave primaria de la tabla usuarios
  - *nombre*: Nombre del usuario. Se utilizará en el App para hacer referencia al usuario.
  - *usuario*: Será el nombre utilizado para loguearse.
  - *pass*: Contraseña utilizada en el logueo.
  - *fechainicio:* Fecha de registro del usuario.

- ***Tabla alergias***
  En ella se encuentra la información acerca de las alergias más frecuentes en la población.
  - *alerg_key*: Clave primaria de la tabla alergias.
  - *alerg_long_desc:* Descriptivo largo de la alergia. Será un alérgeno genérico, por ejemplo frutos secos.
  - alerg_short_desc: Descriptivo corto de la alergia. Será un alérgeno específico, por ejemplo nueces.

- ***Tabla clients_alerg***
  Establece las relaciones entre los usuarios y las alergias particulares.
  - *clialerg_key*:Clave primaria de la tabla clients_alerg.
  - *alerg_key*: Clave foránea que referencia a la clave primaria de la tabla alergias.
  - *id_user:*Clave foránea que referencia a la clave primaria de la tabla usuarios.

- ***Tabla aliments***
  Contiene la información de los productos alimenticios.
  - *aliment_key*: Clave primaria de la tabla.
  - *code:* Código de barras del producto. Se utiliza para realizar las búsquedas mediante el código de barras.
  - *product_name*: Nombre del producto.
  - *generic_name:* Denominación general.
  - *categories*: Clasificación del alimento. Se utiliza en el algoritmo de búsqueda de productos relacionados.
  - *categories_tags*: Etiquetas de clasificación. Se utiliza en el algoritmo de búsqueda de productos relacionados.
  - *quatity:*Cantidad del envase
  - *packaging:* Tipo de envase, formato, material.
  - *brands:* Marca del Producto. Se utilizará a la hora de mostrar una búsqueda.
  - *origins:* Origen del producto.
  - *manufacturing_places:* Lugar de fabricación.
  - *stores:* Sitios donde comprar. Se utiliza para la búsqueda de preferencias de productos.
  - *countries:* País en el que se encuentra a la venta el producto.
  - *ingredients_text:* Composición por ingredientes del producto.
  - *allergens, allergens_tags:* Alérgenos que se encuentran en el producto. Se utiliza en la comprobación de si es o no consumible para un determinado usuario.
  - *traces:* Trazas que puede contener el producto.
  - *additives_tags:* Aditivos.
  - *energy_100g, fat_100g, satured-fat_100g, monounsatured-fat_100g, polyunsatured-fat_100g, carbohydrates_100g, sugars_100g, fiber_100g, proteins_100g:* Información sobre los valores nutricionales. Aunque se encuentra en desuso, se carga esta información en base de datos pensando en futuras funcionalidades que se comentarán más adelante.

- ***Tabla position_aliments:***

Esta tabla se utilizará para identificar, productos identificados por los usuarios con localizaciones de puntos de venta.
  - *pos_alim_key:* Clave primaria.
  - *aliment_key:* Clave foránea que referencia a la calve primaria de la tabla aliments.
  - *altitud:* Coordenada de un alimento específico, para un punto de venta.
  - *Latitud:* Coordenada de un alimento específico, para un punto de venta.
  - *Pos_alim_desc:* Nombre del punto de venta definido desde cliente por el usuario.

- ***Tabla url_enlaces:***
  Esta tabla se utiliza para guardar enlaces de artículos interesantes acerca de nutrición y alergias. Destacar que se trata de una tabla que se rellenará desde la administración de la aplicación, y que de momento no tendrá gestión desde ninguna vista de cliente.
  - *url_enlaces_key:* Clave primaria.
  - *url_enlaces_titulo:* Título del artículo.
  - *url_enlaces_desc:* Url de la web donde se encuentra el artículo.
  - *url_enlaces_tipo:* Tipo de artículo, de nutrición o de alergias.

Destacar que algunos de los campos de la base de datos se han cargado pensando en funcionalidades futuras, en caso de que la App requiera la realización de evolutivos y nuevas versiones.

7.3.2 Desarrollo del servidor Web Apache

Tal y como se ha comentado anteriormente, el desarrollo del servidor web se ha realizado mediante scripts programados en PHP. En estos ficheros se parametriza y configura la conexión a la base de datos. Además se hace todo el intercambio de datos y el tratamiento necesario para que la aplicación los interprete.
A continuación se hace una descripción de los scripts implementados.

- ***config.php*:** En este fichero se definen los parámetros de configuración para la conexión con la base de datos MySQL. La configuración elegida es la siguiente:

```php
<?php
/**
 * Database config variables
 */
define("DB_HOST", "localhost");
define("DB_USER", "root");
define("DB_PASSWORD", "root");
define("DB_DATABASE", "clientsdb");
?>
```

Figura 8. Fichero config.php

- ***connectdb.php*:** Maneja la conexión con la base de datos. Es el fichero encargado de conectarse mediante la configuración del fichero *config.php.*

```php
<?php

class DB_Connect {

    // Connecting to database
    public function connect() {
        require_once 'config.php';
        // connecting to mysql
        $con =new mysqli(DB_HOST, DB_USER,DB_PASSWORD,DB_DATABASE);
        if(mysqli_connect_error()){
            die('Error de Conecxion('. mysqli_connect_errno().')' . mysqli_connect_error());

        }
        return $con;
    }

}

?>
```

Figura 10. Fichero connectdb.php

- ***login.php*:** Es el encargado de gestionar el logueo de un usuario de la aplicación.

```php
<?php
$usuario = $_POST['usuario'];
$pass = $_POST['password'];

        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $result =$con->query("SELECT * FROM usuarios WHERE usuario= '$usuario' AND pass= '$pass'");
        if($result->num_rows > 0){
            $resultado[]=array("logstatus"=>"1");
        }
        else{
            $resultado[]=array("logstatus"=>"0");
        }
        echo json_encode($resultado);
?>
```

Figura 11. Fichero login.php

Como se puede ver, el WebService es muy simple, crea una conexión a la base de datos y realiza una consulta. Si la consulta devuelve algún registro se devuelve un 1 en formato JSON en caso contrario devuelve un 0

- ***register.php*:** Fichero encargado de la gestión en el registro de nuevos usuarios.

- ***downloadperfil.php*:** Se utiliza para realizar una descarga de los datos propios de cada usuario.

```php
<?php

//echo $_POST['usuario'];
//echo $_POST['password'];
$usuario = $_POST['usuario'];
$pass = $_POST['password'];
$resultado["alergysearch_long_desc"] = array();
$resultado["alergysearch_short_desc"] = array();

       require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $result_alergias =$con->query("select alerg_long_desc, alerg_short_desc from alergias
        inner join clients_alerg on alergias.alerg_key = clients_alerg.alerg_key
        where clients_alerg.id_user =(select id_user from usuarios where usuario = '$usuario' and pass = '$pass')");

        while($row_alerg = $result_alergias-> fetch_assoc()) {

            $long_search["alerg_long_desc"] =utf8_encode($row_alerg['alerg_long_desc']);
            array_push($resultado["alergysearch_long_desc"],$long_search);

            $short_search["alerg_short_desc"]=utf8_encode($row_alerg['alerg_short_desc']);
            array_push($resultado["alergysearch_short_desc"],$short_search);

        }
        echo json_encode($resultado);
?>
```

Figura 12. Fichero downloadperfil.php

- ***insertaralergias.php*:** Se encarga de aplicar en la base de datos la configuración de alergias definidas por el usuario.

```php
<?php

$alimentosprohibidos =$_POST['alergeno'];
$usuario = $_POST['usuario'];
$pass =$_POST['pass'];

$alimentos = implode(" ",$alimentosprohibidos);  //passamos el array a string
$sehainsertado=0;


        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $limpiaresult =$con->query("DELETE FROM clients_alerg
        where id_user=(SELECT id_user from usuarios where usuario='$usuario' and pass='$pass')");

            $alergias = explode(" ",$alimentos);
            for($i = 0; $i <= count($alergias); $i++){
                if(isset($alergias[$i])){
                    $alergkeyresult =$con->query("SELECT alerg_key FROM alergias WHERE alerg_short_desc=
                    '$alergias[$i]' or alerg_long_desc = '$alergias[$i]'"); //comprobamos que la alergia existe en bbdd
                }
                if($alergkeyresult -> num_rows > 0){
                    $alergkeyarray= $alergkeyresult -> fetch_assoc();
                    $alergkey=$alergkeyarray['alerg_key'];
                    $checkbbdd = $con->query("SELECT * FROM clients_alerg
                    where id_user=(SELECT id_user FROM usuarios WHERE usuario= '$usuario' AND pass= '$pass')
                    and alerg_key='$alergkey'");
                    if($checkbbdd->num_rows > 0){
                        $sehainsertado=2;
                    }
                    else {
                    $insertalergia =$con->query("insert into clients_alerg(alerg_key,id_user)
                    values('$alergkey',(SELECT id_user FROM usuarios WHERE usuario= '$usuario' AND pass= '$pass'))"); //
                    $sehainsertado=1;
                    }
                }

            }


        if($sehainsertado == 1){
            $resultado[]=array("checkstatus"=>"1");  //se ha insertado
        }
        else if ($sehainsertado==0){
            $resultado[]=array("checkstatus"=>"0"); //no se ha insertado
        }
        else{
            $resultado[]=array("checkstatus"=>"2"); //No se ha actualizado xq ya existe en bbdd

        }



        echo json_encode($resultado);
?>
```

Figura 13. Fichero insertaralergias.php

- ***search.php*:** Realiza una búsqueda en base de datos de productos alimentarios.

```php
<?php
    $alimentobuscado = $_POST['alimentobuscado'];
    $resultado["alimentsearch"] = array();
    $resultado["brandsearch"] = array();

    require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $result_ingredientes =$con->query("SELECT product_name, brands
        FROM aliments WHERE product_name LIKE '%$alimentobuscado%'");

        while($row_alimento = $result_ingredientes-> fetch_assoc()) {

            $alimentsearch["product_name"] =utf8_encode($row_alimento['product_name']);
            array_push($resultado["alimentsearch"],$alimentsearch);

            $brandsearch["brands"]=utf8_encode($row_alimento['brands']);
            array_push($resultado["brandsearch"],$brandsearch);

        }

        echo json_encode($resultado);
?>
```

Figura 14. Fichero search.php

- ***codigobarras.php*:** Al igual que el fichero *jsearch.php* realiza una búsqueda en base de datos del producto, pero esta vez se llamará a este PHP cuando se utilice la funcionalidad de la App del scanner de código de barras.

```php
<?php

    $alimentobuscado = $_POST['alimentobuscado'];
    $resultado["alimentsearch"] = array();
    $resultado["brandsearch"] = array();

    require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $result_ingredientes =$con->query("SELECT product_name, brands FROM aliments WHERE code ='$alimentobuscado'");

        while($row_alimento = $result_ingredientes-> fetch_assoc()) {

            $alimentsearch["product_name"] =utf8_encode($row_alimento['product_name']);
            array_push($resultado["alimentsearch"],$alimentsearch);

            $brandsearch["brands"]=utf8_encode($row_alimento['brands']);
            array_push($resultado["brandsearch"],$brandsearch);

        }

        echo json_encode($resultado);
?>
```

Figura 15. Fichero codigobarras.php

- ***busqueda.php*:** Es el encargado de comprobar que el cliente puede consumir el producto seleccionado.

```php
<?php
$alimentoprohibido =$_POST['alergeno'];
$alimentobuscado = $_POST['alimentobuscado'];

$alimentos = implode(" ",$alimentoprohibido);  //passamos el array a string
$check=1; //para comprobar si se ha encontrado la alergia en los ingredientes


        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $resultado_ingredientes =$con->query("SELECT ingredients_text FROM aliments WHERE product_name='$alimentobuscado'");
        $ingredientes_alimento = $resultado_ingredientes-> fetch_assoc();

        //buscamos tmbien en las trazas
        $resultado_trazas = $con->query("SELECT traces FROM aliments WHERE product_name='$alimentobuscado'");
        $ingredientes_trazas = $resultado_trazas-> fetch_assoc();

        //Ahora tendremos que buscar el alimento prohibido dentro de los strings obtenidos de bbdd

        $alergias = explode(" ",$alimentos);

        for($i = 0; $i <= count($alergias); $i++){
            if(isset($alergias[$i])==true){
                $pos =stripos($ingredientes_alimento["ingredients_text"],$alergias[$i]);
                if($pos!==false){
                    $check=0;
                }

                else{
                    $pos2= stripos($ingredientes_trazas["traces"],$alergias[$i]);
                    if($pos2 !== false){
                        $check=0;
                    }
                }
            }
        }
        //comprobamos si se ha encontrado
        if($check==0){
                $resultado[]=array("checkstatus"=>"0"); //Se ha encontrado, PROHIBIDO EL ALIMENTO

            }
            else{
                $resultado[]=array("checkstatus"=>"1"); //No se ha encontrado, ALIMENTP PERMITIDO
                }

        echo json_encode($resultado);
?>
```

Figura 16. Fichero búsqueda.php

- ***busquedarealacionados.php*:** Se encarga de buscar productos relacionados al producto que contiene el alérgeno.

```php
<?php
    $productos = $_POST['producto'];
    $brand = $_POST['marca'];
    $supermercado = $_POST['supermercado'];
    $alimentos =$_POST['alergeno'];
    $resultado["alimentsearch"] = array();
    $resultado["brandsearch"] = array();

    $check=1; //para comprobar si se ha encontrado la alergia en los ingredientes
    $resultadoArray = array();
        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();
        //Obtener el nombre generico, y la categoria
        $resultado_producto =$con->query("SELECT categories FROM aliments
        WHERE product_name = '$productos' AND brands = '$brand'" );
        $resultado_producto->data_seek(1);
        //Obtengo las categorias las divido y me voy quedando con los resultados.
        if($categories=$resultado_producto -> fetch_assoc()){
            $categoriesSeparada = explode(",",$categories["categories"]);
            if(count($categoriesSeparada) >1){
                $categoriesSeparada = array_slice($categoriesSeparada, 0, 2);
            }
            $categoriesSeparada = implode(",",$categoriesSeparada);
            foreach ($alimentos as $valorAlergia){
                $busquedaCategoria = $con->query("select product_name,brands,code from aliments
                where categories like '%$categoriesSeparada%' and ingredients_text not like '%$valorAlergia%'");
                while($busquedaCategoria_row = $busquedaCategoria -> fetch_assoc()){
                    if(!in_array($busquedaCategoria_row,$resultadoArray)){
                        //echo ' ' .$busquedaCategoria_row["product_name"];
                        array_push($resultadoArray, $busquedaCategoria_row);
                    }
                }
            }
        }
```

```php

//Una vez obtenidos lo productos relacionados, checkear la alergia
foreach ($resultadoArray as $valor) {
    $alimentsearch["product_name"] =utf8_encode($valor['product_name']);
    array_push($resultado["alimentsearch"],$alimentsearch);
    $brandsearch["brands"]=utf8_encode($valor['brands']);
    array_push($resultado["brandsearch"],$brandsearch);


}

echo json_encode($resultado);
```

Figura 17. Fichero busquedarelacionados.php

- ***enlaceArticulos.php*:** Proporciona a la aplicación los datos sobre enlaces de artículos de interés guardados en la base de datos.

```php
<?php
$tipoEnlace = $_POST['tipoEnlace'];
$resultado["titulosearch"] = array();
$resultado["descripcionsearch"] = array();

        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();
        if(mysqli_connect_error()){
            die('Error de Conecxion('. mysqli_connect_errno().')' . mysqli_connect_error());

        }

        //vamos a buscar los ingredientes del alimento
        //primero por nombre luego habra que comprobar tambien la marca para buscar el correcto
        $resultadoEnlaces =$con->query("SELECT url_enlaces_titulo, url_enlaces_desc FROM url_enlaces
        WHERE url_enlaces_tipo = '$tipoEnlace'");

        while($row_enlace = $resultadoEnlaces-> fetch_assoc()) {

            $tituloSearch["TITULO"] =utf8_encode($row_enlace['url_enlaces_titulo']);
            array_push($resultado["titulosearch"],$tituloSearch);

            $descripcionSearch["DESCRIPCION"]=utf8_encode($row_enlace['url_enlaces_desc']);
            array_push($resultado["descripcionsearch"],$descripcionSearch);

        }

        echo json_encode($resultado);
?>
```

Figura 18. Fichero enlaceArticulos.php

- ***anyadirAlimento.php*:** Añade un alimento en la base de datos en caso de que no exista.

```php
<?php
$descripcionProducto = $_POST['nombreProducto'];
$ingredientesProducto = $_POST['ingredientesProducto'];
$codigoBarrasProducto = $_POST['codigoBarrasProducto'];
$denominacionGeneralProducto = $_POST['denominacionGeneralProducto'];
$categoriasProducto = $_POST['categoriasProducto'];
$cantidadProducto = $_POST['cantidadProducto'];
$marcaProducto = $_POST['marcaProducto'];
$tiendasProducto = $_POST['tiendasProducto'];
$trazasProducto = $_POST['trazasProducto'];
$catagoriesGlobal=array();


        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        // Comprobar si el código de barras ya existe
        $resultCodigoBarras =$con->query("SELECT * FROM aliments WHERE code= '$codigoBarrasProducto'");

        //Comprobar si existe el nombre
        $resultDescripcion =$con->query("SELECT * FROM aliments WHERE product_name= '$descripcionProducto'");

        if($resultCodigoBarras->num_rows > 0 || $resultDescripcion->num_rows > 0){
            $resultado[]=array("logstatus"=>"0");  //Elproducto ya existe
        }else{
            $resultCategoriesSQL =$con->query("SELECT categories FROM aliments
            WHERE generic_name like '$denominacionGeneralProducto%' OR product_name like '$descripcionProducto%'");
            while($resultCategoriesSeparado = $resultCategoriesSQL-> fetch_assoc()) {
                $categoriaRowSeparada = explode(",", $resultCategoriesSeparado['categories']);

                for($i = 0; $i < count($categoriaRowSeparada); $i++){
                    if (!in_array(strtoupper($categoriaRowSeparada[$i]),array_map(strtoupper,$catagoriesGlobal))) {
                        array_push($catagoriesGlobal, $categoriaRowSeparada[$i]);
                    }
                }
            }

            // Separo el string separado por comas en un array
            $resultCategoriesPOSTSeparado = explode(",", $categoriasProducto);

            //Añado al string de categorias las recuperadas, pero comprobando que no están ya para no duplicarlas
            for($i = 0; $i < count($resultCategoriesPOSTSeparado) ; $i++){
                if (!in_array(strtoupper($resultCategoriesPOSTSeparado[$i]),array_map(strtoupper,$catagoriesGlobal))) {

                    array_push($catagoriesGlobal,$resultCategoriesPOSTSeparado[$i]);
                }
            }

            $resultCategoriesInsert = implode(",",$catagoriesGlobal);

            //Comprobar si existe el nombre
            $resultDescripcion =$con->query("INSERT INTO aliments
            ('code', 'product_name', 'generic_name', 'categories', 'quantity', 'brands', 'stores', 'ingredients_text', 'traces')
             VALUES($codigoBarrasProducto,'$descripcionProducto','$denominacionGeneralProducto','$resultCategoriesInsert',
             '$cantidadProducto','$marcaProducto','$tiendasProducto','$ingredientesProducto','$trazasProducto')");
        }

        echo json_encode($resultado);
?>
```

Figura 19. Fichero anyadirAlimento.php

- ***localizaciones.php*:** Recupera las localizaciones de un producto.

```php
<?php
    $producto = $_POST['producto'];
    $marca = $_POST['marca'];

    $resultado["altitudSearch"] = array();
    $resultado["latitudSearch"] = array();
    $resultado["tiendaSearch"] = array();


        require_once 'connectdb.php';
        // connecting to database
        $db= new DB_Connect();
        $con=$db->connect();

        $result_productos =$con->query("SELECT latitud,altitud,pos_alim_desc FROM position_aliments WHERE aliment_key =
        (SELECT aliment_key FROM aliments WHERE product_name = '$producto' AND brands = '$marca')");

        while($row_producto = $result_productos-> fetch_assoc()) {

            $latitudsearch["latitud_value"] =utf8_encode($row_producto['latitud']);
            array_push($resultado["latitudSearch"],$latitudsearch);

            $altitudsearch["altitud_value"]=utf8_encode($row_producto['altitud']);
            array_push($resultado["altitudSearch"],$altitudsearch);

            $tiendasearch["tienda_value"]=utf8_encode($row_producto['pos_alim_desc']);
            array_push($resultado["tiendaSearch"],$tiendasearch);

        }

        echo json_encode($resultado);
?>
```

Figura 20. Fichero localizaciones.php

7.3.3 Comunicación entre Cliente y Servidor

Una de las partes más importantes de la arquitectura de la aplicación es la formada por los protocolos de comunicación. De su implementación y rapidez de comunicación depende en gran parte dotar o no de agilidad a la aplicación.

Para cada una de las direcciones de comunicación se ha utilizado una implementación diferente que se va a describir a continuación:

7.3.3.1 Cliente → Servidor. Protocolo HTTP

La aplicación realiza el envío de datos mediante el protocolo de comunicación **HTTP**.
El protocolo HTTP(*Hipertext Transfer Protocol*) consiste en unas reglas sencillas de transferencia de recursos o archivos entre equipos interconectados a una red.
La comunicación se establece a través de una petición de envío, la cual contiene los datos del cliente, como el sistema operativo que usa, el navegador web desde donde se hace la petición, la ubicación del archivo solicitado (URL), etc.
Una petición puede tener múltiples objetivos dependiendo del método que se elija. Los tipos de peticiones más comunes son el *Retorno de datos* y la *Publicación de datos*. También conocidos como métodos GET y POST.

La búsqueda de una página web a través de la URL es un buen ejemplo de una petición GET.
Un ejemplo de método POST consiste en el envío de información como un formulario.

En la aplicación se hace uso del método POST para el envío de información como nombre de usuario, contraseña, alergias...etc. Y para obtener el flujo de datos asociado al recurso que se encuentra en el servidor.

7.3.3.2 Servidor→Cliente. JSON

El formato elegido por parte del servidor para el envío de datos es JSON. ¿Pero, como y qué es JSON?

A continuación se va a realizar un análisis de este formato de intercambio de datos así como un ejemplo de cómo se ha implementado en el desarrollo del proyecto.

7.3.3.2.1.1 Qué es JSON

JSON (JavaScript Object Notation) es un formato para el intercambios de datos. JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos de un modo ligero y sencillo. JSON nace como alternativa a XML y su facilidad de uso lo ha hecho popular en los últimos tiempos.
Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.
JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor, llamados objetos. Que se codifican del siguiente modo { "Producto":"Chocolate " , "Brands":"Hacendado" }
- Una lista ordenada de valores, llamados arrays. alimentsearch":[{"product_name":"Nueces mondadas Hacendado"}, {"product_name":"Mermelada de manzana y nueces Esencia Andalusi"}]

Por lo tanto JSON, es especialmente útil para enviar información entre cliente y servidor de una forma muy sencilla, puesto que cada componente descodifica la información según le convenga, indistintamente del resto del sistema.

7.3.3.2.1.2 JSON vs XML

Durante un buen tiempo, XML ha sido la única opción para compartir datos libremente. No había otros formatos abiertos disponibles y XML fue considerado como la solución para todos los problemas de intercambio de datos.
Este único formato podía manejar datos clásicos como números y texto, pero también podía manejar documentos, formatos, imágenes, audio, vídeo, y mucho más.

7.3.3.2.2 Legilibidad

Ambos formatos se caracterizan por ser fácilmente legibles, y entendibles. Sin embargo, los ficheros JSON son más restrictivos y por lo tanto ligeramente más legibles debido al menor número de formatos soportado por JSON que es mucho menor que XML. Además de esto en JSON la estructura de los datos está más estandarizada debido al hecho de que existen menos opciones cuando se compara con el formato XML.

7.3.3.2.3 Extensibilidad

XML permite la posibilidad de almacenar cualquier tipo de datos, sin embargo JSON está limitado a almacenar sólo datos clásicos como texto y números. Esto se debe a la capacidad de XML de extender los atributos de los datos almacenados en ficheros XMS.
Esto hace a XML más extensible, pero puede que no sea una ventaja, dependerá del tipo de información que trata de transferir. Los documentos requieren extensibilidad para gestionar imágenes, tablas, gráficos, y otros elementos de formato, sin embargo los datos clásicos no requieren esta extensibilidad y pueden beneficiarse de la simplicidad de JSON como es el caso de nuestra aplicación en el que sólo se enviará datos en formato texto.

7.3.3.2.4 Integración

Como ya se ha comentado en el apartado anterior XML permite adjuntar cualquier fichero de cualquier formato(fotos, audio, vídeo) mientras que JSON solamente soporta el formato de datos tradicional.

Como XML admite adjuntar ficheros de cualquier formato, podría incluirse en este un fichero ejecutable que podría tener peligrosas consecuencias para la seguridad. La simplicidad de las estructuras de datos que JSON soporta hace imposible los ataques usando este formato.

7.3.3.2.5 Compartir datos en Formato tradicional

JSON es la mejor herramienta para compartir datos. Esto es porque los datos están almacenados en vectores y registros mientras que XML almacena los datos en árboles. Ambos tienen sus ventajas, pero las transferencias de datos son mucho más fáciles cuando los datos se almacenan en una estructura que está familiarizada a los lenguajes orientados a objetos. Esto hace que sea muy sencillo importar datos desde un fichero JSON a Perl, Ruby, Javascript, Python, y otros muchos lenguajes. Para hacer lo mismo con XML, necesitaría primero transformar los datos antes de que puedan ser importados. Por este motivo, JSON es un formato de fichero superior para las APIs web.

8. Desarrollo del Cliente: Aplicación Android

8.1 Fragment Activities

Cómo se ha comentado anteriormente, las activies en Android pueden definirse de manera sencilla como cada una de las pantallas de la aplicación. En ellas se sobrescriben métodos propios como el *onCreate()* lo que permite inicializar todas las referencias a los componentes gráficos, así como manejar los escuchadores de eventos de los mismos.

Los FragementActivities son componentes que funcionan dentro del ámbito de una activity, son subclases de la misma. Su finalidad, por tanto, es la de añadir funcionalidad y ampliar la lógica de navegación entre pantallas.

Los Fragments aparecieron a partir de la versión 3.0 y se ha hecho uso de ellos en la aplicación debido a que proporcionan diseños más dinámicos y flexibles.

8.2 Layouts

Una de las partes más importantes de toda aplicación es su interfaz de usuario así como la comunicación con el mismo mediante componentes gráficos.

Los *layouts de la aplicación* son los elementos que representan el diseño de la interfaz de usuario y se encargan de actuar como contenedores de vistas, para establecer un orden visual que facilite la comunicación del usuario con la interfaz de la aplicación. Para el desarrollo de esta aplicación se han implementado haciendo uso de ficheros xml.

La forma en la que se han enlazado estos ficheros ha sido mediante el método *onCreate()* nombrado en el apartado anterior.

```
package com.example.domben.allergyhelp;

import ...


public class MainActivity extends Activity {

    private EditText USER;
    private EditText PASS;
    private Button ENVIAR;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        USER = (EditText) findViewById(R.id.txtUserName);
        PASS = (EditText) findViewById(R.id.txtPass);
        ENVIAR = (Button) findViewById(R.id.btnLogin);

        //Boton de envio de login
        ENVIAR.setOnClickListener((v) → {

                //recogemos los TextEdit
                String usuario = USER.getText().toString();
                String password = PASS.getText().toString();

                //Verificar si los datos estan en blanco
                if (CheckLoginNotEmpty(usuario, password)) {
                    new TareasAsync().new tareaLogin(MainActivity.this,usuario,password).execute();
                } else {
                    //Los campos estan vacios lanzamos funcion que muestre error
                    error_login();
                }

        });
    }
```

Figura 21. Método onCreate().

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#e2f0f8"
    android:padding="20dip">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="AllergyHelp"
        android:id="@+id/textView2"
        android:layout_marginTop="5dip"
        android:layout_marginBottom="15dip"
        android:textSize="30dip"
        android:textColor="#4d7acc"
        android:layout_gravity="center_horizontal"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/txtUserName"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_marginTop="10dip"
        android:hint="UserName"
        android:inputType="text"
        android:textColor="#4d7acc"
        android:textStyle="bold" />
```

Figura 22. Ejemplo de layout en XML.

8.3 Tareas en segundo plano

Todos los componentes de una aplicación Android, tanto las actividades, los servicios, o los broadcast receivers se ejecutan en el mismo hilo de ejecución, el llamado hilo principal o main thread, también es el hilo donde se ejecutan todas las operaciones que gestionan la interfaz de usuario de la aplicación. Es por ello, que cualquier operación larga o costosa que se realice en este hilo va a bloquear la ejecución del resto de componentes de la aplicación y por supuesto también la interfaz, produciendo al usuario un efecto evidente de lentitud, bloqueo, o mal funcionamiento en general.
Incluso puede ser peor, dado que Android monitoriza las operaciones realizadas en el hilo principal y detecta aquellas que superen los 5 segundos, en cuyo caso se muestra el famoso mensaje de "Application Not Responding" (ANR) y el usuario debe decidir entre forzar el cierre de la aplicación o esperar a que termine.
En el presente proyecto existía la dificultad encontrada a la hora de realizar una comunicación con el servidor Web, esta tarea de larga duración bloqueaba el hilo principal por lo que no era posible su implementación.

Es por esto que Android nos ofrece dos posibilidades para dar solución a este problema:

1. Crear de forma explícita un nuevo hilo para ejecutar nuestra tarea.
2. Utilizar la clase auxiliar AsyncTask proporcionada por Android.

8.3.1 Thread (Hilo) y Handler, proceso en segundo plano.

Un Thread en Java o Android es la función encargada de crear algún proceso en segundo plano. En una aplicación se pueden crear tantos hilos (Thread) como se quiera, teniendo en cuenta que el hilo deja de formar parte de la aplicación y funciona de manera independiente.

Como se ha comentado antes, siempre que haya que ejecutar un método que requiera bastante tiempo de ejecución no se podrá ejecutar en el hilo principal, dado que este hilo ha de estar siempre disponible para atender los eventos generados por el usuario por lo que nunca debe ser bloqueado. Para solucionar esto se hace uso de la clase *Thread.*

8.3.2 Clase AsyncTask

Tras ver el uso de la herramienta estándar en Java para crear hilos; en este apartado se detallará una clase creada en Android que ayuda a resolver este tipo de problemas de forma más sencilla, y que ha sido la escogida para este proyecto.
La clase AsyncTask.

Para el desarrollo de la aplicación se ha hecho uso de este recurso que ofrece Android para manejar las conexiones con la base de datos, de manera que se pueda agilizar el funcionamiento de la aplicación.

Las tareas AsyncTask se implementan creando una clase que extienda de la propia AsyncTask. Aunque es posible utilizarlo como una clase interna, en a aplicación se ha externalizado para que la Activity desde la que se invoca quede con un código más "limpio", descargándola así de líneas de código.

Como se ha comentado anteriormente el funcionamiento de la clase AsyncTask se lleva a cabo heredando de la propia clase. Junto con esto, se sobrescriben los métodos necesarios, en este caso los siguientes:

- *onPreExecute():* En este método se realizan los trabajos previos a la ejcución de la tarea. Se utiliza normalmente para configurar la tarea y para mostrar en el la interfaz de usuario que empieza la tarea.

```
protected class comprobarAlergia extends AsyncTask<Void, String, Boolean> {

    List<String> alergias;
    String alimentobuscadoalergia;
    Context context;

    protected comprobarAlergia(Context context, List<String> alergias, String alimentobuscadoalergia) {
        this.alergias = alergias;
        this.alimentobuscadoalergia = alimentobuscadoalergia;
        this.context = context;

    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        dialog = new ProgressDialog(context);
        dialog.setTitle("Comprobando Alergia");
        dialog.setMessage("Comprobando...");
        dialog.setIndeterminate(false);
        dialog.show();
    }
```

Figura 23. Método onPreExecute().

- *doInBackground(Parametros...):* Es llamado cuando termina *onPreExecute()*. Es la parte más importante donde se realiza la tarea propiamente dicha. Es el único método de los cuatro que no se ejecuta en el hilo del interfaz de usuario y es transparente para el mismo. Lo va a hacer en un hilo nuevo creado para este propósito.

```
@Override
protected Boolean doInBackground(Void... params) {
    return !webService.checkAlergia(alergias, alimentobuscadoalergia);
}
```

Figura 24. Método doInBackGround().

- ***onPostExecute(Result):*** Este método se usa para mostrar en el interfaz de usuario el resultado de la tarea. El parámetro de entrada (de la clase Result) corresponde con el objeto devuelto por el método *doInBackground()*.

```
@Override
protected void onPostExecute(Boolean result) {

    dialog.dismiss();
    Log.e("onPostExecute=", "" + result);
    if (result) {
        Toast toast = Toast.makeText(context, "MENSAJE: ALIMENTO PROHIBIDO", Toast.LENGTH_SHORT);
        toast.show();

    } else {
        Toast toast = Toast.makeText(context, "MENSAJE: ALIMENTO PERMITIDO", Toast.LENGTH_SHORT);
        toast.show();

    }
}
}
```

Figura 25. Método onPostExecute().

8.4 Mensajes Toast

Se pueden definir los mensajes *Toast* como la forma más sencilla de notificación que existe en Android.

Un Toast es un mensaje que se muestra en pantalla durante unos segundos al usuario para luego volver a desaparecer sin que el usuario haga ninguna acción. Este tipo de mensajes son ideales para mostrar mensajes rápidos y sencillos al usuario.

En la aplicación se ha hecho uso de estos mensajes para mostrar notificaciones de error principalmente y para mostrar al usuario si puede consumir o no un determinado producto alimenticio.

```
//Vibra y muestra un mensaje de error
public void error_login() {
    Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
    vibrator.vibrate(200);
    Toast toast = Toast.makeText(getApplicationContext(), "Error: Nombre de Usuario o contraseña en blanco", Toast.LENGTH_SHORT);
    toast.show();
}
```

Figura 26. Ejemplo de mensaje Toast.

8.5 DialogFragment

Uno de los recursos utilizados en la aplicación y que la dotan de una mayor interacción con el usuario son los *DialogFragments*.

Un *DialogFraagment* es un fragment que flota en el centro de la actividad en ejecución. Se han implementado para que el usuario responda a preguntas como si desea guardar una ubicación o si desea realizar una búsqueda de productos relacionados.

Para su implementación se crea una clase que hereda de la clase DialogFragment. En la clase creada, se sobrescribe el método onCreateDialog() pudiendo así personalizar el dialogo y dotándolo de la apariencia deseada.

Es posible personalizar estos diálogos mediante pero para el presente proyecto se ha implementado de una manera más sencilla.

```
public class MostrarDialogoProducto extends DialogFragment{

    public static final String BOTON_POSITIVE="boton_positive";
    public static final String BOTON_NEGATIVE="boton_negative";
    public static final String TITULO="titulo";
    public static final String ACCION="accion";


    public static MostrarDialogoProducto newInstance(final String title,final String botonPositive,
                                                     final String botonNegative, final int accion){
        MostrarDialogoProducto f = new MostrarDialogoProducto();
        Bundle args = new Bundle();
        args.putString(TITULO,title);
        args.putString(BOTON_POSITIVE,botonPositive);
        args.putString(BOTON_NEGATIVE,botonNegative);
        args.putInt(ACCION,accion);
        f.setArguments(args);
        return f;
    }
```

Figura 27. Implementación DialogFragment I.

```
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {

        final String titulo = getArguments().getString(TITULO);
        final String botonPositive = getArguments().getString(BOTON_POSITIVE);
        final String botonNegative = getArguments().getString(BOTON_NEGATIVE);
        final Integer accion = getArguments().getInt(ACCION);

        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        if(accion.equals(0)){
            builder.setMessage(titulo)
                    .setPositiveButton(botonPositive, (dialog, id) -> {
                            ((busquedarelacionados) getActivity()).doPositiveClick();
                    })
                    .setNegativeButton(botonNegative, (dialog, id) -> {
                            ((busquedarelacionados) getActivity()).doNegativeClick();
                    });
        }else{
            LayoutInflater inflater = getActivity().getLayoutInflater();
            builder.setMessage(titulo)
            .setView(inflater.inflate(R.layout.dialogo_localizaciones, null))
                    .setPositiveButton(botonPositive, (dialog, id) -> {
                            ((busquedarelacionados) getActivity()).doPositiveClick();
                    })
                    .setNegativeButton(botonNegative, new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id) {
                            ((busquedarelacionados) getActivity()).doNegativeClick();
                            dialog.dismiss();
                        }
                    });
        }
        return builder.create();
    }
```

Figura 28. Implementación DialogFragment II.

La llamada al dialogo desde la actividad en ejecución se realiza mediante el método *show()*.

Además se han implementado los métodos doPositiveClick() y doNegativeClick() a los que se realizará la llamada desde la actividad DialogFramgent cuando el usuario pulse alguno de los dos botones.

```
public void dialogoProductos() {
    ACCIONES = 0;
    MostrarDialogoProducto newFragment = MostrarDialogoProducto.newInstance("Escoge una opción",
            "Mostrar localizaciones","Añadir localización",ACCIONES);
    newFragment.show(getFragmentManager(),"dialogo_productos");
}

public void doPositiveClick() {
    if(ACCIONES.equals(0)) {
        try {
            localizacionesProducto = new TareasAsync().new recuperaLocalizacionProducto(busquedarelacionados.this,
                    producto, marca).execute().get();
            anynadirMarkers(localizacionesProducto);
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (ExecutionException e) {
            e.printStackTrace();
        }
    }
}

public void doNegativeClick(){
    if (ACCIONES.equals(0)) {
        btGuardar.setVisibility(Button.VISIBLE);
        buscarLocalizacion.setVisibility(EditText.VISIBLE);
        listview.setVisibility(View.GONE);
        mapa.moveCamera(CameraUpdateFactory.newLatLngZoom(miPosicion, 13));
        markerAnyadir.title("Localización alimento")
                .position(miPosicion)
                .draggable(true)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE));
        mapa.addMarker(markerAnyadir);
    }
}
```

Figura 29. Implementación DialogFragment III

8.6 API de Google Maps

Para el desarrollo de la aplicación se hace uso de la API de Google, mediante la cual se ofrece la posibilidad de utilizar la potente plataforma de recursos de Mapas de Google Maps.
Para la integración de esta API se ha seguido el tutorial que existe en la web oficial.

Tras seguir los pasos descritos en la web y una vez se obtiene la API KEY, se configura el *AndroidManifest.xml* con dicha clave para que la aplicación pueda utilizar dichos recursos. Además se añaden los permisos de localización necesarios de localización.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.domben.allergyhelp" >

    <application
        android:name="com.example.domben.allergyhelp.GlobalClass"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.AppCompat.Light" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version"/>
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyBuIZ1ntDIH3PyevuQ15b-qb48mjHdlguk"/>
```

Figura 30. Androidmanifest.xml I.

```
<uses-permission android:name="android.permission.INTERNET" ></uses-permission>
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
<permission
    android:name="com.example.domben.allergyhelp.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />
<uses-permission
    android:name="com.example.domben.allergyhelp.permission.MAPS_RECEIVE"/>
<uses-permission
    android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-feature
android:glEsVersion="0x00020000"
android:required="true" />
```

Figura 31. AndroidManifest.xml II.

8.7 Análisis Funcional

A continuación se va a realizar un análisis funcional de la aplicación que describa los principales flujos y escenarios de uso de la misma. Además este análisis tiene como finalidad mostrar las diferentes pantallas de usuario.

8.7.1 Login en la aplicación o registro

Álvaro es un usuario Android que acaba de bajarse la aplicación y que tiene desde pequeño alergia a los frutos secos, en concreto a las nueces. Buscando mediante su Smartphone por la Play Store algo que le ayude en su día a día, descubre una aplicación dirigida a gente con alergias.
Decide descargársela ya que es gratuita y probarla.
Una vez instalada la aplicación en el smartphone, lo primero que se encuentra son las pantallas de login y de registro.

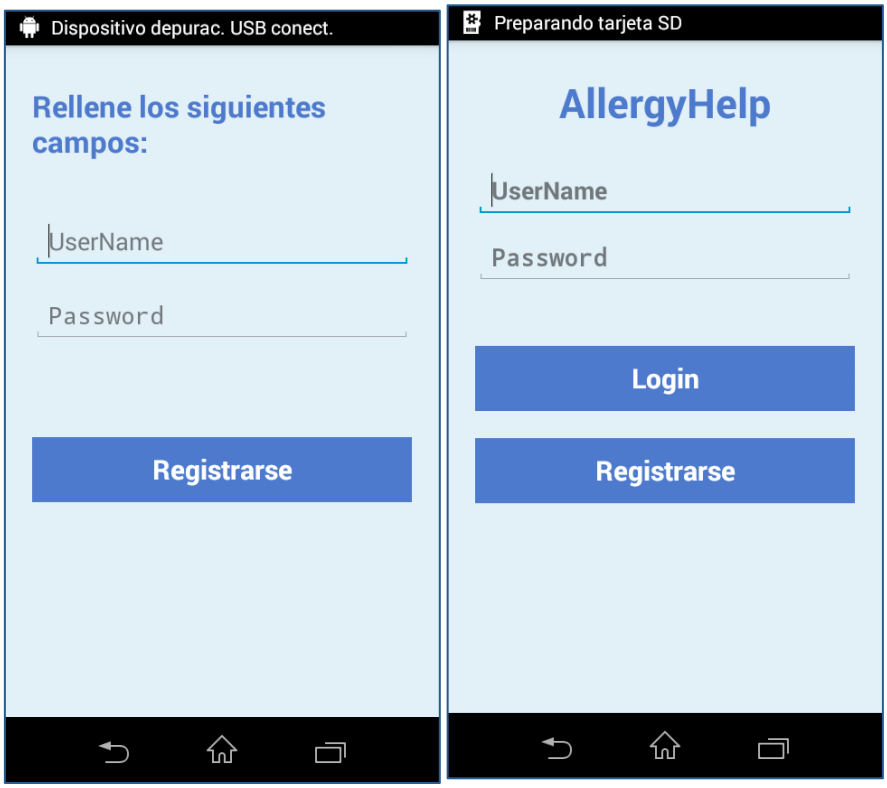

Figura 32. Pantallas de Login I.

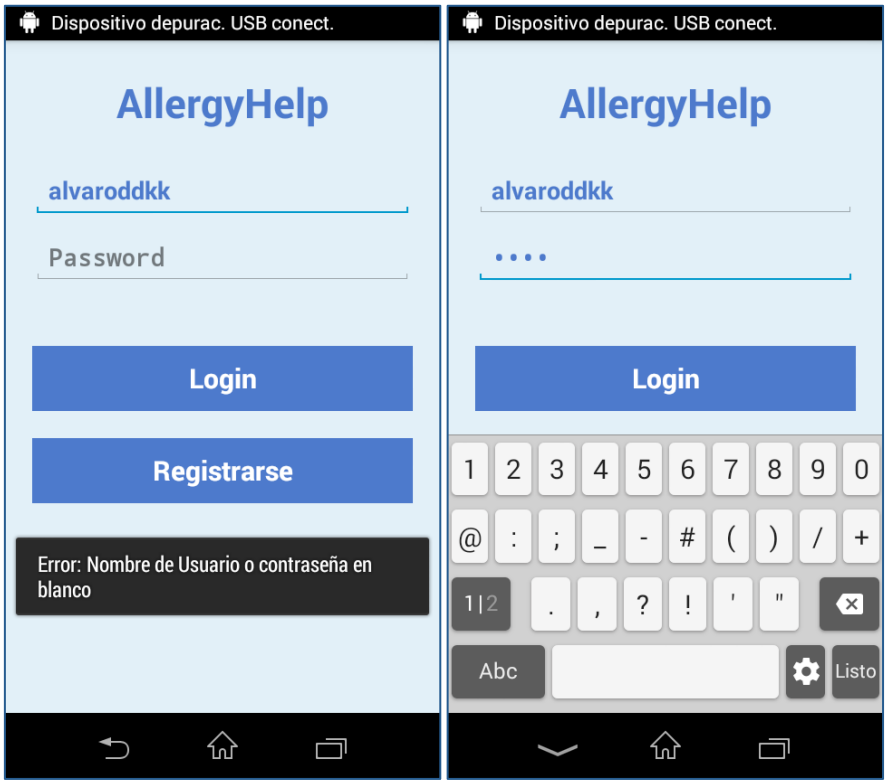

Figura 33. Pantallas de Login II

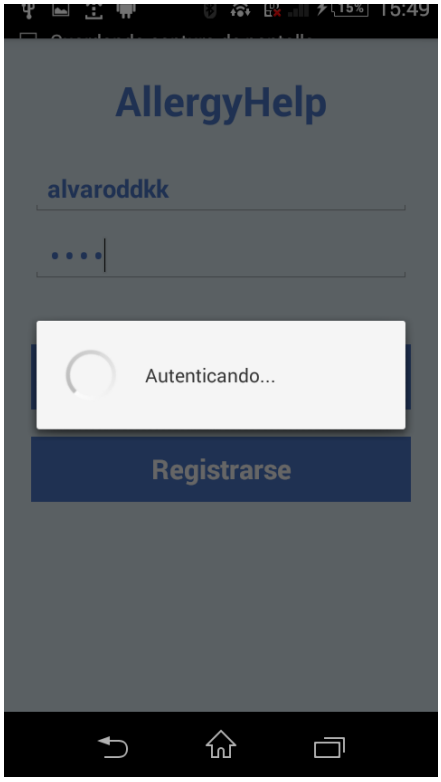

Figura 34. Pantallas de Login III.

8.7.2 Pantalla Principal

Una vez Álvaro se registra y hace el login en la aplicación, esta le lleva la pantalla padre o principal.

Esta pantalla es un fragment con dos pestañas. La primera pestaña se denomina “Mi Perfil” y desde ella se gestiona todo el acceso a las diferentes funcionalidades de la aplicación. Las segunda pestaña, “ALERGIAS” que permite la configuración de las alergias del usuario.

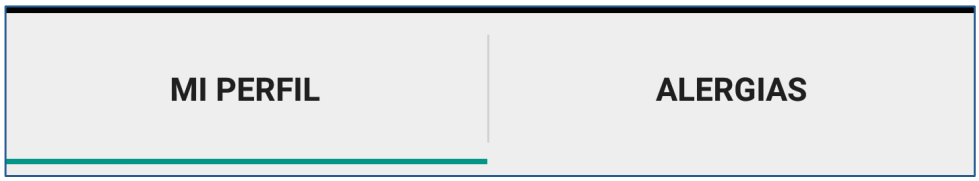

Figura 34. Pestañas Mi Perfil.

8.7.2.1 Configuración de Alergias

Álvaro decide configurar en la aplicación la alergia que tiene a las nueces. Para ello pulsa en la pestaña alergias y selecciona como alergia las nueces.

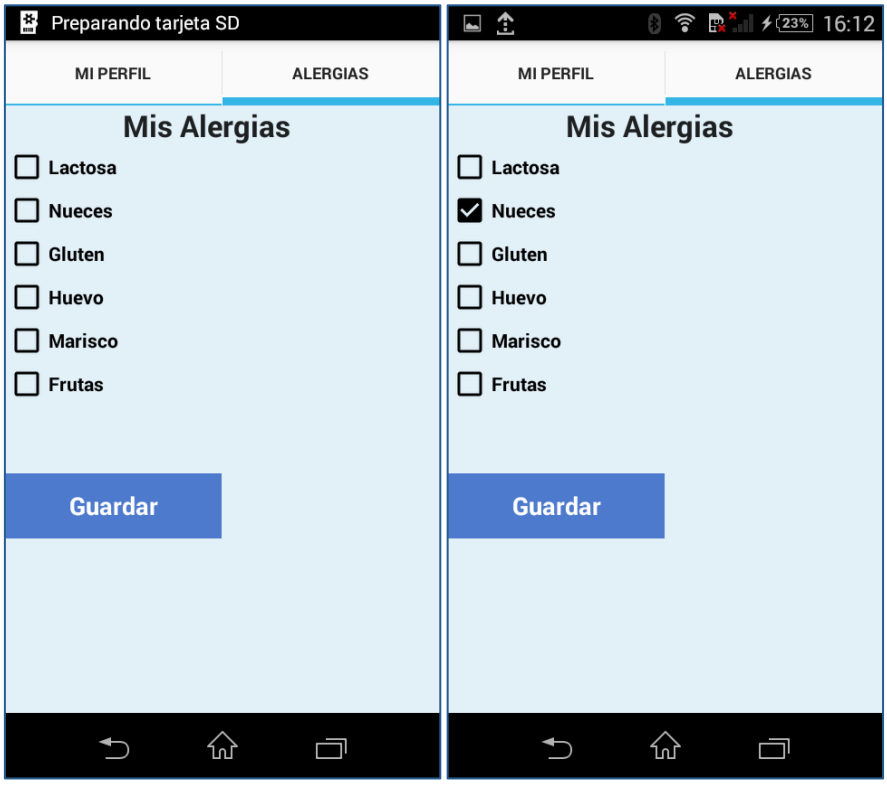

Figura 35. Configuración de Alergias I.

Tras seleccionar la alergia se pulsa el botón Guardar.

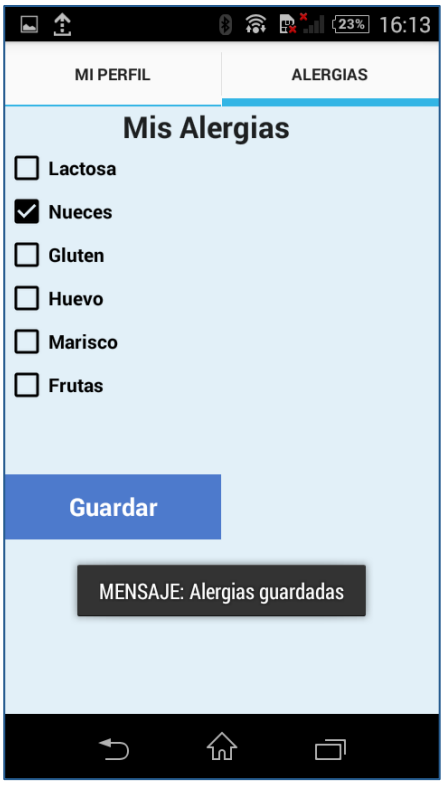

Figura 36. Configuración de Alergias II.

8.7.2.2 Mi Perfil

Una vez el usuario ha registrado en la aplicación la alergia decide volver a la vista principal de la aplicación.

En esta vista tiene la posibilidad de realizar la búsqueda de un producto, añadir productos a la base de datos o buscar artículos de alimentación y alergias que le puedan resultar interesantes.

Figura 37. Mi perfil.

8.7.3 Artículos de interés

Son las 19.00 de la tarde y Álvaro acaba de llegar del trabajo. Después de un duro día le apetece desconectar un rato y leer un rato.
Le gusta cuidar su alimentación y estar al día de las noticias relacionadas con su enfermedad por lo que decide coger su Smartphone y abrir su aplicación nueva, AllergyHelp.

Desde ella navega hasta el menú “Enlaces” y descubre que hay colgado un nuevo artículo sobre nutrición que parece bastante interesante. Pulsa sobre él y se sumerge en el artículo donde habla de la eficacia de ciertos alimentos a la hora de curar enfermedades.

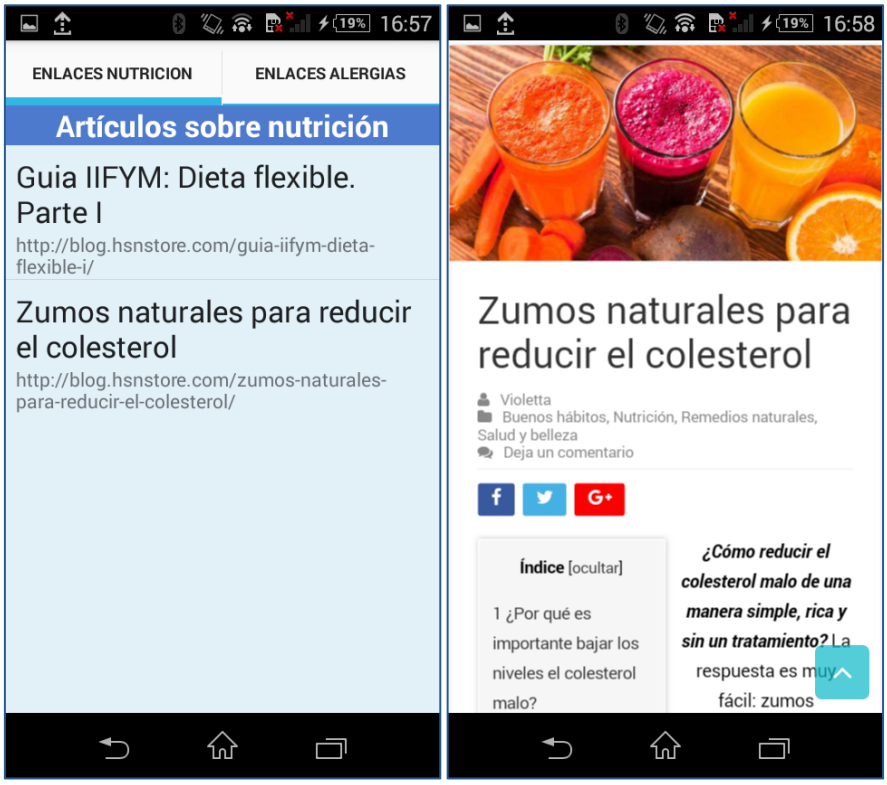

Figura 38. Artículos de Interés.

Como se puede ver, se ha elegido una interfaz lo más sencilla e intuitiva visualmente hablando. Con esto perseguimos el que cualquier persona, de cualquier edad o nivel cultural sea capaza de manejarla sin problemas.

8.7.4 Añadir Alimento

Mientras Álvaro disfruta de su lectura y de un refrescante té con limón, se le ocurre que puede ser un producto muy interesante para poder compartir con el resto de usuarios de la aplicación. Con esto decide abrir el formulario y añadir un nuevo alimento en la base de datos.

Desde esta vista se comprueba que el alimento aún no exista en la aplicación y lanza un mensaje al usuario indicándole que se ha guardado el alimento con éxito, en caso de que así sea.

Figura 39. Añadir Alimento I.

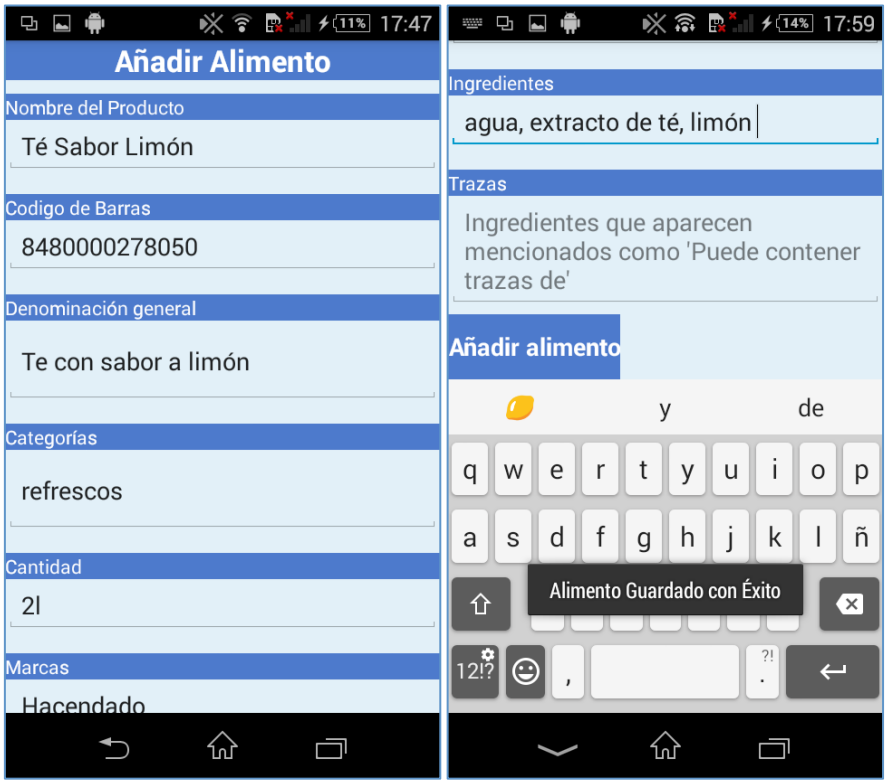

Figura 40. Añadir Alimento II.

8.7.5 Buscar Producto Alimenticio

Para facilitar tanto el tanto el desarrollo de la aplicación como para la realización de la memoria, se ha modificado por base de datos un producto y de este modo recrear con mayor facilidad una casuística real.

Se acerca la hora de cenar y Álvaro se da cuenta que tiene la nevera vacía, por lo que se dirige a su supermercado de confianza y comprar algo que le sirva de cena.

Una vez llega al supermercado, observa un producto que le apetecería comprar para cenar. Él, consciente de su alergia quiere saber de una manera rápida si puede consumir dicho producto y en caso de no ser posible obtener productos semejantes que sí pueda consumir.

8.7.5.1 Opción de búsqueda I

Este caso no tendría sentido en la vida real, pero se ha usado para facilitar tanto el desarrollo y debug como para realizar la memoria y su entendimiento

La primera opción de la que dispone el usuario para realizar un búsqueda es mediante un sencillo buscador.

Al introducir un texto y buscar se obtiene una lista de productos, relacionado con la búsqueda que se ha realizado.

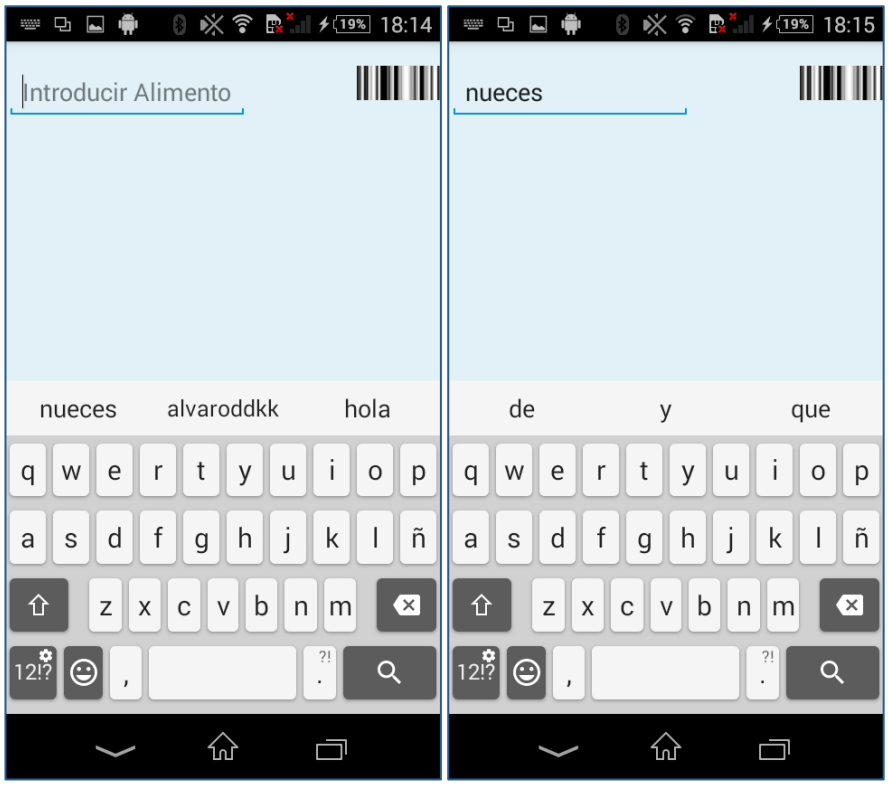

Figura 42. Buscar Alimento I.

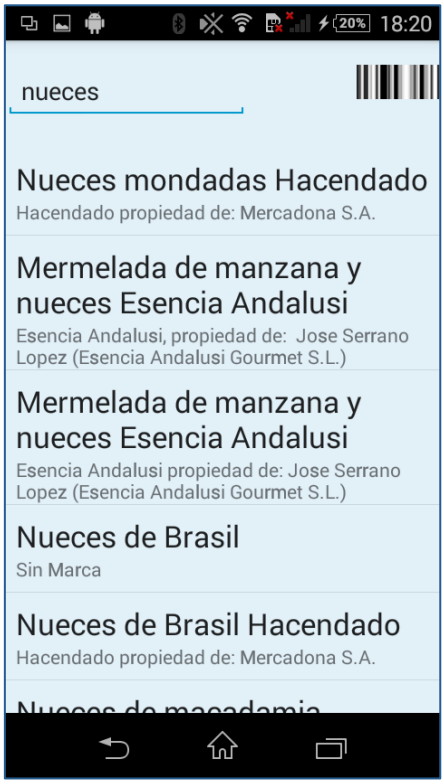

Figura 43. Buscar Alimento II.

8.7.5.2 Opción de búsqueda II

En esta ocasión Álvaro decide que no le apetece escribir y prefiere realizar un escaneo del código de barras para obtener el producto.

Figura 44. Función de Scanner.

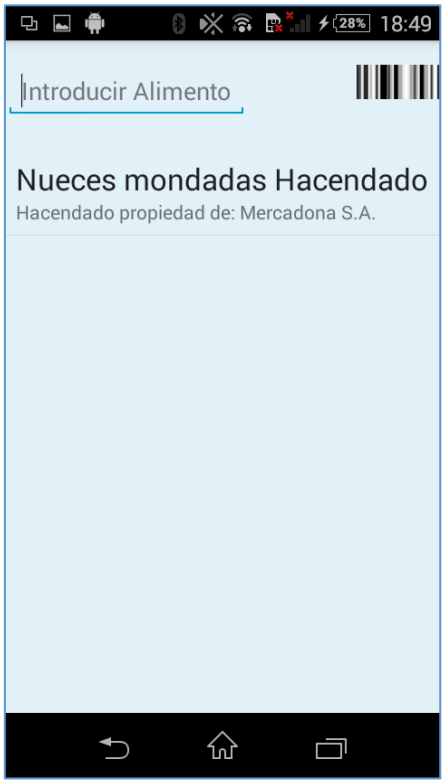

Figura 45. Resultado de la búsqueda.

8.7.5.3 Comprobar si es un alimento consumible

Una vez la aplicación ha devuelto los resultados, tan solo es necesario pulsar sobre uno de los productos y se comprobará si el producto es apto para el usuario.
De este modo, Álvaro podrá al fina saber si debe o no consumir sus tan deseadas nueces.

La aplicación calcula en relación a las alergias si debe consumir el producto. En caso que tenga alergia al producto se ofrece al usuario la posibilidad de buscar productos relacionados que sí sean consumibles.

En primer lugar se selecciona el primer ítem de la lista, que ha sido modificado por base de datos para que sea consumible por el usuario. La aplicación le indica que el alimento esta permitido.

En segundo lugar el usuario pulsa sobre el segundo ítem, al no estar permitido se le muestra un mensaje avisándole que no está permitido.

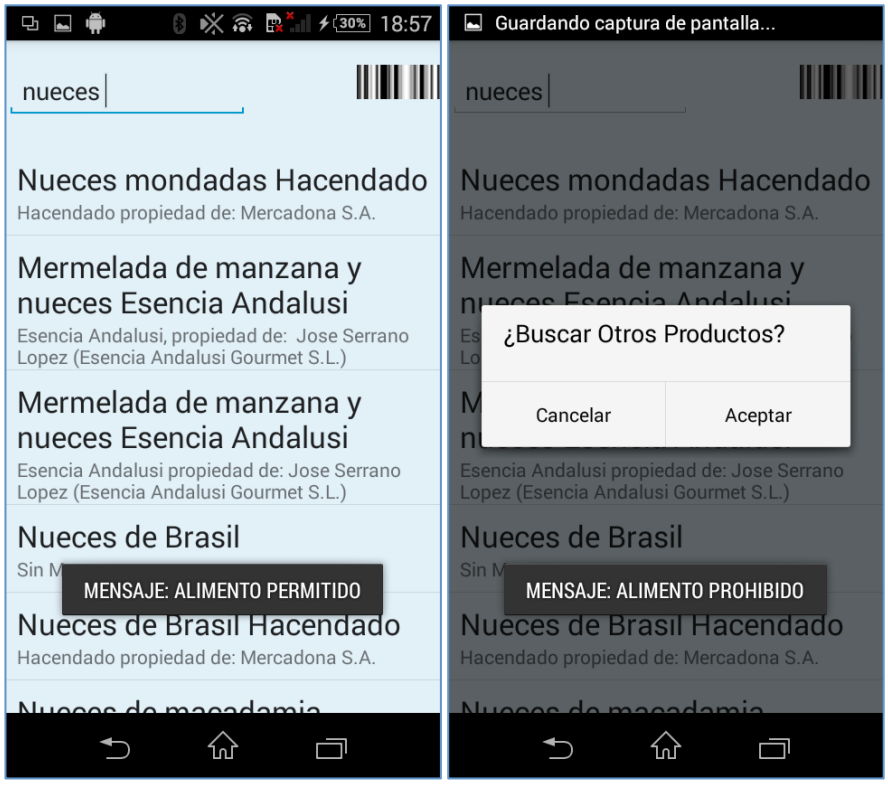

Figura 46. Comprobar Alimento.

8.7.5.4 Búsqueda de relacionados

Una de las funcionalidades que ofrece la aplicación es la búsqueda de productos alimenticios relacionados con la búsqueda.

La categorización de alimentos en base de datos permite relacionar los productos por categorías, lo que permite que esta búsqueda sea sencilla y rápida.

El usuario Álvaro escoge la opción de buscar productos relacionados. La aplicación es capaz de realizar un cálculo y devolver aquellos productos que sí puede consumir el usuario en cuestión.

Junto con la lista de relacionados, se muestra un mapa con la localización del usuario.

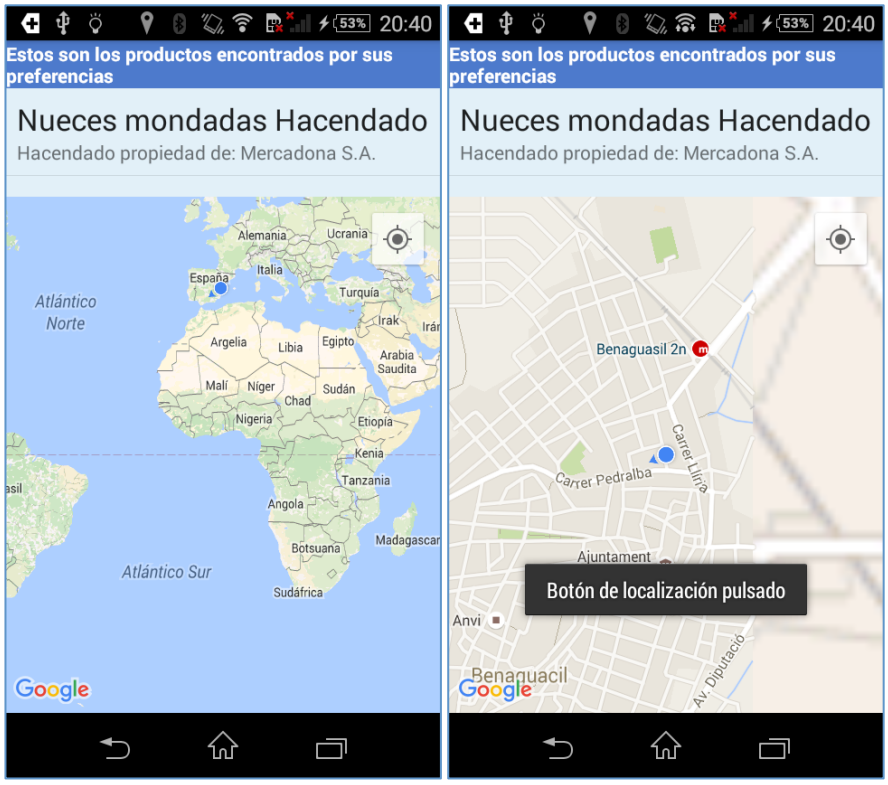

Figura 47. Vista de localizaciones I.

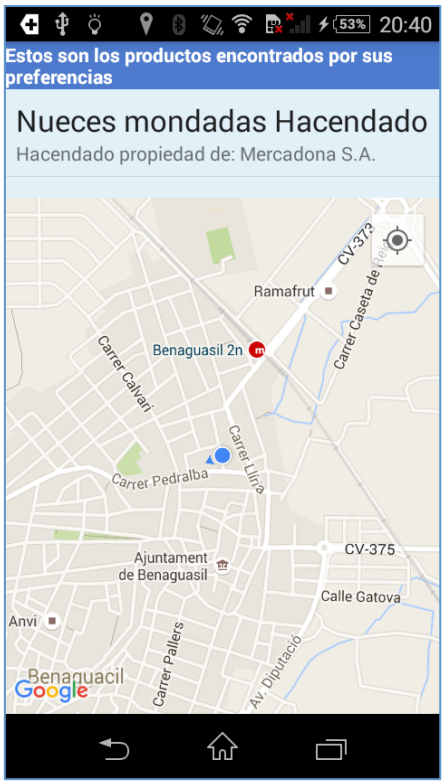

Figura 48. Vista de localizaciones II

Álvaro decide que quiere conocer más información sobre este producto, así que decide pulsar sobre él y escoge la opción de mostrar ubicaciones.
Mediante esta opción, la aplicación le ofrece al usuario la información acerca de localizaciones guardadas por usuarios donde se puede encontrar dicho producto.

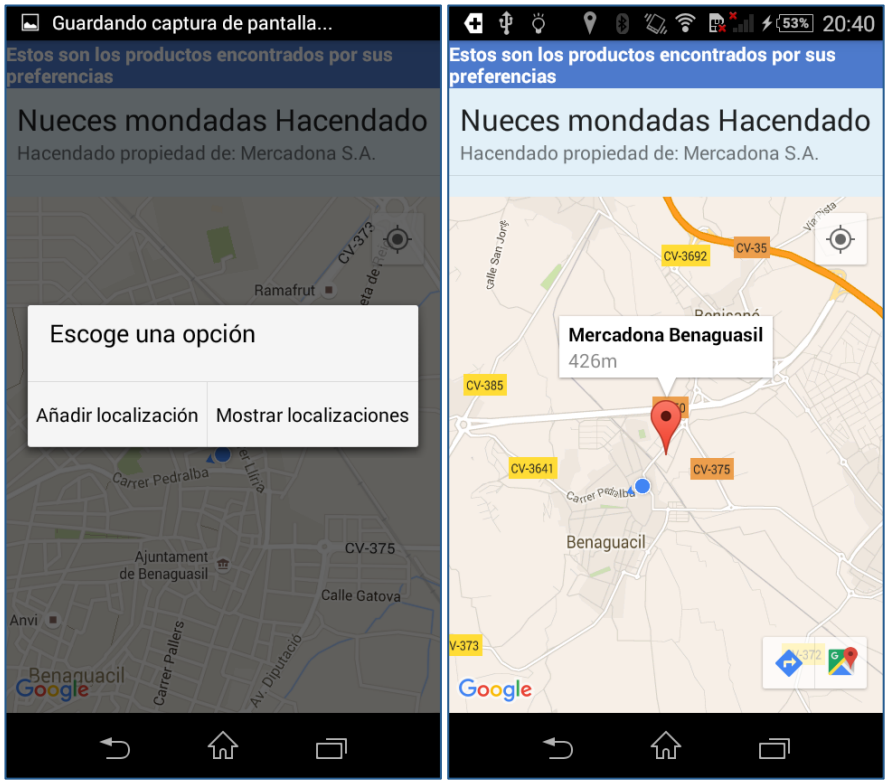

Figura 49. Vista de localizaciones III.

Esta vista ofrece al usuario información aproximada acerca de la distancia entre su ubicación y la tienda donde se encuentra el producto.

Como se ve en la imagen la tienda se encuentra cerca, por lo que Álvaro decide ir hasta la misma haciendo uso de los botones de navegar, que lo redirigen a la aplicación Google Maps y le señalan la localización del producto seleccionada.

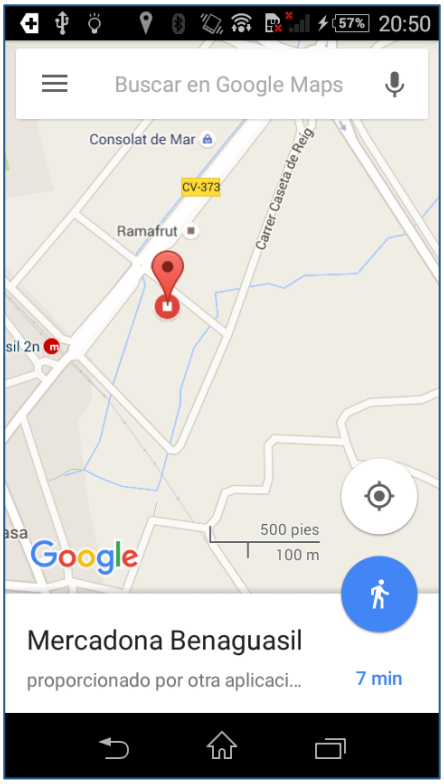

Figura 50. Navegación con Google Maps.

Continuando con el ejemplo de este usuario, de camino hacia el supermercado se encuentra una tienda pequeña.
Álvaro decide entrar, ya que tiene tiempo de sobra hasta la hora de cenar.

Una vez dentro localiza el producto que está buscando, por lo que decide añadir la localización al producto y que de esta manera otros usuario tengan también la opción de encontrarlo en este mismo sitio.

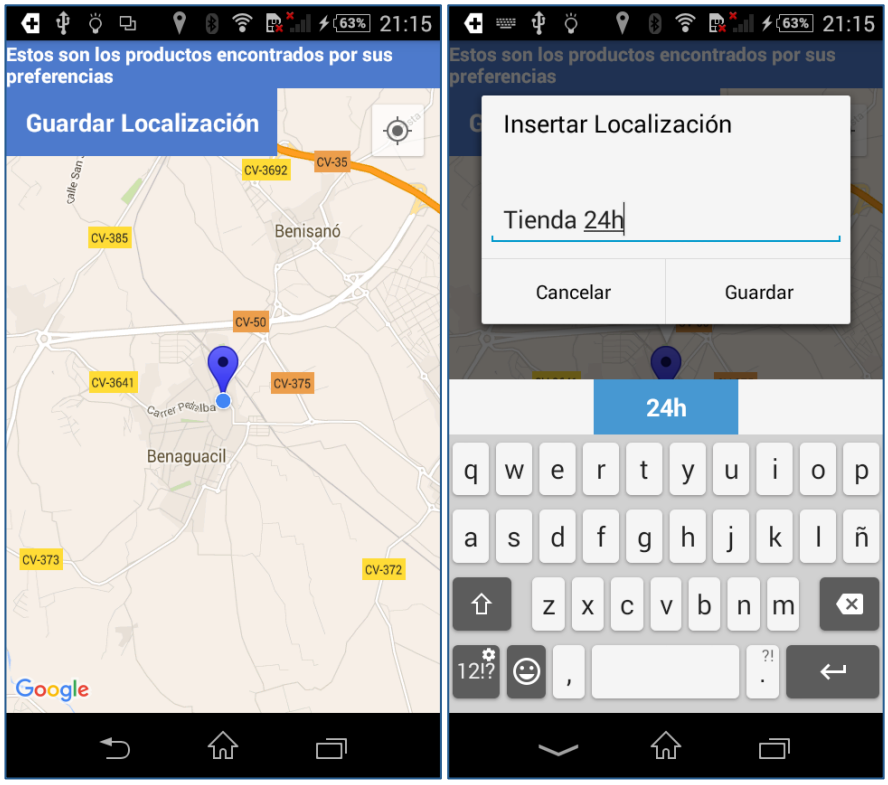

Figura 51. Guardar Localización I.

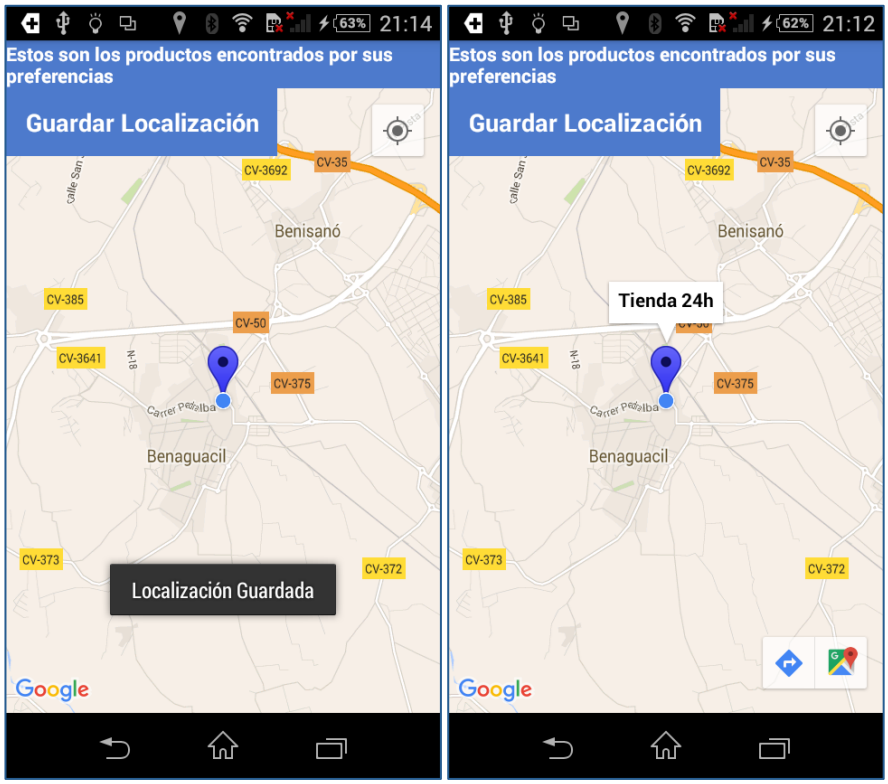

Figura 52. Guardar Localización II.

9. Mejoras e implementaciones futuras

9.1 Rendimiento

Con la experiencia ganada en el desarrollo del Trabajo de Fin de Grado y los conocimientos adquiridos, se va a realizar una serie de tareas de refactorización de código para mejorar costes y rendimiento de la aplicación, que la haga más eficiente y fluida.

9.2 Seguridad

Mejora de la seguridad de la cuenta de usuario utilizando un mecanismo de encriptación de la contraseña.
Para ello, PHP incluye el método ***string base64_encode (string $data).***

9.3 PlayStore

Publicación de la aplicación en la PlayStore de Android.

9.4 Marketing

Las redes sociales se han convertido en una parte indispensable de la estrategia de cualquier empresa para llegar al consumidor y se han convertido en un elemento clave del día a día. Por esto y por el escaso presupuesto del proyecto AllergyHelp, las redes sociales son el principal canal para hacer llegar el producto a los usuarios.
Las utilización de las siguientes herramientas nos ayudarán a realizar este marketing digital:

- ***Buffer*** – una herramienta fantástica para promocionar contenido con fechas programadas en Linkedin, G+, Twitter y Facebook.
- ***Monitor Wildfire*** – para ver el incremento de seguidores de una cuenta de Twitter, Facebook o G+, puedes inclusive comparar tu marca con otras.
- ***Rowfeeder*** – herramienta para monitorizar en real-time (con Google Docs) los tweets de un hashtag o perfil. También te da informes de resultados.
- ***Pirendo*** – analítica en Facebook y Twitter.
- ***Acumbamail*** – plataforma para envíos de campañas de email marketing.

9.5 Notificaciones

Incluir notificaciones con avisos sobre nuevos productos para alérgicos.

9.6 Origen de los datos

Integración con un sistema de información con mayor fiabilidad(Aecoc). Aecocdata es el catálogo centralizado de datos de producto **líder en España**, que permite a proveedores y clientes compartir toda la información necesaria para su actividad comercial con altos niveles de calidad y de forma rápida, segura y estándar.
Este servicio es no es gratuito y es una opción a estudiar en caso de que la rentabilidad de la aplicación lo permita.

9.7 Servidor Externo protegido

Configuración de un servidor web que sea accesible externamente.

9.8 Plan de pruebas

Utilización de herramientas(TestLinK) que permitan diseñar un plan de pruebas y el control de las mismas.

9.9 Interfaz de Usuario

Uno de los aspectos fundamentales es el aspecto de la aplicación, por lo que es imprescindible el seguir trabajando en mejorar la interfaz de usuario.

10.Dificultades encontradas

A la hora de comunicar una aplicación Android con una base de datos, es posible realizar una conexión de manera directa o como se ha hecho en este proyecto, mediante la implementación de WebServices.

Realizar una conexión directa podía suponer problemas para la integridad de la base de datos, además de aumentar el tamaño de la aplicación debido a que se debe incorporar en esta todo el código.

Por estos motivos se decidió que lo más eficiente era la utilización de un servidor web que fuera el que realizara todas las tareas de comunicarse con la base de datos y que liberara a Android de esta carga.
La implementación de los ficheros del servidor se realizó utilizando el lenguaje PHP, lenguaje que tuve que aprender. Junto con el PHP, se utilizan consultas a base de datos mediante SQL, lenguaje que también desconocía.

Siguiendo con la base de datos, la carga se ha realizando mediante la herramienta ETL *Pentaho*. Fue necesario conocer el funcionamiento y aprender a desarrollar pequeños proyectos con esta herramienta.

Centrándonos en la parte de desarrollo Android, fue necesario elegir un formato para la transmisión de datos entre el servidor y Android, decidí utilizar JSON. Fue necesario aprender detalladamente este tipo de formato y a cómo lograr su implementación tanto en PHP como en Android.

Por último y sin duda alguna, la mayor dificultad con la que me he encontrado ha sido el obtener el origen de los datos. Internet ofrece miles de datos e información sobre productos alimenticios, pero ha sido prácticamente imposible encontrar una fuente de datos sobre composición de ingredientes.

11.Conclusiones

Tras finalizar el proyecto y la memoria del mismo, llega la hora de echar la vista atrás y reflexionar sobre la evolución personal y del proyecto.

A pesar de que *AllergyHelp* nació con la idea de ser únicamente un proyecto de fin de grado, la motivación de poder ayudar a la gente con alergias alimentarias y ver hasta donde se ha llegado hace que esto no sea un punto y final para la aplicación, sino que se estudiará la posibilidad de otorgarle algunas mejoras que permitan su publicación en la Play Store de Android.

En lo personal, elegí un trabajo de fin de grado basado en Android con el objetivo de mejorar mis conocimientos de programación y ver si era capaz de llevar adelante un proyecto de estas características. Finalmente, no sólo he aprendido a realizar códigos más complejos y eficientes sino que he aprendido acerca de Servidores Web, PHP, Bases de Datos, lenguaje SQL y ETLs.

Con los conocimientos obtenidos, pero sobre todo con la superación personal conseguida, puedo decir que soy una persona mucho más preparada para el mercado laboral y he aumentado mi valor como profesional para mi empresa.

12. Bibliografía

[1] Curso Android. http://www.androidcurso.com

[2] Android developers:***developer.android.com***

[3] GitHub: https://github.com

[4] PHP: http://php.net

[5] Blog Udemy, *"JSON vs XML: Cómo JSON es Superior a XML"*: blog.udemy.com

[6] StackOverflow stackoverflow.com

[7] Página oficial de Mamp: www.mamp.info

[8] Google Developers: developers.google.com

13. Apartados de la Memoria

La memoria se estructurará de forma coherente, en hojas numeradas. Se recomienda que la memoria no exceda de una extensión de 75 pag. incluyendo anexos.

Se recomienda que la memoria incluya los siguientes apartados:

- **Portada**: el trabajo presentado incluirá en la portada, al menos, el nombre del autor, el título, el nombre del tutor y cotutor, en su caso, la titulación, el curso académico, y los logotipos de la ETSIT y la UPV.
- **Resumen**: con una extensión entre 150 y 200 palabras.
- **Índice de la memoria**.
- **Introducción**.
- **Objetivos** del TFG.
- **Metodología** de trabajo del TFG.
    - Gestión del proyecto.
    - Distribución en tareas.
    - Diagrama temporal.
- **Desarrollo y resultados** del trabajo.
- **Pliego de condiciones** (en su caso) que puede incluir presupuesto, planos, esquemas técnicos, materiales y equipos, etc.
- **Conclusiones y propuesta de trabajo futuro**.
- **Bibliografía** utilizada para la realización del TFG.
- **Anexos** (en su caso).