Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

APLICACIÓN PARA DISPOSITIVO MÓVIL PARA EL CONTROL DEL USO DE MEDICAMENTOS

Alumno: **Antonio Cristóbal Montoro Moreno**

Tutor: Prof. D. Damián Martínez Muñoz
Tutor: Prof. D. José Enrique Muñoz Expósito

Depto.: Ingeniería de Telecomunicación

Junio, 2022

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

APLICACIÓN PARA DISPOSITIVO MÓVIL PARA EL CONTROL DEL USO DE MEDICAMENTOS

Alumno: Antonio Cristóbal Montoro Moreno

Tutor: Prof. D. Damián Martínez Muñoz
Tutor: Prof. D. José Enrique Muñoz Expósito

Depto.: Ingeniería de Telecomunicación

Junio, 2022

Firma del autor:

Firmado por MONTORO MORENO ANTONIO CRISTOBAL - ***2902** el día

Firma de los tutores:

MARTINEZ MUÑOZ DAMIAN - 26202116X
Firmado digitalmente por MARTINEZ MUÑOZ DAMIAN - 26202116X
Fecha: 2022.06.21 23:40:56 +02'00'

MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Firmado digitalmente por MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-27386513E, givenName=JOSE ENRIQUE, sn=MUÑOZ EXPOSITO, cn=MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E
Fecha: 2022.06.22 10:58:31 +02'00'

Índice

Índice de figuras 4
Índice de tablas 5
Resumen 6
CAPÍTULO 1: INTRODUCCIÓN Y OBJETIVOS 7
1.1. Introducción 7
1.2. Objetivos 7
CAPÍTULO 2: ESTADO DEL ARTE 10
2.1. Introducción 10
2.2. Android 10
2.2.1. Estructura de Android 11
2.2.2. Versiones del sistema operativo 12
2.3. Ionic 13
2.3.1. Introducción 13
2.3.2. Ventajas de Ionic 13
2.4. Angular 16
2.4.1. Versiones de Angular 16
2.4.2. Características de Angular 17
2.4.3. Ventajas de Angular 22
2.4.4. Limitaciones de Angular 23
2.5. Visual Studio Code 24
2.6. Android Studio 24
2.7. Introducción al código QR 25
2.7.1. Ventajas principales de los códigos QR 25
2.7.1. Otros tipos de códigos 2D 27
2.8. Wireframes 29
2.9. Enfoques existentes 29
CAPÍTULO 3: TRABAJO REALIZADO 32
3.1. Introducción 32
3.2. Trabajo realizado 33
3.2.1. Diseño de la aplicación 33
3.2.2. Desarrollo de la aplicación 35
3.2.2.1. Primer paso 36
3.2.2.2. Segundo paso 39
3.2.2.2.1. Dificultades en la petición 43

3.2.2.3. Tercer paso 45
3.2.2.3.1. Dificultades en el QRScanner 46
3.2.2.4. Último paso 46
CAPÍTULO 4: CONCLUSIONES Y LÍNEAS FUTURAS 48
4.1. Conclusiones 48
4.2. Proyección a futuro y mantenimiento 48
4.2.1. Actualizaciones 48
4.2.1.1. RFID 48
4.2.1.2. Base de datos propia 49
4.2.2. Lanzamiento 50
4.2.3. Mantenimiento 50
ANEXOS 51
Anexo I: Creación de un proyecto Ionic 51
Anexo II: Instalación de Visual Studio Code y Android Studio 53
Anexo III: Manual de usuario de la aplicación 54
1. ¿Cómo acceder a M-Spec? 57
1.1. ¿Cómo crear una cuenta? 57
2. Interfaz de la aplicación 58
2.1. Primera impresión al acceder 58
2.2. Principales funciones de la interfaz 60
2.2.1. Buscador 61
2.2.1. Buscador manual de medicamentos 63
2.2.1.1. ¿Dónde encontrar el Código Nacional de un medicamento? 64
2.2.2. Escáner de medicamentos 66
2.2.3. Base de datos del ministerio 67

Índice de figuras

Figura 1. Esquema del desarrollo del TFG 8
Figura 2. Plantillas de Ionic 14
Figura 3. Integración de Ionic con lenguajes de programación 14
Figura 4. Creación proyecto Ionic 15
Figura 5. Angular 16
Figura 6. Estructura DOM (Document Object Model) 17
Figura 7. Arquitectura de Angular 18
Figura 8. Ejemplo de componente Angular 19
Figura 9. Ejemplo de template Angular 19
Figura 10. Ejemplo de DOM resultante 20
Figura 11. Ejemplo de interpolación de valores en Angular (1) 20
Figura 12. Ejemplo de interpolación de valores en Angular (2) 20
Figura 13. Ejemplo de interpolación de valores en Angular (3) 21
Figura 14. Ejemplo de ngClass 22
Figura 15. Ejemplo de ngModel 22
Figura 16. Ejemplo de código QR 25
Figura 17. Código de barras y código QR 26
Figura 18. Niveles de corrección de errores de los códigos QR 27
Figura 19. Cuadros de posicionamiento 27
Figura 20. Vista de NinjaMock 29
Figura 21. Ejemplos de funcionalidades de NinjaMock 34
Figura 22. Ejemplo de trabajo realizado en NinjaMock 35
Figura 23. Menú de la aplicación 36
Figura 24. Menú del buscador 37
Figura 25. Listado de medicamentos 37
Figura 26. Detalles del medicamento (Listado) 38
Figura 27. Vista 'buscador manual' 38
Figura 28. Ubicación Código Nacional en envases de medicamentos 39
Figura 29. Página principal CIMA 39
Figura 30. Función encargada de realizar la petición a la base de datos 41
Figura 31. Fichero JSON devuelto tras la petición GET (1) 41
Figura 32. Fichero JSON devuelto tras la petición GET (2) 42
Figura 33. Ejemplo de datos mostrados en pantalla 43
Figura 34. Esquema del BackEnd de la aplicación (1) 43
Figura 35. Fichero 'proxy.conf.json' 44
Figura 36. Esquema del BackEnd de la aplicación (2)(final) 44
Figura 37. Comandos instalación librería QRScanner 45
Figura 38. Import necesario para QRScanner 45
Figura 39. Función encargada de leer códigos QR 45
Figura 40. Parámetros de la función ‘startScanning’ 46
Figura 41. Comando 'ionic build' 46
Figura 42. Paquete @Capacitor/android 47
Figura 43. Proyección a futuro - RFID 49
Figura 44. Pantalla de Registro de M-Spec 57
Figura 45. Interfaz de Inicio de M-Spec (1) 58

Figura 46. Interfaz de Inicio de M-Spec (2) 59
Figura 47. Menú lateral de M-Spec 60
Figura 48. Menú del Buscador de M-Spec (1) 61
Figura 49. Menú del Buscador de M-Spec (2) 62
Figura 50. Buscador manual de M-Spec (resultado) 63
Figura 51. Ayuda al usuario para encontrar el Código Nacional 64
Figura 52. Ubicación del Código Nacional (1) 65
Figura 53. Ubicación del Código Nacional (2) (ejemplo) 65
Figura 54. Escáner QR (1) 66
Figura 55. Escáner QR (2) (resultado) 67
Figura 56. Buscadores nacionales de medicamentos (1) 68
Figura 57. Buscadores nacionales de medicamentos (2) 68

Índice de tablas

Tabla 1. Estructura de Android 11
Tabla 2. Tabla de versiones de Android 13
Tabla 3. Capacidad máxima de datos del código QR (HUIDOBRO, 2009) 26
Tabla 4. Los distintos tipos de códigos 2D (GUTIÉRREZ GARCÍA, 2011) 28
Tabla 5. Atributos CIMA REST API 40

Resumen

Este proyecto se centra en el estudio y desarrollo de una aplicación móvil que permita visualizar parámetros de medicamentos (tales como prospecto y dosis) mediante la lectura de datos a través de etiquetas QR. Todo esto mediante el acceso a información debidamente actualizada y operativa en España.

En primer lugar, se ha realizado un estudio de las diferentes tecnologías existentes para poder realizar el proyecto, así como de la compatibilidad entre ellas y su viabilidad. Una vez seleccionadas las herramientas se ha procedido con el desarrollo.

En cuanto al desarrollo, en la siguiente memoria se detallan todos los pasos y procedimientos llevados a cabo para conseguir los objetivos fijados para esta primera versión, así como los siguientes pasos para posteriormente continuar el trabajo de la creación y puesta en marcha de la App, tomando la idea y el producto desarrollado como la base para la creación de una *StartUp.*

Finalmente, se incluye un apartado de anexos; con manuales de instalación y mantenimiento del software usado y de la App desarrollada, y la bibliografía utilizada en este trabajo final de grado.

CAPÍTULO 1: INTRODUCCIÓN Y OBJETIVOS

1.1. Introducción

La idea

En esta se plantea con exactitud la dimensión que se desea para la app. Se deben tener en cuenta las necesidades de los usuarios o los problemas que se quieren solucionar.

¿Cómo surge la idea?

La idea de realizar esta aplicación nace del concurso de ideas de la Universidad de Jaén: INSIDE UJA, donde en la primera fase de éste se plantea la siguiente necesidad:

"¿Quién no tiene en su casa un armario específico para los medicamentos? ¿No habéis pasado tiempo buscando entre un mar de blísteres? Seguramente muchos de ellos puede que estén caducados y ni lo sepas. Además, no es raro encontrar noticias sobre el uso incorrecto de los medicamentos: Olvido de la toma, confusión con la dosis, equivocación con el fármaco... "

A lo que se plantea la idea de la aplicación en desarrollo en este trabajo, la cual cuenta con los siguientes objetivos:

1.2. Objetivos

Este trabajo fin de grado tiene como objetivo desarrollar una aplicación móvil que permita visualizar parámetros de medicamentos (tales como fecha de caducidad y prospecto) mediante la lectura de datos a través de etiquetas NFC, RFID y/o QR o el propio código de barras.

Se pretende conectar a una base de datos pública donde se encuentra la información de todo medicamento de España para acceder a datos generales relativos al fármaco (como prospecto, dosis, laboratorio desarrollador, etc).

También se contempla la posibilidad de conectar con etiquetas disponibles en los envases de medicamentos que contengan información más específica del producto, como es la fecha de caducidad.

Para la consecución de este objetivo principal es necesario satisfacer los siguientes

objetivos:

- Estudio previo de las tecnologías y técnicas implicadas así como del estado del arte.
- Planificación de las tareas a realizar.
- Selección del entorno de desarrollo y de la tecnología concreta a usar.
- Familiarización con las herramientas seleccionadas.
- Diseño del software de la aplicación para móvil.
- Desarrollo y testeo de la aplicación desarrollada.
- Redacción de la documentación.

A continuación, se proceden a detallar las fases llevadas a cabo en la realización del proyecto (figura 1):

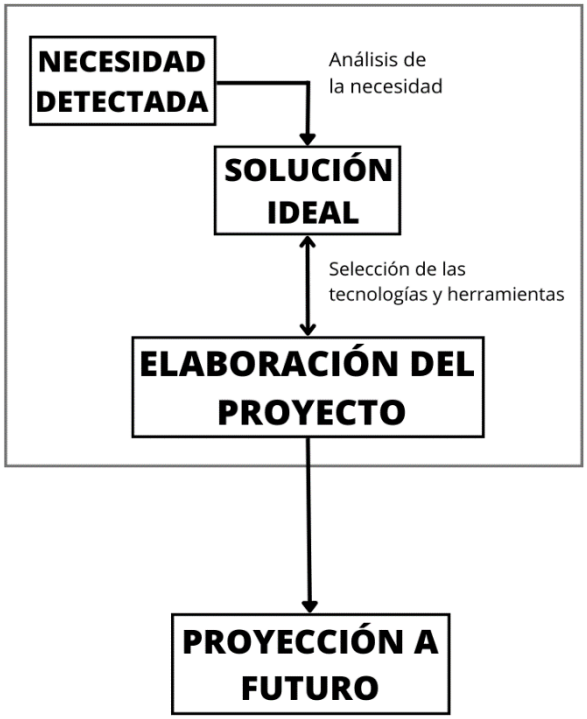

Figura 1. Esquema del desarrollo del TFG

- **Necesidad detectada:**

  En esta primera fase, se percibe la necesidad cotidiana de una herramienta o método efectivo para organizar los medicamentos de una vivienda (en especial para personas con enfermedades crónicas o alergias) y tener mejor control acerca de su información.

  A continuación, se analiza dicha necesidad, detectando además el hecho de que la impresión del prospecto del fármaco supone un gasto importante en papel que puede reducirse mediante la digitalización de éste.

- **Solución ideal:**

  En la fase de Solución ideal, se coge todo lo analizado previamente y se establece una solución que sea novedosa, sencilla y esté al alcance de la mayoría: una aplicación para *smartphone*.

Al tratarse de un desarrollo de una App para móvil, es necesario seleccionar aspectos importantes tales como el entorno de desarrollo, las tecnologías a usar y el lenguaje. Este proceso va intrínsecamente relacionado con el siguiente (Elaboración del proyecto).

- **Elaboración del proyecto:**
  En cuanto a la elaboración del proyecto, es la fase principal ya que es el grueso de este trabajo. Se detalla en el Capítulo 3: Trabajo realizado.

- **Proyección a futuro:**
  Como última fase de este trabajo, y como en cualquier otro proyecto de este tipo, se ha realizado un estudio y planteamiento acerca de qué camino ha de tomar la aplicación tras su primera versión desarrollada para este Trabajo de Fin de Grado.

En cuanto a la estructura seguida en esta memoria, se divide en capítulos, donde en el primer capítulo, como ya hemos visto, se introduce el trabajo realizado, así como su organización y los objetivos a cumplir durante este proyecto. En el capítulo segundo, se da a conocer el marco teórico que ocupa en este proyecto, las tecnologías, entornos y lenguajes empleados, así como diversos enfoques ya existentes en el mercado. En el siguiente capítulo, capítulo número 3, se trata el grueso del proyecto, su desarrollo y la metodología llevada a cabo para la creación de la App. Por consiguiente, en el siguiente capítulo al de desarrollo, se relatan los resultados obtenidos, así como en el capítulo 5, la proyección a futuro que se prevé tomar, con sus respectivos pasos en el tiempo. Por último, en esta memoria, encontramos el apartado de anexos, donde se hallan los manuales de instalación y mantenimiento del software usado en la App desarrollada.

A continuación se exponen en esta memoria todas y cada una de las tecnologías, herramientas y lenguajes empleados para el compilado de la aplicación desarrollada a lo largo de todo este trabajo. A su vez, se muestra el estudio llevado a cabo al analizar diferentes variantes y enfoques ya existentes.

CAPÍTULO 2: ESTADO DEL ARTE

2.1. Introducción

En este punto se describen los paquetes y tecnologías empleadas en el proyecto. Para conocer bien el proyecto es necesario hablar de Android, de su estructura y de sus versiones; así como de Ionic, entorno de desarrollo empleado en la realización del proyecto. El porqué de la utilización de Ionic viene redactado en su punto 2.2. Ionic.

Debido a la elección por Ionic, se debía usar uno de los muchos lenguajes compatibles con éste; se ha optado por Angular (TypeScript) debido al previo conocimiento de este lenguaje por anteriores trabajos.

Una vez seleccionado todo lo anterior, se debía escoger un editor de código fuente, y por el mismo motivo que Angular, se opta por Visual Studio Code, así como por su interfaz sencilla y su fácil integración.

Android Studio, por el contrario, no ha sido elegido sino “impuesto”, ya que para generar el fichero .apk para el sistema Android se ha de hacer por Android Studio. Sucede de manera similar con Xcode e iOS. No se redactan ninguno de estos últimos ya que no han sido empleados en este trabajo por el mero hecho de que solamente se ha desarrollado la app en Android como versión beta.

Por otro lado, es necesario conocer el funcionamiento y estado de un código bidimensional, como un QR, ya que forma parte importante de la aplicación, tal y como se puede ver en el apartado de desarrollo del proyecto.

También, además de todo el estudio de tecnologías a emplear previo, se ha investigado acerca de otras aplicaciones de igual o similar valor en el mercado, para así conocer otros enfoques ya existentes y poder orientar mejor el trabajo: ver qué utilidades buscan los usuarios, así como aspectos a mejorar o añadir a la aplicación.

Por último, se comenta una herramienta bastante útil en este tipo de desarrollos por su ayuda a la hora de esquematizar/diseñar el funcionamiento y orden de ventanas en nuestra App: NinjaMock. Este tipo de herramienta es muy popular para realizar también un PMV (Producto Mínimo Viable) y exponer con más claridad el proyecto ante posibles inversores.

2.2. Android

Android es un sistema operativo basado en el kernel de Linux y diseñado principalmente para dispositivos móviles con pantalla táctil, tales como smartphones o tabletas. Inicialmente desarrollado por Android Inc, y respaldado económicamente por Google, que más tarde, en el año 2005 adquirió la empresa. Uno de los aspectos fundamentales del sistema operativo de Android fue su orientación a la multiplataforma, algo realmente novedoso, debido a que hace unos años, un sistema operativo se asociaba a un único dispositivo. Rápidamente esta característica hizo que Android alcanzase sus objetivos, convirtiéndose en el sistema operativo más utilizado.

2.2.1. Estructura de Android

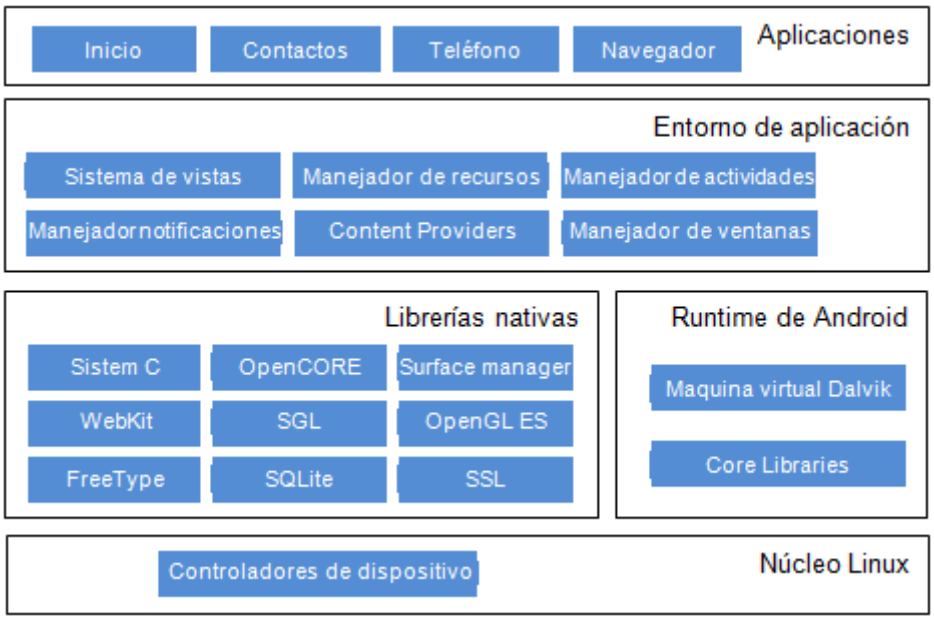

Tabla 1. Estructura de Android

Si se parte de una vista ascendente, lo primero que se encuentra es el núcleo de Android. Basado en el kernel del sistema operativo Linux versión 2.6. Esta capa proporciona servicios de seguridad, manejo de memoria, multiproceso y soporte para controladores de dispositivo.

A continuación se localiza el entorno de ejecución (Android Runtime), basado en la máquina virtual de Java. Dadas las limitaciones de los dispositivos Google decidió crear la máquina virtual Dalvik, la cual respondió mucho mejor a dichas limitaciones. Entre sus características destacan, la optimización de recursos debido a la ejecución de ficheros Dalvik ejecutables (.dex) y la delegación al kernel de Linux procesos como el threading y el manejo de memoria a bajo nivel.

Al mismo nivel se ubican las librerías nativas, escritas en C/C++ y compiladas en código nativo del procesador. Muchas librerías utilizan código abierto. Entre ellas se destacan:

- System C Library: adaptada para dispositivos embebidos en Linux.
- WebKit: Soporta el navegador web de Android y su vista webview. Misma librería que Google Chrome y Safari de Apple.
- SQLite: Ligero pero potente motor de bases de datos relacionales.
- SSL: Proporciona servicios de encriptación (Secure Socket Layer)

El entorno de aplicación (Application Framework) ofrece una plataforma de desarrollo libre para aplicaciones, donde su principal riqueza reside en la reutilización de componentes. Componentes desarrollados por Google, o por usuarios. Los servicios más importantes son:

- Views: conjunto de vistas.
- Location Manager: proporciona servicios de localización a aplicaciones.
- Activity Manager: manejador del ciclo de vida de las actividades.

- Notification Manager: permite mostrar notificaciones a las actividades.
- Content Provider: concede accesos a datos de otras aplicaciones

Por último se encuentra la capa de aplicaciones (Applications) formada por el conjunto de aplicaciones del dispositivo, ya sean nativas o instaladas por el usuario. Todas ellas deben correr en la máquina virtual Dalvik para garantizar la seguridad del sistema. La mayoría de ellas escritas en Java.

2.2.2. Versiones del sistema operativo

Android no ha parado de evolucionar desde el lanzamiento de su primera versión. En Febrero de 2009 Google lanza su primera actualización 1.1 que se diferenciaba de la primera en poder adjuntar archivos en los mensajes. Si echáramos la vista atrás y lo comparásemos con una versión actual, comprenderíamos la inmensa evolución del sistema operativo. Todas las actualizaciones siguientes incorporan funcionalidades nuevas o mejoran las ya existentes.

Todas las versiones de Android reciben del inglés el nombre de diferentes postres, siguiendo, además, orden alfabético.

- A: Apple Pie (v1.0): tarta de manzana.
- B: Banana Bread (v1.1): pan de plátano.
- C: Cupcake (v1.5): panqué.
- D: Donut (v1.6): rosquilla.
- E: Éclair (v2.0/v2.1): pastel francés.
- F: Froyo (v2.2) (abreviatura de «frozen yogurt»): yogur helado.
- G: Gingerbread (v2.3): pan de jengibre.
- H: Honeycomb (v3.0/v3.1/v3.2): panal de miel.
- I: Ice Cream Sandwich (v4.0): emparedado de helado.
- J: Jelly Bean (v4.1/v4.2/v4.3): gominola.
- K: KitKat (v4.4): tableta de chocolate con leche.

Actualmente conviven muchas de las versiones mencionadas en el mercado, debido a que muchos terminales no se han actualizado a nuevas versiones. En la siguiente tabla puede observarse el porcentaje de terminales que usan las diferentes versiones.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.6%
4.3		18	0.5%
4.4	KitKat	19	7.8%
5.0	Lollipop	21	3.6%
5.1		22	14.7%
6.0	Marshmallow	23	21.6%
7.0	Nougat	24	19.0%
7.1		25	10.3%
8.0	Oreo	26	13.4%
8.1		27	5.8%

Tabla 2. Tabla de versiones de Android

2.3. Ionic

En este apartado se explicarán los apartados más importantes del entorno de desarrollo, como el porqué de su elección, sus versiones y las partes principales.

2.3.1. Introducción

Como ya se ha comentado anteriormente, el *framework* seleccionado para la realización del proyecto es Ionic, principalmente por su gran versatilidad. Ionic es un entorno de desarrollo sin costo alguno y de código abierto para el desarrollo de aplicaciones híbridas multiplataforma que utiliza HTML5, CSS (generado por SASS) y Cordova como base.
Es uno de los framework del momento por utilizar AngularJS para gestionar las aplicaciones, lo que asegura aplicaciones rápidas y escalables.

2.3.2. Ventajas de Ionic

- Componentes de interfaz de usuario prediseñados

Los componentes de la interfaz de usuario de Ionic vienen ya pre diseñados para todos los dispositivos y plataformas móviles. Esto permite al usuario comenzar con prefabricados,

tipografía y un tema que se adapte a cualquier plataforma, agilizando así la labor de diseño notablemente.

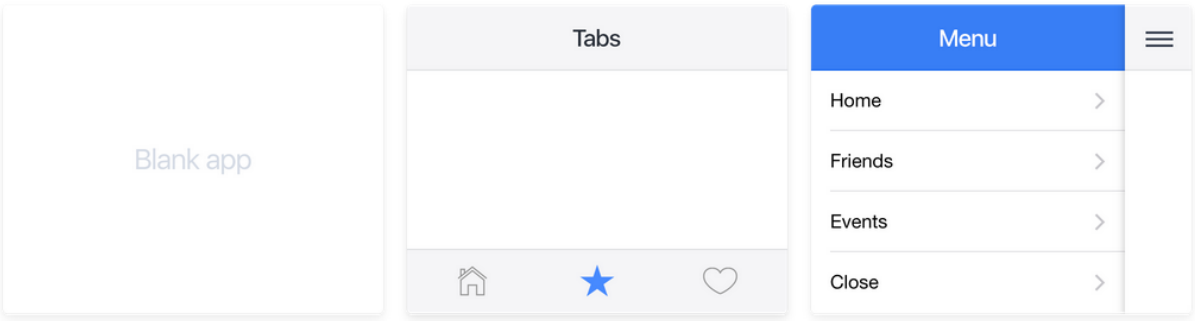

Figura 2. Plantillas de Ionic

- Válido para cualquier mercado

Ionic permite a los desarrolladores enviar sus aplicaciones a cualquier *app store* y como PWA con un único código base. Con *Adaptive Styling*, las aplicaciones se ven y funcionan de igual manera en cualquier dispositivo.

- Herramientas fáciles de usar para desarrolladores

Ionic posee una gran cantidad de herramientas muy útiles para los desarrolladores, como su propio CLI donde se puede crear, probar e implementar la aplicación en desarrollo.

- Múltiples tecnologías

Ionic está diseñado para integrarse a la perfección con todos los mejores marcos *frontend*, incluidos Angular, React, Vue o incluso JavaScript estándar.

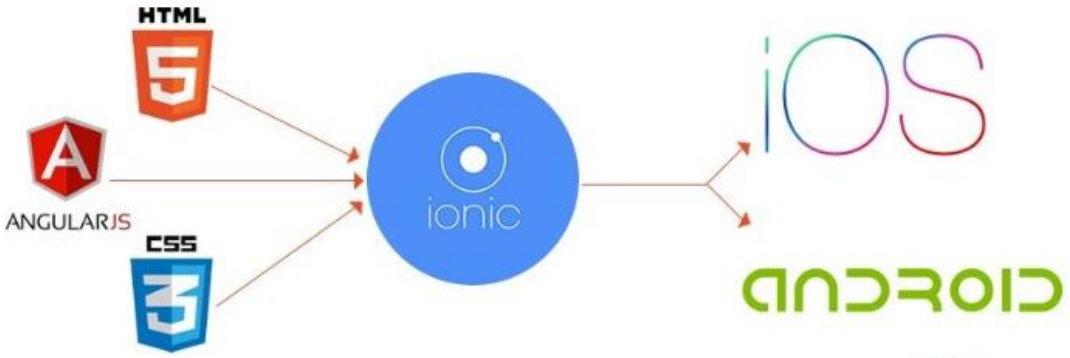

Figura 3. Integración de Ionic con lenguajes de programación

Lo realmente interesante, conjuntamente con todo lo anterior, es que nos proporciona una biblioteca de componentes muy amplia para la elaboración de nuestra interfaz de usuario, y también tenemos acceso a una documentación muy extensa al respecto. Con estos

componentes es muy difícil distinguir una de estas aplicaciones con una aplicación móvil nativa.

La forma de trabajar con Ionic es realmente cómoda, ya que desde una terminal podemos crear proyectos vacíos o con estructuras predeterminadas, añadirle plugins, compilar los proyectos, ejecutarlos en simuladores o en nuestro propio dispositivo si este está conectado con USB a nuestro ordenador (siempre que lo configuremos en modo desarrollador). Por tanto es así de sencillo. Además, a partir de la creación de un proyecto en una carpeta determinada, ya nos estructurará las distintas carpetas de una forma organizada.

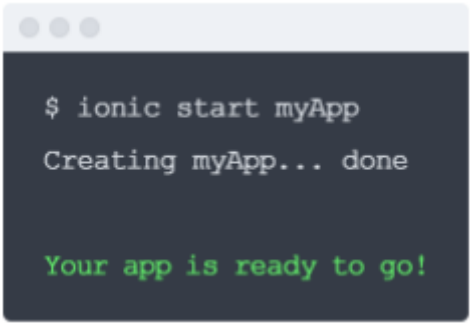

Figura 4. Creación proyecto Ionic

En definitiva este framework nos permite desarrollar aplicaciones híbridas con una arquitectura robusta, utilizando AngularJS, HTML y CSS, con un muy buen rendimiento y con una apariencia de aplicación móvil nativa.

¿Por qué se necesita un Framework?

En concreto, en el desarrollo de software, un framework es una estructura de soporte conceptual y tecnológica definida, generalmente, con artefactos o módulos de software específicos, que pueden servir como base para la organización y desarrollo de software.

Es decir, un framework es una especie de plantilla, esquema o estructura conceptual basada en tecnología que permite trabajar de una forma mucho más sencilla. De esta forma, se evitan posibles errores de programación.

Por tanto, un marco es un conjunto de herramientas y módulos que se pueden reutilizar para diferentes proyectos. Facilitando en varios aspectos el desarrollo, mejorando el tiempo, esfuerzo, organización.

Y, por motivos ya expuestos anteriormente, para la realización de este Trabajo de Fin de Grado se ha optado por Ionic como framework para sostener AngularJS.

2.4. Angular

Figura 5. Angular

Angular es una plataforma de desarrollo, construida sobre TypeScript. Es un framework basado en componentes para crear aplicaciones web escalables. Una colección de bibliotecas bien integradas que cubren una amplia variedad de características, que incluyen enrutamiento, administración de formularios, comunicación cliente-servidor y más. Un conjunto de herramientas para desarrolladores que permiten desarrollar, compilar, probar y actualizar el código fuente de la aplicación.

2.4.1. Versiones de Angular

«Angular» es el término general para las distintas versiones que existen. Angular se desarrolló en 2009 y, como resultado, ha ido evolucionando cada vez más.

Primero, estaba el Angular original, llamado Angular 1 y eventualmente conocido como AngularJS. Luego vinieron Angular 2, 3, 4, 5, 6, 7 hasta que finalmente, la versión actual, Angular 12, lanzada el 12/05/2021. Cada versión posterior de Angular mejora su predecesora, corrige errores, aborda problemas y se adapta a la creciente complejidad de las plataformas actuales.

2.4.2. Características de Angular

Document Object Model (DOM)

DOM (Document Object Model) trata un documento XML o HTML como una estructura de árbol en la que cada nodo representa una parte del documento.

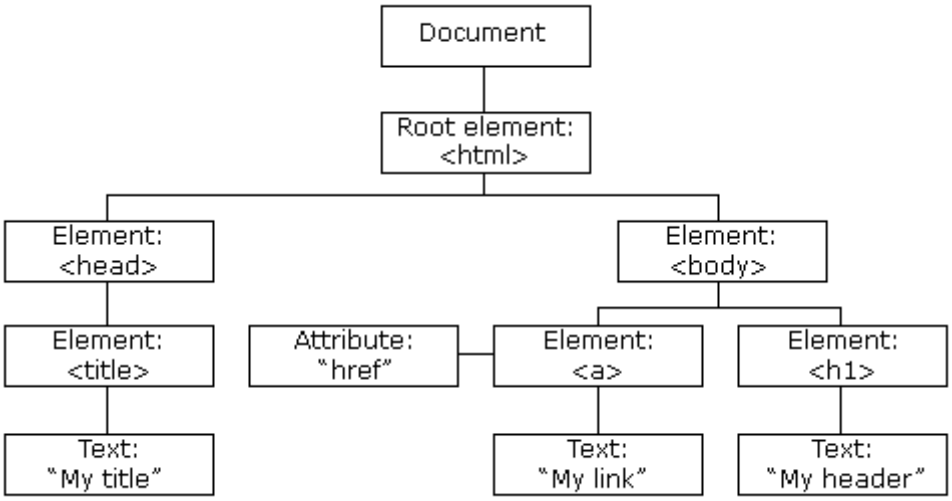

Figura 6. Estructura DOM (Document Object Model)

Angular usa DOM regular. Hay que tener en cuenta que se realizan diez actualizaciones en la misma página HTML. En lugar de actualizar las que ya se actualizaron, Angular actualizará toda la estructura de árbol de las etiquetas HTML. A diferencia de otros frameworks como React.

TypeScript

TypeScript define un conjunto de tipos de JavaScript, lo que ayuda a los usuarios a escribir código JavaScript que es más fácil de entender. Todo el código TypeScript se compila con JavaScript y se puede ejecutar sin problemas en cualquier plataforma. TypeScript no es obligatorio para desarrollar una aplicación Angular. Sin embargo, es muy recomendable ya que ofrece una mejor estructura sintáctica, al tiempo que hace que la base de código sea más fácil de entender y mantener.

Data Binding (Enlace de datos)

El **enlace de datos** (data binding) es un proceso que permite a los usuarios manipular elementos de la página web a través de un navegador web. Emplea HTML dinámico y no requiere secuencias de comandos ni programación complejas. El enlace de datos se utiliza en páginas web que incluyen componentes interactivos, como calculadoras, tutoriales, foros y juegos. También permite una mejor visualización incremental de una página web cuando las páginas contienen una gran cantidad de datos.

Angular usa el enlace bidireccional. El estado del modelo refleja los cambios realizados en los elementos de la interfaz de usuario correspondientes. Por el contrario, el estado de la interfaz de usuario refleja cualquier cambio en el estado del modelo. Esta característica permite que el marco conecte el DOM a los datos del modelo a través del controlador.

Testing (Pruebas)

Angular usa el Framework de prueba **Jasmine**. Jasmine proporciona múltiples funcionalidades para escribir diferentes tipos de casos de prueba. Karma es el ejecutor de tareas para las pruebas que usa un archivo de configuración para configurar la puesta en marcha, los reportes y el framework de prueba.

Arquitectura de Angular

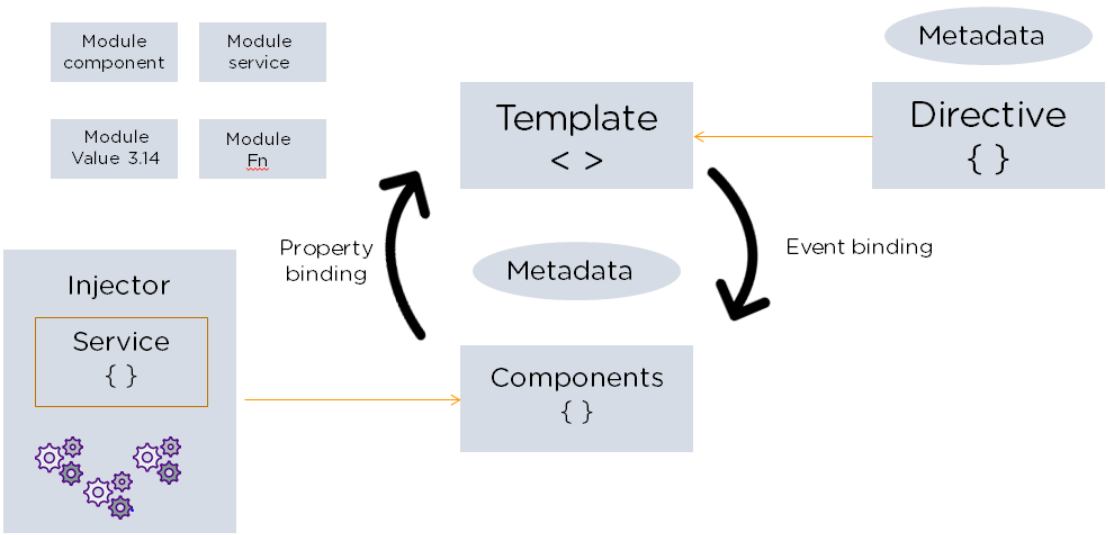

Figura 7. Arquitectura de Angular

Angular es un marco modelo-vista-controlador (MVC) completo. Proporciona una guía clara sobre cómo se debe estructurar la aplicación y ofrece un flujo de datos bidireccional al tiempo que proporciona un DOM real.

Módulos

Una aplicación Angular tiene un módulo raíz, llamado **AppModule**, que proporciona el mecanismo de arranque para iniciar la aplicación.

Componentes

Estos son piezas o fragmentos de código que define una clase que contiene la lógica y los datos de la aplicación. Un componente por lo general define una parte de la interfaz de usuario (UI).

Ejemplo mínimo:

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Figura 8. Ejemplo de componente Angular

Esto genera el siguiente template para así ser usado:

```
<hello-world></hello-world>
```

Figura 9. Ejemplo de template Angular

Cuando Angular haga el render del componente, el DOM resultante será así:

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

Figura 10. Ejemplo de DOM resultante

Plantillas

Es una combinación entre el marcado Angular con HTML para modificar los elementos HTML antes de que se muestren. Hay dos tipos de enlace de datos:

Enlace de eventos: permite que su aplicación responda a la entrada del usuario en el entorno de destino actualizando los datos de su aplicación.

Enlace de propiedad: permite a los usuarios interpolar valores que se calculan a partir de los datos de su aplicación en el HTML.

Ejemplo:

```
<p>{{ message }}</p>
```

Figura 11. Ejemplo de interpolación de valores en Angular (1)

Nótese que el valor de **message** viene del **componente**:

```
import { Component } from '@angular/core';

@Component ({
  selector: 'hello-world-interpolation',
  templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
    message = 'Hello, World!';
}
```

Figura 12. Ejemplo de interpolación de valores en Angular (2)

Cuando la aplicación cargue el componente y su template, el usuario verá lo siguiente:

```
<p>Hello, World!</p>
```

Figura 13. Ejemplo de interpolación de valores en Angular (3)

Metadatos

Los metadatos le dicen a Angular cómo procesar una clase. Se utiliza para decorar la clase para que pueda configurar el comportamiento esperado de una clase.

Servicios

Los servicios sirven para compartir información entre componentes o incluso hacer peticiones http a apis para obtener la información. Los servicios funcionan solo en ámbito de lógica o datos, no están asociados a la vista.

Inyección de dependencia

Esta característica le permite mantener sus clases de componentes nítidas y eficientes. No obtiene datos de un servidor, no valida la entrada del usuario ni se registra directamente en la consola. En cambio, delega tales tareas a los servicios.

Directivas Angular

Las directivas amplían el HTML proporcionándole una nueva sintaxis. Puede detectar fácilmente las directivas porque tienen el prefijo «ng-«. Considérelos marcadores en el elemento DOM, indicando a Angular que adjunte un determinado comportamiento al elemento, o incluso que lo cambie por completo.

Aquí hay dos directivas de muestra:

- **La Directiva modelo ng**

El modelo ng vincula el valor del control HTML con el valor de expresión

src/app/app.component.html

```
<!-- toggle the "special" class on/off with a property -->
<div [ngClass]="isSpecial ? 'special' : ''">This div is special</div>
```

Figura 14. Ejemplo de ngClass

- **La Directiva ng-bind**

Esta directiva reemplaza el valor de control HTML con un valor de expresión

src/app/app.component.html (NgModel example)

```
<label for="example-ngModel">[(ngModel)]:</label>
<input [(ngModel)]="currentItem.name" id="example-ngModel">
```

Figura 15. Ejemplo de ngModel

2.4.3. Ventajas de Angular

Componentes personalizados

Los usuarios tienen total libertad de construir sus propios componentes que pueden empaquetar la funcionalidad junto con la lógica de renderizado en piezas reutilizables en el código

Enlace de datos

Una de las grandes ventajas es que permite a los usuarios mover datos sin esfuerzo desde el código JavaScript a la vista y reaccionar a los eventos del usuario sin tener que escribir ningún código manualmente.

Inyección de dependencia

Angular permite a los usuarios escribir servicios modulares e inyectarlos donde sea que se necesiten. Esto mejora la capacidad de prueba y la reutilización de los mismos servicios.

Pruebas

Las pruebas son herramientas de primera clase y Angular no es la excepción en esto, se ha creado desde cero teniendo en cuenta la capacidad de prueba. Este permite realizar pruebas a cada parte de su aplicación, lo cual es muy recomendable.

Integral

Angular es un framework completo y proporciona soluciones listas para usar para la comunicación del servidor, el enrutamiento dentro de su aplicación y más.

Compatibilidad del navegador

Angular es multiplataforma y compatible con muchos navegadores. Una aplicación angular normalmente se puede ejecutar en todos los navegadores (por ejemplo; Chrome, Firefox) y sistemas operativos, como Windows, macOS y Linux.

2.4.4. Limitaciones de Angular

Curva de aprendizaje empinada

Los componentes básicos de Angular que todos los usuarios deben conocer incluyen directivas, módulos, decoradores, componentes, servicios, inyección de dependencias, pipes y plantillas. Los temas más avanzados incluyen detección de cambios, zonas, compilación de AoT y Rx.js. Por ende, para los nuevos desarrolladores toma tiempo en aprender este nuevo Framework.

Opciones de SEO limitadas

Angular ofrece opciones de SEO limitadas y poca accesibilidad para los rastreadores de motores de búsqueda.

Migración

Una de las razones por las que las empresas no utilizan Angular con frecuencia es la dificultad de trasladar el código heredado basado en js / jquery a una arquitectura de estilo angular. Además, cada nueva versión puede ser problemática de actualizar y varias de ellas no son compatibles con versiones anteriores. Por ende, al desarrollar una webapp con

Angular casi siempre se prefiere dejarla en la versión que se desarrolló en vez de realizar una migración a una nueva versión.

Complejo

Un problema común en la comunidad Angular es la forma en cómo debe escribirse el Framework. También es bastante complejo en comparación con otras herramientas de front-end.

2.5. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Se trata de una plataforma a través de la cual se pueden escribir programas en cualquier lenguaje de programación, permitiendo descargar diferentes tipos de componentes y extensiones, que ayudan a la edición, permiten resaltar el texto que pueda estar mal escrito, corrigen errores y aportan la finalización inteligente de código. Es por esto que es el editor de código seleccionado para la realización de este TFG, junto con Android Studio, aunque este último solamente se utilice para la generación de la APK para cargarla en el dispositivo móvil.

Además, Visual Studio Code permite recorrer los diferentes documentos y carpetas de un proyecto, a través del explorador de archivos. Visual Studio Code combina la simplicidad y el carácter intuitivo de un editor de código fuente con potentes herramientas de desarrollador, como la finalización y depuración de código IntelliSense. Esto permite que sea utilizado de forma sencilla por el usuario para desarrollar el código.

Entre las ventajas de este editor de código destaca la posibilidad de emplear Git, que permite trabajar con el código fuente controlando qué cambios se han realizado y guardarlos sin salir del editor. Finalmente, es importante añadir que Visual Studio Code es un software gratuito, de código abierto y libre distribución, aunque su descarga oficial está bajo software propietario e incluye características personalizadas por Microsoft.

2.6. Android Studio

El entorno de desarrollo o IDE (Integrated Development Environment) utilizado para la implementación de la aplicación en este trabajo ha sido Eclipse junto con el Android SDK (Software Development Kit) que contiene las herramientas necesarias para el desarrollo de apps como son: depurador, emulador o simulador, documentación y ejemplos, y librerías.

La descarga es gratuita y viene detallada en el Anexo II. Android Developer Tools contiene las herramientas de desarrollo y depuración de cada versión de Android dependientes de cada plataforma y están actualizadas para la nueva versión que haya aparecido.

2.7. Introducción al código QR

Los códigos de barras bidimensionales son una nueva generación en el ámbito tecnológico de los códigos de barras. El código QR es un tipo de código, que sirve para almacenar información en una matriz 2D de puntos.
El código QR (Quick Response Code) fue creado en 1994 por la compañía japonesa Denso-Wave. En sentido literal, se refiere a una respuesta rápida que expresa el concepto original de los inventores para el desarrollo del código, cuyo enfoque se dirige a “lectura de alta velocidad” (high-speed reading).

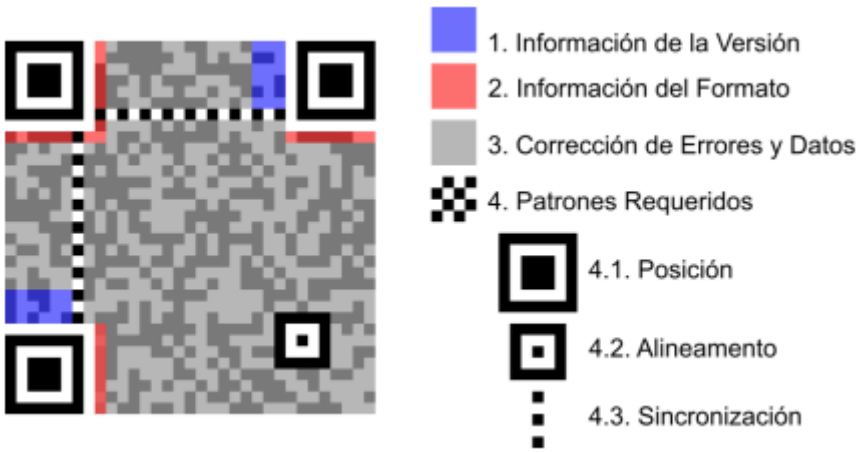

Figura 16. Ejemplo de código QR

2.7.1. Ventajas principales de los códigos QR

- **Mayor capacidad de almacenamiento de la información**

Los códigos de barras tradicionales son capaces de almacenar como máximo 20 dígitos aproximadamente, mientras que el código QR tiene una capacidad de procesar varias decenas o varios cientos de veces más gracias a su diseño, que ofrece la posibilidad de presentar la información de forma horizontal y vertical (2D).

El código QR es capaz de manejar cualquier tipo de información, por ejemplo, los caracteres numéricos y alfabéticos, Kanji, Kana, símbolos, informaciones binarias, etc. Un código QR puede almacenar como máximo 7089 caracteres (cuando se trata únicamente de caracteres numéricos). Dicha capacidad nos permitiría convertir contenidos variopintos en códigos QR y mediante el escáner lograremos una mayor difusión de la información. (Tabla 3)

Capacidad máxima de datos del código QR	
Solo numérico	7.089 caracteres
Alfanumérico	4.296 caracteres
Binario (8 bits)	2.953 bytes
Kanji/Kana	1.817 caracteres
Micro código QR	35 caracteres

Tabla 3. Capacidad máxima de datos del código QR (HUIDOBRO, 2009)

- **Codificación de alta densidad con un tamaño más pequeño**

El código QR posee la capacidad de poder procesar la información en dos direcciones –horizontal y vertical–, el código QR ocuparía solamente el 10% del espacio que ocupa en un código de barras (Fig. 16).

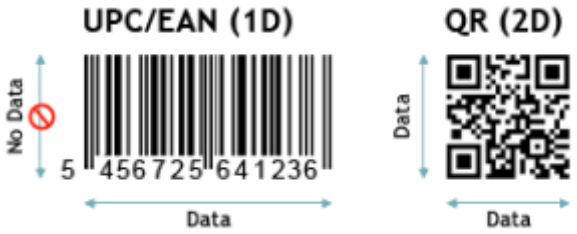

Figura 17. Código de barras y código QR

- **Resistentes a daños y manchas**

El código QR tiene la función de corregir errores. Es decir, tiene la capacidad de recuperar la información aunque una parte del código QR está dañada o manchada en cierto grado. La información puede ser recuperada como máximo hasta el 30% dependiendo del nivel de corrección de errores que han aplicado los usuarios para los códigos QR según casos en distintos contextos. Existen cuatro niveles de la corrección de errores. Cuando tiene el nivel más alto, mayor capacidad de la corrección de errores pero también más densa la imagen del código QR. Los dos módulos marcados en rojo en la Fig. 17 representarán el nivel de corrección de errores en ese código QR.

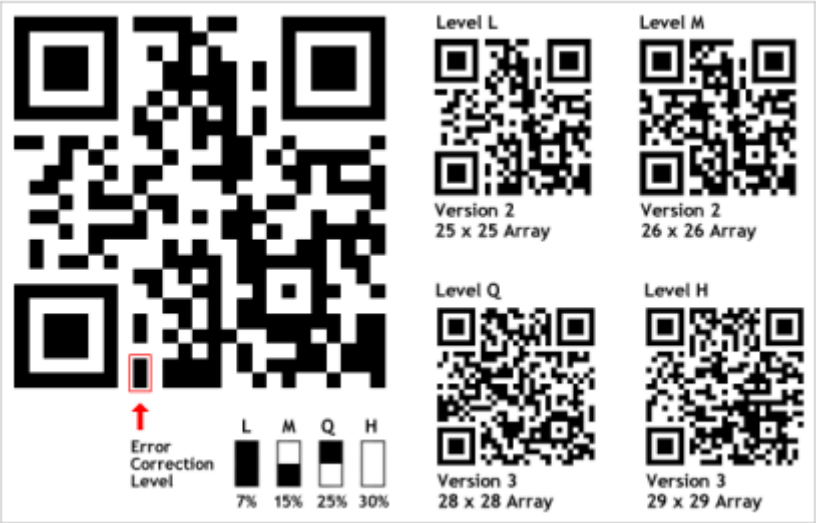

Figura 18. Niveles de corrección de errores de los códigos QR

- **Identificables en 360º**

Los códigos QR son legibles desde todas las direcciones gracias a los tres cuadros situados en las esquinas que sirven como imágenes de posicionamiento (Fig. 18).

Figura 19. Cuadros de posicionamiento

2.7.1. Otros tipos de códigos 2D

Hay que tener en cuenta que el código QR es uno de los distintos tipos de códigos de barras bidimensionales. A través de la Tabla 4 podemos conocer las principales características de los otros tipos de códigos 2D y entender por qué el código QR ha sido uno de los códigos más utilizados en todo el mundo.

		Código QR	PDF417	DataMatrix	Maxi Code
					
Creado (país)		DENSO(Japón)	Symbol Technologies (USA)	RVSI Acuity CiMatrix (USA)	UPS (USA)
Tipo		Matriz	Código de Barras Apilado	Matriz	Matriz
Capacidad de Datos	Numéricos	7089	2710	3116	138
	Alfanuméricos	4296	1850	2355	93
	Binarios	2953	1018	1556	
	Kanjis	1817	554	778	
Características básicas		Alta capacidad, pequeño espacio de imprimación. Alta velocidad de escaneado	Alta capacidad	Pequeño espacio de imprimación.	Alta velocidad de escaneado
Usos frecuentes		Todas las categorías	OA	FA	Logística
Estandarización		AIM Internacional JIS ISO	AIM Internacional ISO	AIM Internacional ISO	AIM Internacional ISO

Tabla 4. Los distintos tipos de códigos 2D (GUTIÉRREZ GARCÍA, 2011)

Actualmente los códigos QR se ven en todos los sitios: en los periódicos, los carteles, los billetes de avión, las revistas, las paradas de autobús, etc. Con capturar simplemente ese código (normalmente representa un URL) a través de la cámara de un Smartphone, la gente puede obtener la información que quiere transmitir ese código en un instante.

Con el desarrollo del Internet móvil, la integración de los Smartphones y los códigos QR, el valor práctico de estos ha cobrado mayor amplitud. Se trata de una nueva tecnología muy útil, tanto para el almacenamiento como la transmisión y la identificación de la información. Los códigos QR podrán promover el proceso de mejora de los servicios en los centros de información de una manera eficaz y hacerles realmente reflexionar cómo satisfacer al máximo las necesidades de las masas lectoras.

2.8. Wireframes

NinjaMock

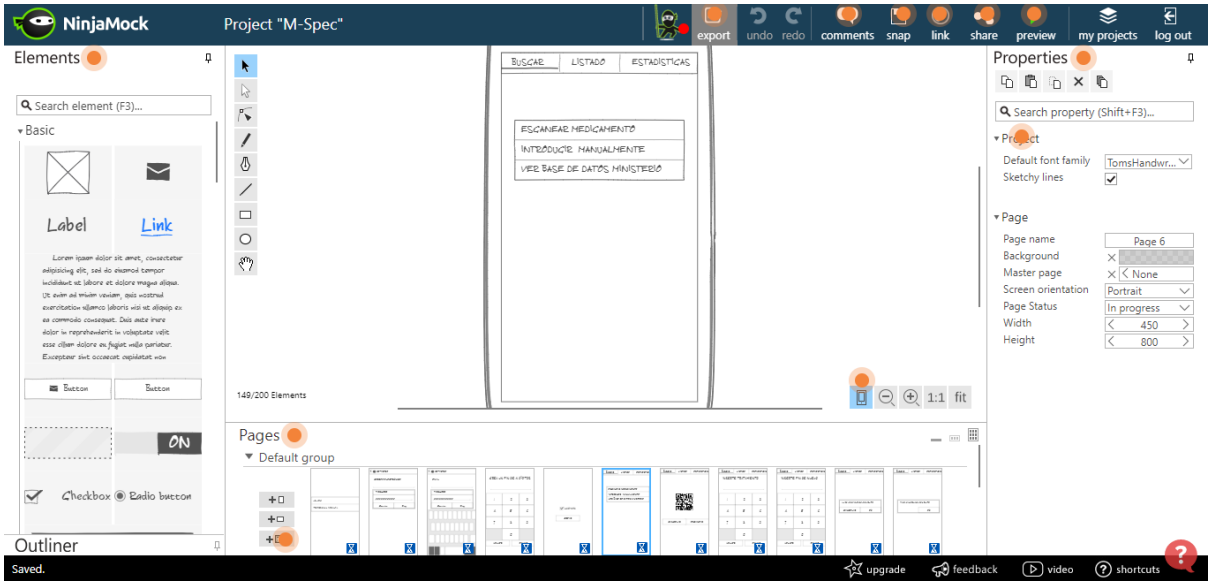

Figura 20. Vista de NinjaMock

NinjaMock es un constructor gráfico de maquetas. Permite al diseñador organizar y diseñar las maquetas utilizando su potente herramienta drag&drop, con la que puede arrastrar elementos y moverlos a su antojo dentro de la maqueta. Dentro de los numerosos editores de maquetas existentes, NinjaMocks permitía la creación de un proyecto para Android, pudiendo hacer uso de la mayoría de widgets que posee este sistema operativo. Gracias a esta plataforma, se ha podido establecer un boceto acerca de las vistas de la aplicación lo que ha facilitado mucho la organización a la hora de su programación.

2.9. Enfoques existentes

En el ámbito de las aplicaciones dedicadas al control de medicamentos hay dos vertientes, una de ellas dedicada al recordatorio de la toma de la medicación, donde podemos encontrar un gran repertorio, y otra encargada de mostrar la información relativa al fármaco al usuario.

Sobre el primer enfoque encontramos varias aplicaciones:

MediSafe: Esta app de recordatorio para la toma de pastillas es otra de las más populares.
Gracias a la misma, no te olvidarás nunca de tomar un medicamento a su hora, ya que mostrará una alarma cada vez que toque una toma.

Como ventaja adicional, MediSafe te permite llevar un control de todos los medicamentos y avisarte cuando te queda poca cantidad de alguno.
Además, puede conectarse con otros móviles para que la familia y/o cuidadores lleven un control exhaustivo de si se están haciendo todas las tomas correctamente.
MediSafe es gratuito para sus funciones de recordatorio de tomas. Tiene algunas funciones adicionales para las que sí será necesario adquirir la versión de pago.

MedsReminder: Otra aplicación sencilla y muy intuitiva de configurar y para añadir todos nuestros recordatorios y alarmas de medicamentos es Meds Reminder.

Con ella, podrás programar todas las tomas de medicamentos en un calendario y recibir una alarma cada vez que sea necesario.

BiVa: Con esta completísima app de recordatorio de toma de medicamentos no solo tendrás una función de alarma. También te ofrece un calendario completo en el que podrás ver de un solo vistazo todas las tomas planificadas, de forma que puedas llevar un control más exhaustivo sobre ellos.

Biva no solo sirve como recordatorio para los medicamentos, también puedes incluir otro tipo de actividades relacionadas con tu salud. Otra de sus funcionalidades destacadas es que además puedes añadir a otras personas, por ejemplo, cuidadores o familiares, para que puedan ver cómo se está llevando a cabo la toma de medicamentos y controlar que se está realizando correctamente.

Sobre la segunda vertiente encontramos:

AEMPS CIMA: Esta aplicación incluye todos los medicamentos autorizados en España y su finalidad es la de proporcionar información de cada uno de ellos. Los medicamentos, cuyas autorizaciones de comercialización han sido revocadas, se mantienen a título informativo 5 años. En el caso de aquellos medicamentos cuya autorización haya sido temporalmente suspendida se mantienen durante la suspensión.

Para los pacientes y demás usuarios, la aplicación ofrece acceso a la última versión del prospecto autorizado, que es el documento que se incluye dentro de la caja del medicamento y cuya finalidad es la de informar al paciente.

Información disponible:

Esta aplicación contiene la información de los medicamentos autorizados en España con la finalidad de constituir una herramienta de valor para utilizarlos correctamente.

Para cada medicamento se incluyen los siguientes datos y documentos:

- Nombre del medicamento
- Nombre del principio o principios activos
- Nombre del titular de la autorización de comercialización
- Código nacional
- Número de registro
- Fecha de autorización
- Nombre de la presentación
- Estado de la presentación: autorizada/suspendida temporalmente/revocada
- Datos sobre las condiciones de prescripción y uso
- Necesidad de receta y tipo
- Pictograma de advertencia de conducción
- Ficha técnica (profesionales sanitarios)
- Prospecto (ciudadanos no sanitarios)

CAPÍTULO 3: TRABAJO REALIZADO

3.1. Introducción

A la hora de realizar el trabajo de desarrollo y creación de la aplicación que nos ocupa en este Trabajo de Fin de Grado, se ha considerado oportuno seguir el orden de trabajo empleado por cualquier otra empresa del sector de desarrollo de aplicaciones.

Las fases de desarrollo de una aplicación móvil abarcan desde el presupuesto y concepción de la idea, hasta el mantenimiento de la app. A continuación se presentan dichas etapas:

- **Elaboración del presupuesto**: Antes de pensar el tipo de aplicación que se desea, hay que analizar los recursos que se destinarán al proyecto. Partiendo del presupuesto, se podrá desarrollar una aplicación móvil de mayor envergadura y con múltiples funcionalidades u otra más sencilla. Una vez establecido, es el momento de determinar la idea que se quiere llevar a cabo. Dado que en este trabajo no hay clientes externos, esta fase no ha sido necesaria llevarla a cabo, pero es importante mencionarla.
- **La idea:** En la segunda de las fases de desarrollo de una aplicación móvil, se plantea con exactitud la dimensión que se desea para la app. Se deben tener en cuenta las necesidades de los usuarios o los problemas que se quieren solucionar. Partiendo de esas características, se pasará a definir un concepto determinado y el valor añadido que puede aportar.
- **Elegir la tecnología:** Una vez debidamente escogida la idea, es necesario seleccionar las herramientas que nos permitan hacerla realidad. A esto se le dedica el Capítulo 2: Estado del arte de esta memoria por completo.
- **Análisis de los requerimientos:** Tras una investigación del sector y de la competencia, es momento de definir el alcance del proyecto. Pasa de lo general a lo concreto, qué se va a ofrecer en cada una de las pantallas. Con las funcionalidades bien definidas, la fase de desarrollo será más fluida.
- **La planificación:** Cualquier proyecto requiere de una fase de *planning* en la que se establecerán las directrices. Un calendario de trabajo que especifica la lista de acciones a realizar hasta el cierre del proyecto.

Hasta aquí, todas las fases se han ido viendo respectivamente en los anteriores capítulos de esta memoria, pero se requería de su debida mención en este capítulo en el que detalla el trabajo realizado.

- **UX y diseño de la aplicación**: En esta fase de desarrollo de una aplicación móvil se definirán el contenido y las interacciones de la app acorde a la experiencia de usuario. Se elabora la propuesta visual de las pantallas, siguiendo la imagen de marca y las tendencias del diseño de apps, siempre pensando en ofrecer la mejor experiencia para los usuarios. Estos diseños se crearán mediante *wireframes* y prototipos, que serán la base para el desarrollador (NinjaMock, 3.2.1. Planteamiento).
- **Desarrollo app:** El desarrollador mediante código comienza a construir la aplicación móvil con el lenguaje de programación y tecnología indicados en la fase de análisis.

Es muy recomendada la existencia de entregas parciales y periódicas para que se vea la evolución del producto. Gracias a ello, se pueden corregir a tiempo los fallos que surjan según que avanza el desarrollo. Por esto, este trabajo en la parte de desarrollo se ha dividido en varios pasos debidamente redactado a continuación en 3.2.2. Desarrollo de la aplicación.

Hasta aquí, estos dos puntos son el principal objetivo de este capítulo, Capítulo 3: Trabajo realizado, ya que son las fases más técnicas del desarrollo de una aplicación y en esta memoria se ha considerado dedicar el capítulo 3 entero a los aspectos más técnicos del trabajo.

- **Testing:** En esta etapa se realizan una serie de pruebas que aseguren la calidad de la plataforma. QA (Quality Assurance) es un proceso de evolución y mejora continua donde se realizan acciones para comprobar que todas las acciones que pueda hacer un usuario dentro de la app funcionen correctamente en todos los dispositivos.
- **Lanzamiento:** Una vez se garantice la calidad, es momento de ofrecer la app a los usuarios. Se puede subir en las tiendas de aplicaciones como App Store y Google Play, por lo que se recomienda tener en cuenta el tiempo de publicación y aprobación de ambas para la fecha de lanzamiento de la app.
- **Mantenimiento:** Todo software se actualiza, por ejemplo, se van publicando nuevos sistemas operativos, nuevas funcionalidades o desarrollos, cambian las políticas de las tiendas o se venden versiones superiores de los modelos de smartphones.

Por último, estas tres fases finales del desarrollo de una App, son el contenido principal del capítulo número 5, Conclusiones y líneas futuras, donde se entra más en detalle acerca de estas etapas finales del trabajo.

3.2. Trabajo realizado

3.2.1. Diseño de la aplicación

En primer lugar se comenzó planteando la organización e interacción entre las diferentes vistas que tendrá la App, para ello se usó la herramienta mencionada anteriormente en el apartado 2.8. Wireframes**,** NinjaMock.

En NinjaMock, prácticamente como en cualquier otro *wireframe* o herramienta de esquemáticos para aplicaciones, se pueden encontrar diversas funcionalidades (Figura 21).

La primera funcionalidad reseñable de esta herramienta es el hecho de constar con ciertos elementos básicos en cualquier aplicación, tales como cuadros de texto, uso de teclado, menús, desplegables, etc.

Dichos elementos, se van emplazando en páginas que la plataforma nos permite crear, de tal manera que enlazando entre unas y otras, así como haciendo uso de los elementos según indiquemos, podemos ir creando una pseudo aplicación que aparentemente desarrolla todas las funcionalidades de una real, salvo por el hecho de que no hay nada de código detrás. Realizar este diseño previo, sin nada de código ni desarrollo tecnológico,

permite adelantar la manera de proceder en la aplicación y evitar posibles errores, una vez comenzado el desarrollo en sí.

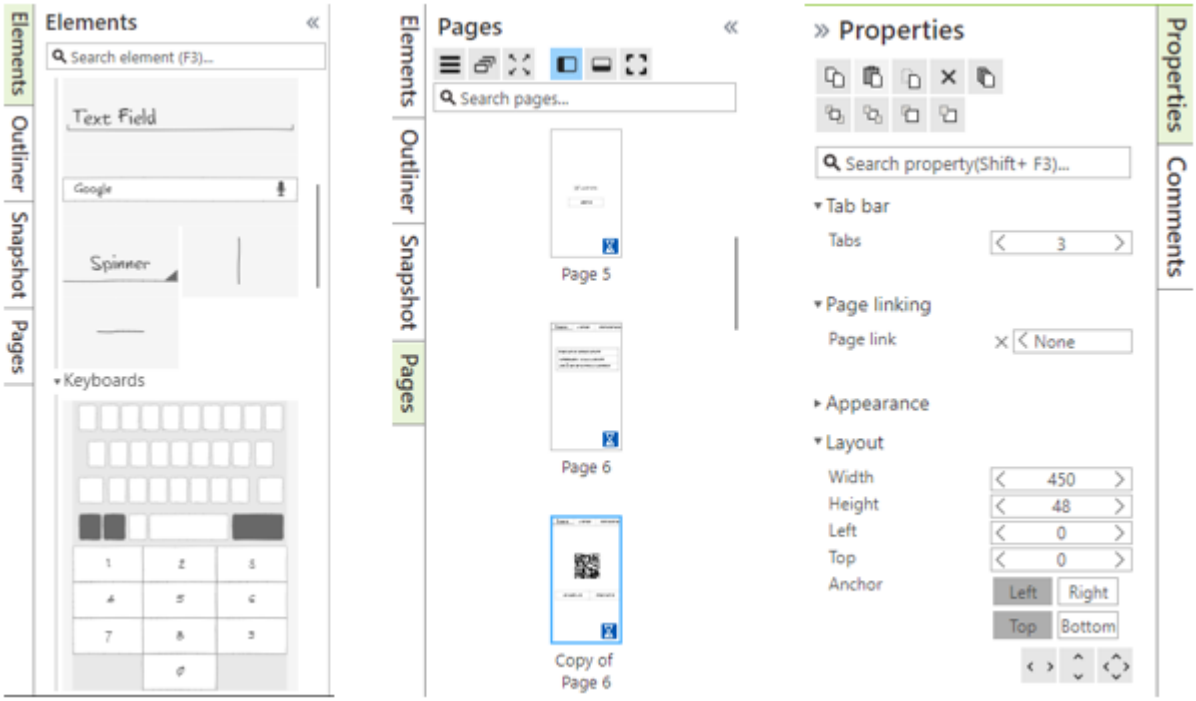

Figura 21. Ejemplos de funcionalidades de NinjaMock

En NinjaMock, para esta aplicación en concreto se ha procedido de la siguiente manera:

- En primer lugar se desarrolló el menú principal, en el cual se indicaba el salto entre vistas y la relación entre ellas. También se usó para detallar las diferentes opciones que ofrecería el apartado 'Buscador' (Figura 22).
- Posteriormente, se usaron los diversos elementos ofrecidos por esta herramienta para la ubicación de los aspectos importantes en las diversas páginas. Cuadros de texto para la página buscador manual y elementos de lista para la página de listado, para así saber en la fase de desarrollo de la app dónde el usuario ha de introducir manualmente el indicador del medicamento deseado y dónde encontrar los medicamentos escaneados, respectivamente (Figura 22).
- Por último, una vez establecido la manera de interactuar entre páginas y cómo se comportarán todos sus elementos en la fase de desarrollo de la app, se realiza una simulación para ver si el resultado de este diseño en *wireframe* es el deseado. Una vez se da el visto bueno, se emplea este diseño como base para la etapa de desarrollo, permitiendo así proceder de una manera directa y más eficaz.

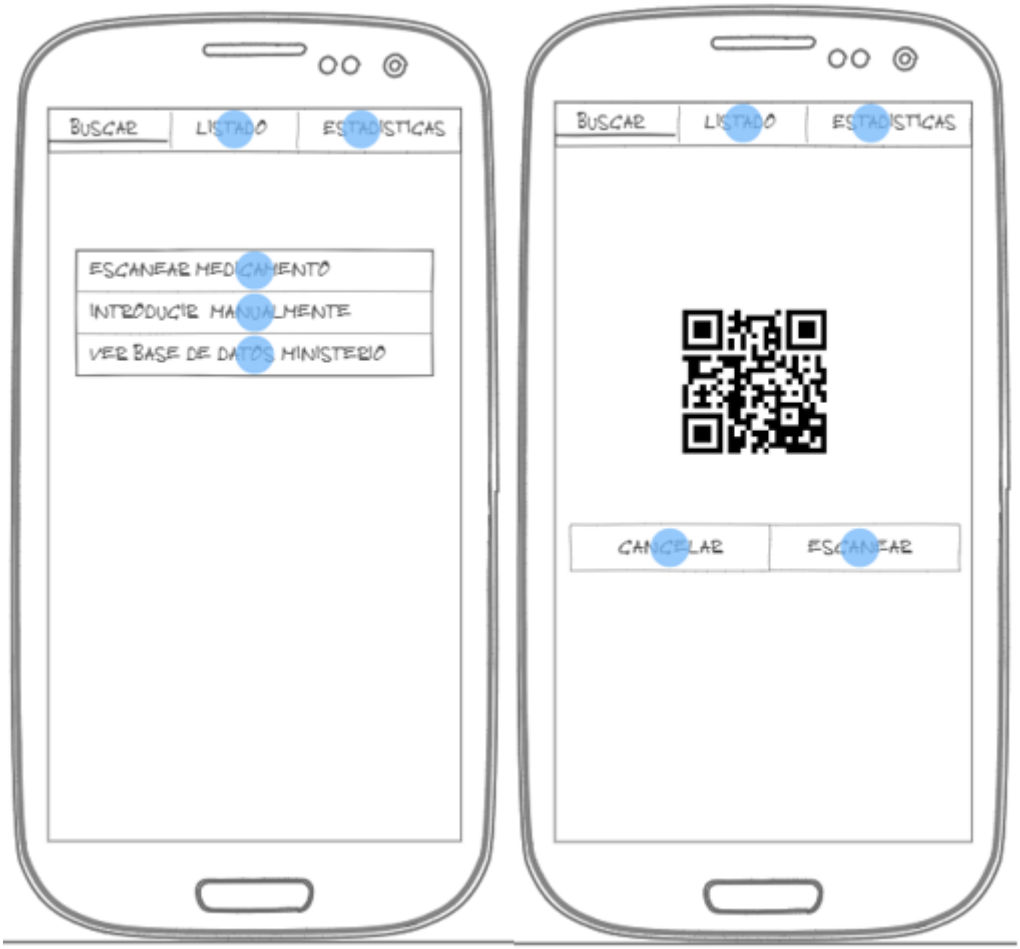

Figura 22. Ejemplo de trabajo realizado en NinjaMock

3.2.2. Desarrollo de la aplicación

En este apartado se detalla el grueso del trabajo, el desarrollo llevado a cabo para conseguir los objetivos propuestos partiendo de la idea original, el desarrollo de la aplicación en sí. En cuanto a la manera de proceder, se ha segmentado el trabajo de programación para así ir cumpliendo objetivos, que una vez comprobados, deriven en un producto final terminado y listo para la fase de *testing*, tal y como se desea. Los respectivos pasos (explicados en detalle a continuación), tratan el desarrollo y organización de todas y cada una de las vistas de la interfaz de la aplicación (primer paso) que sirve de base para el conexionado e interacción con la base de datos de la AEMPS (Agencia Española de Medicamentos y Material Sanitario) (segundo paso), y la creación de un lector de códigos QR que permita al usuario obtener los datos del medicamento de una manera directa y actual (tercer paso). La elaboración de todos estos pasos deriva en implementar un fichero '.apk' (APK, Android Application Package, los archivos APK contienen una aplicación para ser instalada en un dispositivo) el cual nos permita ejecutar la aplicación desarrollada en un smartphone (último paso).

3.2.2.1. Primer paso

Una vez hecho el modelo de NinjaMock, se considera como primer paso la realización de un menú en la aplicación tal y como se ha descrito anteriormente:

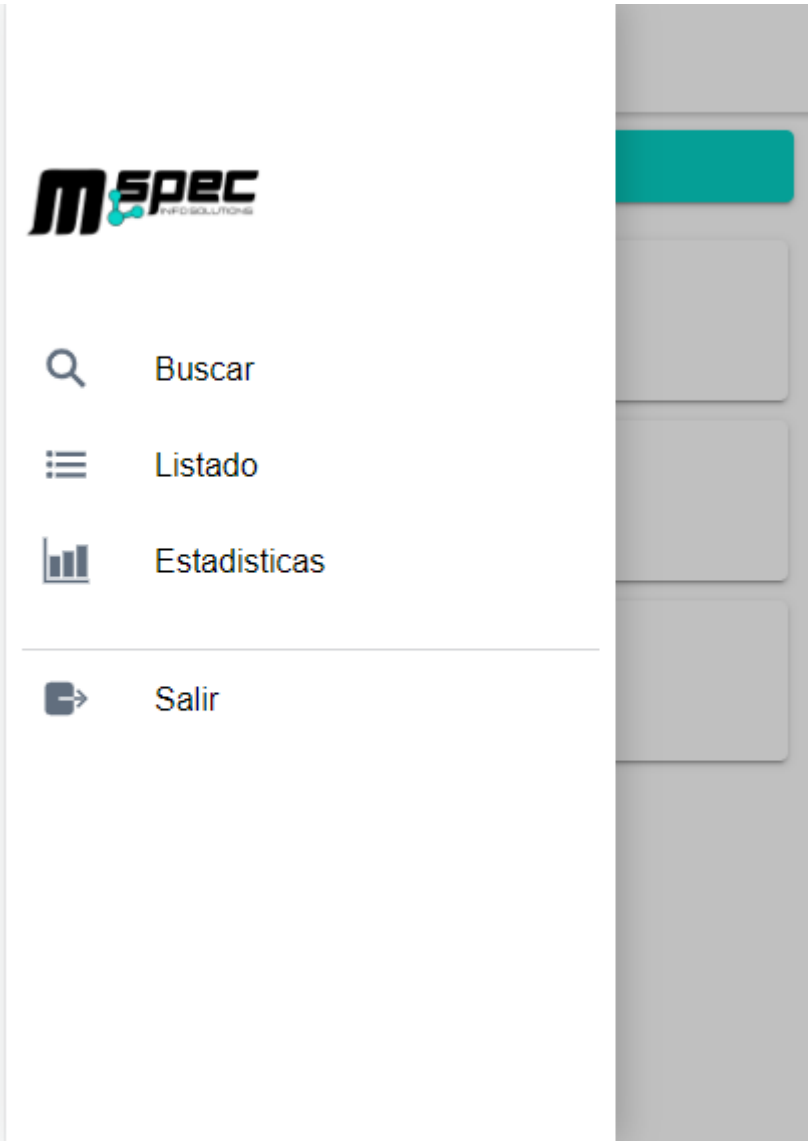

Figura 23. Menú de la aplicación

En orden encontramos:

- La opción de 'Buscar', la cual nos deriva a un submenú llamado buscador que permite al usuario escoger entre varias opciones de búsqueda de medicamentos, encontramos un submenú tal y como el que vemos en la Figura 24.
- La opción de 'Listado', en este bloque se encuentran los medicamentos que hemos escaneado y que podremos consultar su respectiva información (Figuras 25 y 26).
- La opción 'Estadísticas', muestra el índice de actividad llevada a cabo, para así tener un control más exhaustivo (previsto su desarrollo para usuarios *premium*).

Figura 24. Menú del buscador

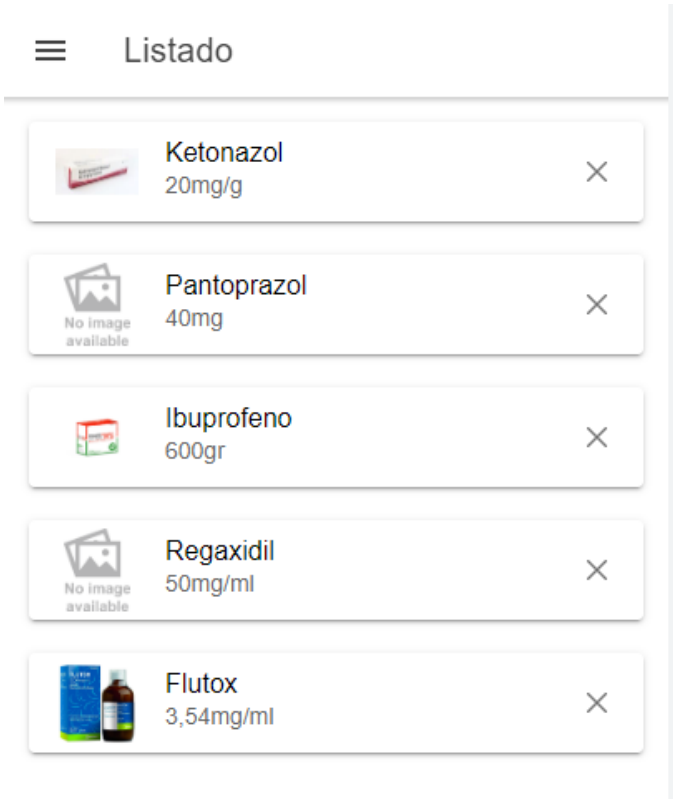

Figura 25. Listado de medicamentos

Figura 26. Detalles del medicamento (Listado)

El submenú 'Buscador' brinda al usuario las opciones de:

- Escanear medicamentos: aquí es donde podemos leer las etiquetas situadas en las cajas con nuestro smartphone y así obtener toda la información necesaria automáticamente.
- Introducir manualmente: en este apartado realizaremos exactamente lo mismo que en el anterior, salvo por introducir manualmente el Código Nacional del medicamento (Figura 26) en la aplicación.

Figura 27. Vista 'buscador manual'

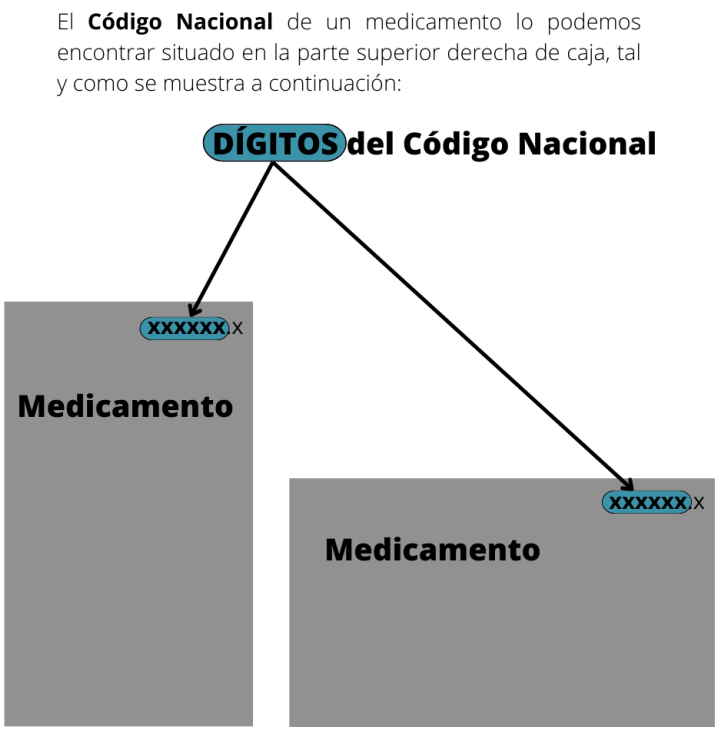

Figura 28. Ubicación Código Nacional en envases de medicamentos

- Ver base de datos del ministerio: aquí se nos presenta un enlace mediante el cual podemos acceder directamente al buscador de medicamentos público.
  https://cima.aemps.es/cima/publico/home.html

Figura 29. Página principal CIMA

3.2.2.2. Segundo paso

Una vez creado el menú de la aplicación y sus respectivas vistas, se ha procedido a realizar la petición a la base de datos pública de CIMA.AEMPS.ES. El acceso a la base de datos se realiza a través de unos servicios web accesibles mediante las direcciones:

- https://cima.aemps.es/cima/rest/medicamento?nregistro=num_registro,
  donde num_registro se corresponde al número de registro del medicamento.

- https://cima.aemps.es/cima/rest/medicamento?cn=cod_nacional,
  donde cod_nacional es el código nacional del medicamento.

Los atributos disponibles a la hora de realizar la consulta a dicha base de datos se encuentran debidamente detallados en la siguiente tabla:

Atributos

Nombre	Tipo	Descripción
nregistro	Texto	Nº de registro del medicamento
cn	Texto	Código nacional de la presentación
nombre	Texto	Nombre de la presentación
pactivos	Texto	Lista de principios activos separada por comas
labtitular	Texto	Laboratorio titular del medicamento
estado	estado	Estado de registro de la presentación
cpresc	Texto	Condiciones de prescripción del medicamento
comerc	Lógico	Indica si está o no comercializada la presentación
conduc	Lógico	Indica si el medicamento afecta o no a la conducción
triangulo	Lógico	Indica si el medicamento tienen asociado el triangulo negro
huerfano	Lógico	Indica si el medicamento está considerado como medicamentos huérfano
ema	Lógico	Indica si el medicamento se ha registrado por procedimiento centralizado (EMA) o no
psum	Lógico	Indica si la presentación tiene problemas de suministro abiertos
docs	documento[]	Lista de documentos asociados al medicamento
notas	Lógico	Indica si existen notas asociadas al medicamento

Tabla 5. Atributos CIMA REST API

Por ejemplo:
https://cima.aemps.es/cima/rest/medicamento?nregistro=51347 , donde nregistro=51347 corresponde al número de registro de ASPIRINA C 400 mg/240 mg COMPRIMIDOS EFERVESCENTES, y variará en función del medicamento a escanear.
https://cima.aemps.es/cima/rest/medicamento?cn=712729 , siendo cn=712729 el código nacional correspondiente con ASPIRINA C 400 mg/240 mg COMPRIMIDOS EFERVESCENTES igualmente.

Una vez establecido el acceso a la base de datos pública de CIMA.AEMPS.ES, el siguiente paso es la creación de una función en código que permita al usuario de la aplicación ser el encargado de introducir los datos a consultar y la aplicación los procese (Figura 30).

```
onSubmit(){
  this.submitted = true;
  console.log('submitted')
  this.enlace = this.url+this.medicamento.value;
  console.log(this.enlace)
  this.http.get(this.enlace).subscribe(data =>{
    if (data == null ){
      console.log("null")
      this.toast.showToastError("Introduce un numero de registro válido")
    }else{
      this.info =          data;
      this.nombre =        data['nombre'];
      this.labtitular =    data['labtitular']
      this.docs =          data['docs'][0]['urlHtml'];
      this.fotos =         data['fotos'][0]['url'];
      this.cpresc =         data['cpresc'];
      console.log(this.info);
    }
  }, err =>{
    console.log("error");
    this.toast.showToastError("Algo ha salido mal, intentelo de nuevo")
  })
}
```

Figura 30. Función encargada de realizar la petición a la base de datos

Como resultado de todo lo anterior, el acceso a la base de datos de CIMA.AEMPS.ES pasando sus respectivos atributos y la función que lo realiza, se obtiene una respuesta a los servicios web con un mensaje JSON tal que así:

Figura 31. Fichero JSON devuelto tras la petición GET (1)

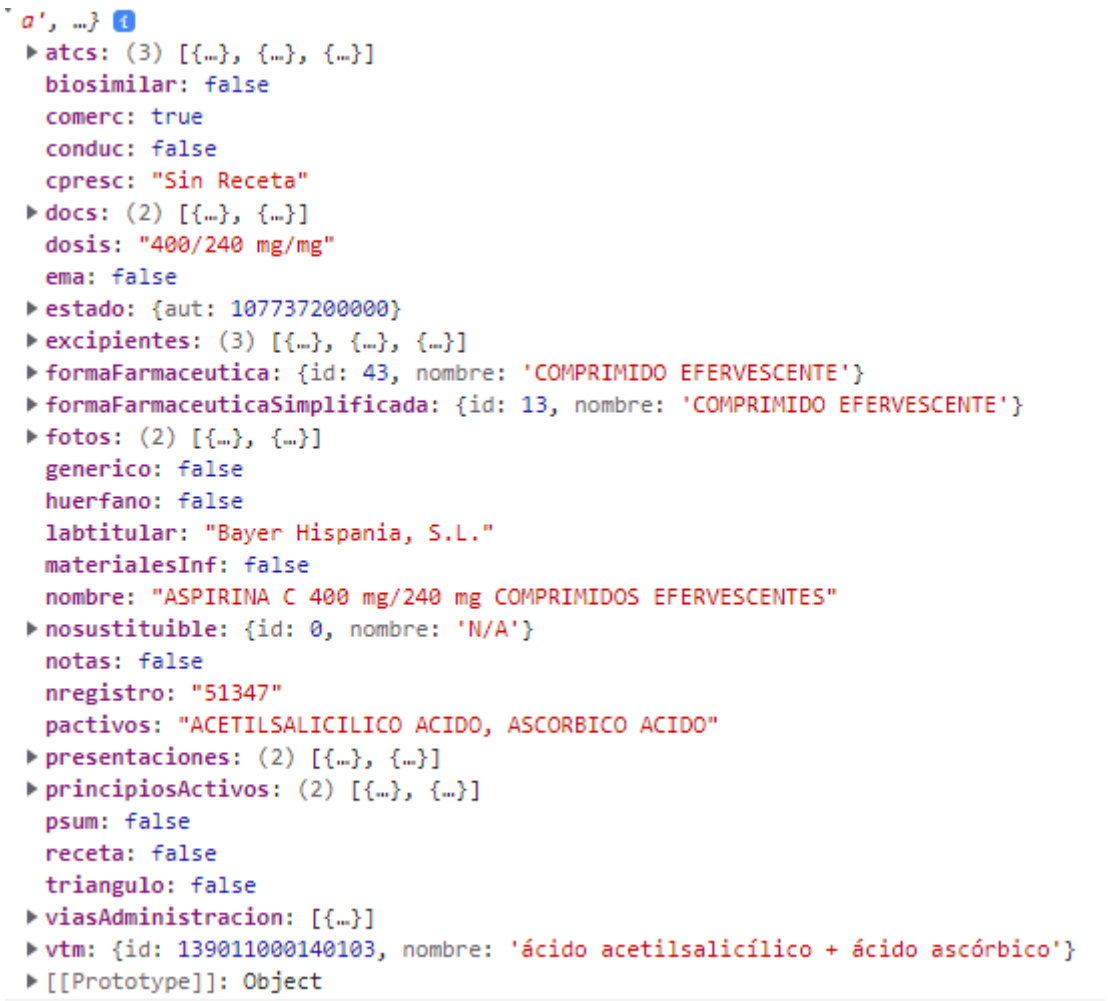

Figura 32. Fichero JSON devuelto tras la petición GET (2)

Este fichero JSON obtenido como resultado de una petición exitosa, se analiza y se seleccionan los atributos pertinentes para ser mostrados en pantalla para que el usuario final los pueda consultar debidamente. Tal y como vemos en la Figura 33, los datos seleccionados son:

- Foto/imagen: se corresponde con ‘fotos’ en el fichero JSON, si la(s) hubiera.
- Nombre y dosis del medicamento: se corresponde con ‘nombre’ en el fichero JSON.
- Laboratorio fabricante: ‘labtitular’ en el JSON.
- Prospecto: se corresponde con ‘docs’ y a su vez con ‘urlHtml’ en la respuesta JSON.
- Receta: llamada ‘cpresc’ en el fichero JSON.

Figura 33. Ejemplo de datos mostrados en pantalla

3.2.2.2.1. Dificultades en la petición

A la hora de la realización de la petición a la base de datos del CIMA, los primeros intentos no dieron los resultados esperados. Esto se debía a un error por petición cruzada (CORS error). Los errores de uso compartido de recursos entre orígenes (CORS) ocurren cuando un servidor no devuelve los encabezados HTTP que exige el estándar CORS. Para resolver un error CORS de una API REST de API Gateway o API HTTP, se debe configurar de nuevo la API para que cumpla con el estándar CORS.

Sin embargo, en este trabajo se ha optado por una solución más efectiva y sencilla para solucionar este error: implementando un servidor proxy intermedio que realice la petición directamente por la aplicación. Es una manera directa de eliminar dicho error y que ha agilizado notablemente el trabajo, obteniendo el resultado esperado.
Gráficamente sería un conexionado tal que así:

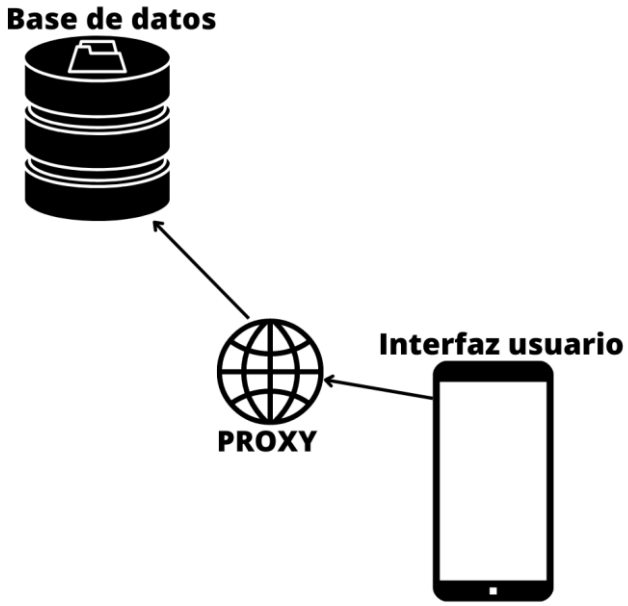

Figura 34. Esquema del BackEnd de la aplicación (1)

Además, durante la resolución del mencionado error, se ha constatado que un proyecto *Ionic* permite la creación de un fichero dentro de la carpeta de proyecto en el cual directamente programar el servidor proxy, lo cual hace que se ejecute conjuntamente con la aplicación y por tanto dé aún mejor resultado.

Dicho servidor PROXY se implementa en la propia aplicación en un fichero llamado proxy.conf.json:

```
{} proxy.conf.json > ...
1   {
2       "/cima/*": {
3           "target": "https://cima.aemps.es",
4           "secure": false,
5           "changeOrigin": true
6       }
7   }
```

Figura 35. Fichero 'proxy.conf.json'

Donde en el apartado *target* (objetivo) establecemos la *URL* del *backend* objeto de nuestra petición, en este caso GET. Desactivamos la seguridad mediante pasarle el valor *false* en el apartado *secure*. Por último, dado que nuestro *backend* no se está ejecutando localmente debemos cambiar el valor a *true* en el apartado *changeOrigin*. Se pueden configurar más parámetros como: *pathRewrite*: es el nombre que usaremos para invocar el servicio; *logLevel*: Para comprobar que todo funcione correctamente debemos indicar el valor en debug; *bypass*: Si se necesita omitir opcionalmente el proxy o bien cambiar la solicitud antes de mandarla, se debe definir la configuración en un archivo proxy.conf.js; pero en nuestro caso concreto no son necesarios, con los descritos anteriormente se ha llegado al resultado esperado.

Esquema del resultado final una vez creado el PROXY dentro de nuestro proyecto
:

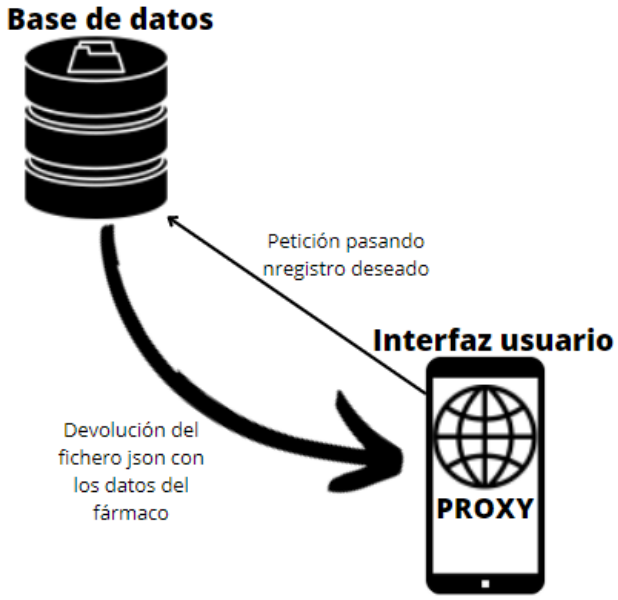

Figura 36. Esquema del BackEnd de la aplicación (2)(final)

3.2.2.3. Tercer paso

Como tercer paso en el desarrollo de esta aplicación se ha procedido a programar el escáner de códigos QR que permita acceder a la información del medicamento de una manera mucho más fácil al usuario final. Para esto se han tenido que instalar una serie de librerías de ionic que nos permitan acceder a la cámara del teléfono, así como leer el propio código bidi.

Para la instalación de la librería se introducen los siguientes comandos en el *cmd,* ubicados en la carpeta del proyecto:

```
ionic cordova plugin add cordova-plugin-qrscanner
npm install @ionic-native/qr-scanner
```

Figura 37. Comandos instalación librería QRScanner

Y se llama en la correspondiente vista donde queramos situar nuestro escáner mediante:

```
import {QRScanner, QRScannerStatus} from "@ionic-native/qr-scanner/ngx";
```

Figura 38. Import necesario para QRScanner

A continuación, habiendo instalado debidamente las librerías en el proyecto, se procede a crear una función que nos lea los parámetros:

```
startScanning() {
  this.qrScanner.prepare().
    then((status: QRScannerStatus) => {
      if (status.authorized) {
        this.qrScanner.show();
        this.scanSub = document.getElementsByTagName('body')[0].style.opacity = '0';
        debugger
        this.scanSub = this.qrScanner.scan()
          .subscribe((textFound: string) => {
            document.getElementsByTagName('body')[0].style.opacity = '1';
            this.qrScanner.hide();
            this.scanSub.unsubscribe();
            this.qrText = textFound;
            this.medicamento = textFound;
          }, (err) => {
            alert(JSON.stringify(err));
          });
      } else if (status.denied) {
      } else {
      }
    })
    .catch((e: any) => console.log('Error is', e));
}
```

Figura 39. Función encargada de leer códigos QR

A la cual se le pasan como parámetros los siguientes:

```
// Atributos para escaner QR:
scanSub: any;
qrText: string;
```

Figura 40. Parámetros de la función 'startScanning'

3.2.2.3.1. Dificultades en el QRScanner

Al comienzo de este paso, se optó por una versión con etiquetas NFC, pero dado que la librería para leer NFC en Ionic no se encontraba actualizada a la hora de la realización de este proyecto y no permitía su implementación, se desistió con esta tecnología. Además de otro punto en contra: el hecho de implementar un lector NFC conllevaría situar etiquetas NFC en los envases de medicamentos y esto supondría un incremento considerable en el coste de la fabricación de éstos, algo que para una primera versión no era viable, así como el hecho de que no supone una mejoría para el usuario de la App con respecto a leer un código QR. Sin embargo, a futuro se plantea una alternativa, también avanzada, pero algo más efectiva (redactada en el apartado 5.1. Proyección a futuro, 5.1.1. RFID de esta memoria).

3.2.2.4. Último paso

Una vez terminado todo el código de Ionic, se ha procedido a la generación de una apk para que este proyecto pueda ser leído en un smartphone. Como se mencionó anteriormente, Ionic es multiplataforma y acepta IOS y Android, a continuación se detalla su implementación en Android:

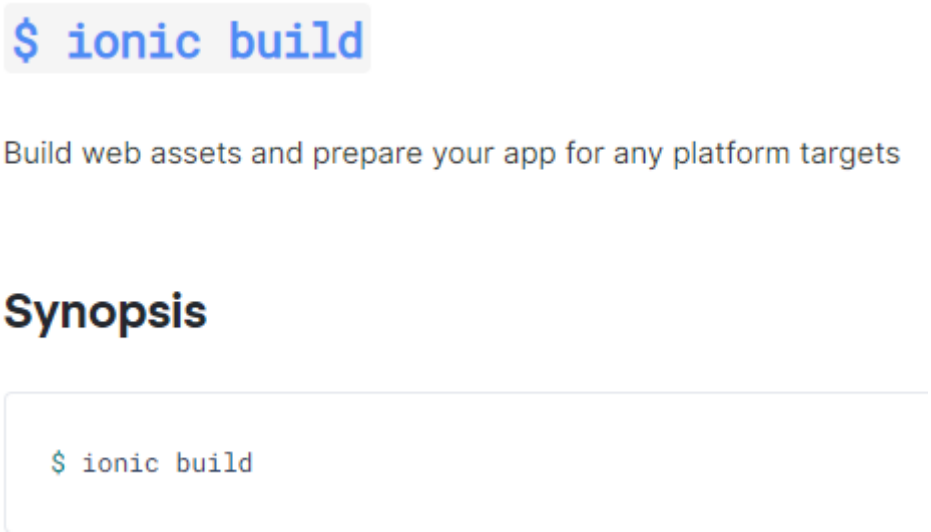

Figura 41. Comando 'ionic build'

Para comenzar, se ejecuta el comando *ionic build* para preparar todo nuestro proyecto. Este comando comprime todo nuestro proyecto en una carpeta llamada *www* la cual

tiene todo lo necesario para ser leída por Android Studio (instalación detallada en Anexo II) y por XCode.

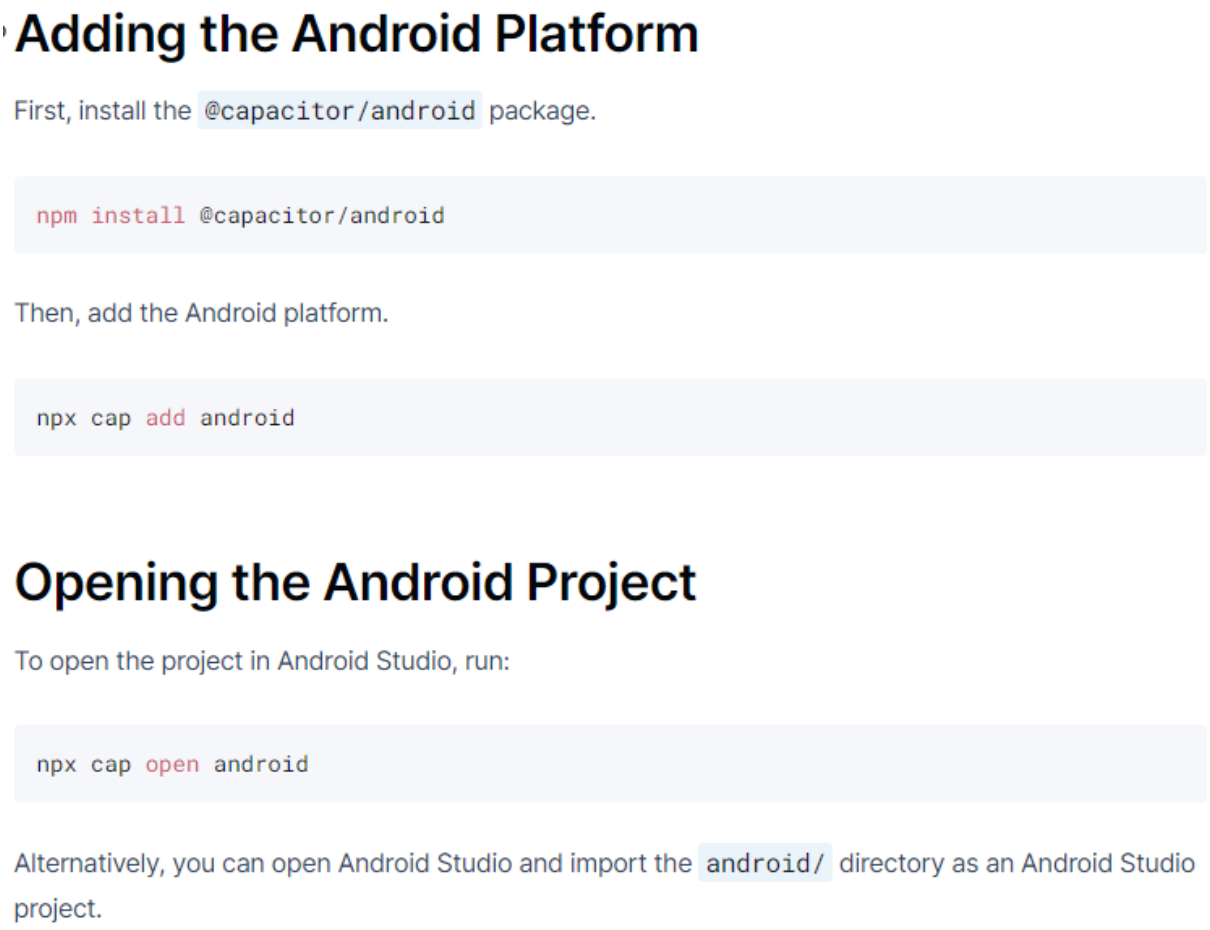

Figura 42. Paquete @Capacitor/android

A continuación, se procede a instalar el paquete *@capacitor/android* y a añadir *the Android platform* mediante el comando *npx cap add android*, este comando coge el contenido de la carpeta *www* creada anteriormente y lo adapta en otra nueva carpeta llamada *android*, la cual posteriormente lea Android Studio. para finalizar ejecutamos el comando *npx cap open android* para abrir la carpeta *android* en Android Studio.

Una vez creada la aplicación y ésta se encuentra funcionando en nuestro dispositivo móvil de prueba, se procede al *testing* o prueba de la misma para establecer sus debidas conclusiones y analizar si se han cumplido los objetivos planteados al comienzo del trabajo. Lo que deriva al Capítulo 4: Conclusiones y líneas futuras.

CAPÍTULO 4: CONCLUSIONES Y LÍNEAS FUTURAS

4.1. Conclusiones

En este TFG se han alcanzado los objetivos propuestos, desarrollando una App que permite al usuario consultar la información contenida en el prospecto del fármaco de una manera digital y actualizada. El acceso a esta información se puede llevar a cabo de manera manual: introduciendo el Código Nacional del medicamento, así como de una manera completamente directa con un lector de códigos QR debidamente integrado en la aplicación.

4.2. Proyección a futuro y mantenimiento

4.2.1. Actualizaciones

En cuanto a la línea futura planteada para tomar en esta aplicación se incluyen dos principales actualizaciones a medio y largo plazo que hagan de la primera versión desarrollada una versión completamente comercial y de éxito. Se detallan a continuación.

4.2.1.1. RFID

Como línea futura a este proyecto se plantea desarrollar un esquema de conectividad aún más avanzado y práctico para el usuario, basado en RFID (*Radio Frequency Identification* o Identificación por Radiofrecuencia en español).

Para llevar esto a cabo, se propone el desarrollo de un recipiente de X tamaño, que sirva de contenedor para los medicamentos, al cual se le dotaría de una antena capaz de leer etiquetas RFID, la cual haría la función de leer las etiquetas del contenido del contenedor (medicamentos). Esta antena se encarga de leer la información y la transmitirla a su vez a un servicio web montado previamente en la configuración, el cual, éste estaría conectado con nuestra aplicación en todo momento, mostrándonos así en cualquier instante el contenido del recipiente como su respectiva información.
Todo esto tendría gran utilidad en hogares con personas con enfermedades crónicas o recurrentes, y que por tanto se vean obligados a tener gran cantidad de medicamentos, ya que les facilitaría en gran medida la organización de éstos; teniendo al alcance de su mano la cantidad y/o estado del medicamento deseado.
A su vez, resultaría de gran ayuda en farmacias a la hora de llevar un inventario y control de todas sus provisiones. En una farmacia se plantearía algo más grande que un recipiente, como por ejemplo, un cajón o incluso un armario completo con antenas en distintas subsecciones.

A continuación se muestra un esquema generalizado de cómo sería el funcionamiento de todo lo descrito previamente:

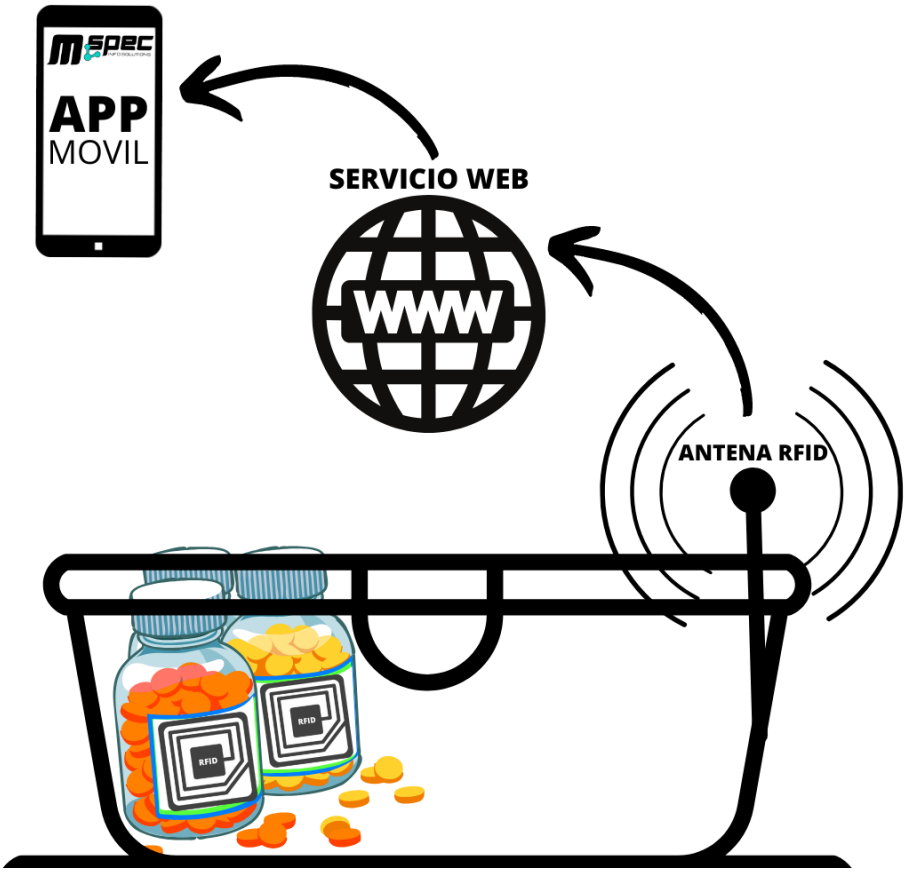

Figura 43. Proyección a futuro - RFID

4.2.1.2. Base de datos propia

Dado que por el momento no ha sido necesario crear una base de datos en la cual se contengan todos los datos de los medicamentos registrados en España, porque, como se comenta anteriormente, ya hay una base de datos existente y, aún más importante, actualizada por el ministerio; a la hora de mostrar por pantalla los datos nos conectamos a este almacenamiento.

Sin embargo, si se plantea un registro más personalizado al usuario, en el cual éste pueda llevar un historial y demás, entonces sí es necesario confeccionar una base de datos propia para la aplicación en la cual contener todo esto.

Es necesario distinguir dos perfiles de usuario en este proyecto a futuro:

- Usuarios particulares: éstos con su *login* podrían guardar un historial en un apartado de la aplicación los medicamentos que han escaneado, pudiendo así incluso volver a encargarlos y recogerlos en su farmacia más cercana. Incluso se contempla la posibilidad de tener acceso a todos sus datos médicos y el estado de sus recetas.

- Farmacias (usuarios profesionales): al *logearse* se les permitirá tener guardado y a su alcance continuamente el *stock* de fármacos que tengan en tienda.

Ambas posibles líneas futuras, tanto el desarrollo y testeo con la tecnología RFID, como la creación de una base de datos en la cual almacenar la información de cada usuario, guardan una estrecha relación.

4.2.2. Lanzamiento

Una vez se haya garantizado la calidad tanto de la primera versión, como de sus correspondientes actualizaciones mencionadas anteriormente, y siguiendo el orden de fases en el desarrollo de una aplicación, se procederá al lanzamiento de un producto final al mercado con las expectativas de ofrecer todo el valor esperado al usuario final.

4.2.3. Mantenimiento

Tras el paso anterior, el lanzamiento, como en toda app, hay que estar pendientes al cambio y no quedarse rezagados, por esto como línea futura es altamente importante la etapa de mantenimiento. Etapa necesaria en el futuro de cualquier aplicación para la corrección de posibles errores o problemas resultantes una vez puesto el producto final en el mercado. Para estar al tanto de esto se tratará de tener un continuo *feedback* con los usuarios de la aplicación, en especial con los farmacéuticos con el nicho de mercado principal de la App.

ANEXOS

Anexo I: Creación de un proyecto Ionic

En primer lugar, se procede a explicar el proceso de configuración para la creación de un proyecto Ionic:

Herramientas que necesitaremos:

- Framework NodeJS instalado: NodeJS es un framework de servidor en JS (v10.15.03 en el momento de esta publicación de blog, sin embargo, 12.x también es muy estable ahora.
- NPM : administrador de paquetes de nodo (v6.10.2 en el momento de esta publicación de blog)
- Ionic CLI : la herramienta de línea de comandos instalada en el sistema operativo para la aplicación (v6.10.2 en el momento de la publicación del blog)

También se necesitará el navegador más reciente para el entorno de desarrollo. Para crear compilaciones de producción en Android e iOS, se necesitarán Android Studio y XCode respectivamente.

Instalamos el nodo descargando el paquete compatible desde su sitio web . NPM se instala junto con node.js. NPM se utiliza para otras instalaciones.

Para verificar las instalaciones, ejecutamos estos comandos en nuestro terminal:

```
node --version
npm --version
```

E instalamos Ionic usando el siguiente comando:

```
npm install -g @ionic/cli
```

Las herramientas del SDK de Android y XCode requerirán instalaciones por separado

Puede instalar la última versión de Android Studio desde su sitio web. XCode solo se puede instalar en sistemas Apple.

Creación de un proyecto Ionic

Una vez que su entorno de desarrollo está configurado, comenzar a crear una aplicación Ionic 5 es tan fácil como ejecutar un comando:

```
ionic start
```

Ahora, después de ejecutar el comando anterior, tenemos dos opciones para elegir entre una es Angular y la otra es React.

Después de eso, pedirá el Nombre del proyecto , podemos dar el nombre que deseemos y después de eso, nos pedirá el diseño que deseamos elegir para su aplicación y nos dará tres opciones como *en blanco, menú lateral y pestañas.*

Después de eso, pedirá que integremos nuestra aplicación con Capacitor o no.

En lugar del proceso paso a paso anterior, también se puede escribir un comando para todo. Aquí hay algunos comandos para mostrar las posibles variaciones.

```
ionic start // simple interactive start command
ionic start --list //lists available starter templates
ionic start myApp //interactive start command with app
name
ionic start myApp blank //interactive start with name
and type
ionic start myApp tabs --cordova //start with angular
cordova
ionic start myApp tabs --capacitor //start with angular
capacitor
ionic start myApp super --type=ionic-angular //start
with angular
ionic start "My App" blank --type=react --capacitor
```

Eso es todo lo que se necesita para crear una aplicación Ionic. Presionamos *enter* e instalaremos la plantilla de aplicación básica directamente en nuestro sistema

Para ejecutar la aplicación en el navegador, escribimos en el terminal lo siguiente (estando en el directorio raíz de nuestro proyecto):

```
ionic serve
```

Y ya podemos ver nuestro proyecto Ionic funcionando en el puerto local de nuestro ordenador.

Anexo II: Instalación de Visual Studio Code y Android Studio

Para instalar VS Code solo tendremos que abrir una terminal (Ctrl+Alt+T) y ejecutar el siguiente comando:

```
sudo snap install --classic code
```

Para instalar Android Studio solo tendremos que abrir una terminal (Ctrl+Alt+T) y ejecutar el siguiente comando:

```
sudo apt install android-studio
```

Aunque ambos se pueden descargar igualmente desde sus respectivas páginas web:

- Visual Studio Code: https://code.visualstudio.com/download
- AndroidStudio: https://developer.android.com/studio?hl=es&gclid=Cj0KCQjwgYSTBhDKARIsAB8KuksEPGBk4r0uPEvog5S6eMeEw62BxWg6BGnW0nD4jOfpU3WKnXEfRHMaAg2dEALw_wcB&gclsrc=aw.ds

En la página viene explicado también paso a paso el proceso de configuración de la herramienta Android SDK.

Anexo III: Manual de usuario de la aplicación

MANUAL DE USO BÁSICO DE LA APLICACIÓN

Bienvenido/a al Manual de uso básico de la aplicación M-Spec, la herramienta encargada de facilitarle la información de todos los medicamentos al alcance de su mano, permitiéndole llevar un mejor control de éstos. Este concepto de aplicación surge en la Escuela Politécnica Superior de Linares como parte de INSIDE UJA y con la finalidad de digitalizar la información relativa a los fármacos del mercado, así como de facilitar el control de éstos a todos los usuarios de esta App. Las ventajas más destacadas que suponen su uso abarcan:

- Digitalización de la información
- Reducción en papel
- Acceso a la información en todo momento

M-Spec pone en marcha este servicio para ofrecer al usuario medio y profesional las mejores y más completas herramientas de gestión y acceso a la información de los medicamentos comerciales en España, sin coste alguno para ellos y con las mayores garantías de calidad, seguridad y servicio.

Contenido

1. ¿Cómo acceder a M-Spec? 57

1.1. ¿Cómo crear una cuenta? 57

2. Interfaz de la aplicación 58

2.1. Primera impresión al acceder 58

2.2. Principales funciones de la interfaz 60

2.2.1. Buscador 61

2.2.1. Buscador manual de medicamentos 63

2.2.1.1. ¿Dónde encontrar el Código Nacional de un medicamento? 64

2.2.2. Escáner de medicamentos 66

2.2.3. Base de datos del ministerio 67

1. ¿Cómo acceder a M-Spec?

Para acceder a esta aplicación solo tendremos que descargarla en nuestro dispositivo móvil e instalarla en este.

1.1. ¿Cómo crear una cuenta?

Para crear una cuenta en M-Spec nos dirigimos al icono “Iniciar” de la página principal:

Figura 44. Pantalla de Registro de M-Spec

Y dado que se trata aún de una versión de prueba, no es necesario realizar ningún registro introduciendo nuestros datos personales.

2. Interfaz de la aplicación

La interfaz que nos encontramos en M-Spec es muy intuitiva y sencilla, veámosla.

2.1. Primera impresión al acceder

Una vez dentro, lo primero que encontramos es la pantalla de “Inicio”, en la que se nos muestra el menú principal de la aplicación, el cual incluye las opciones de “Buscar”, “Listado” y “Estadísticas”:

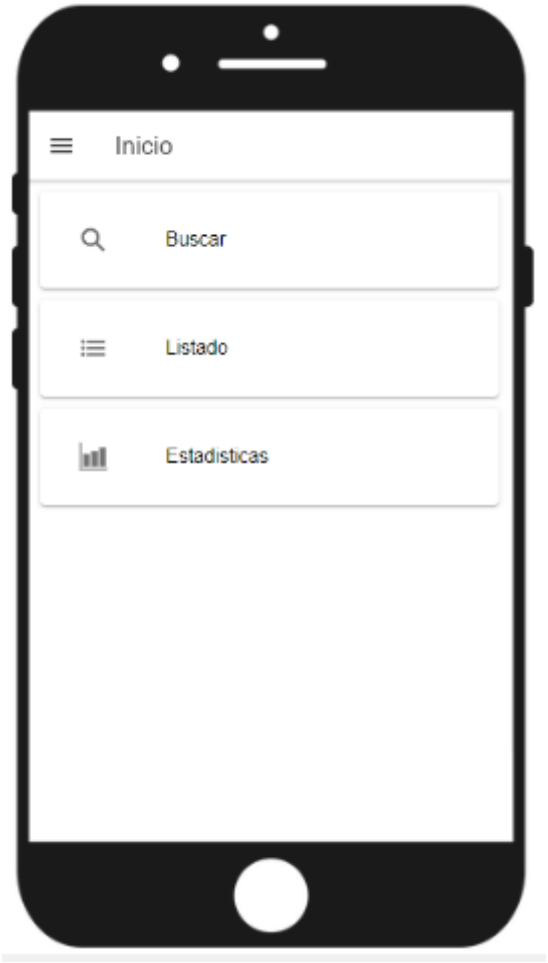

Figura 45. Interfaz de Inicio de M-Spec (1)

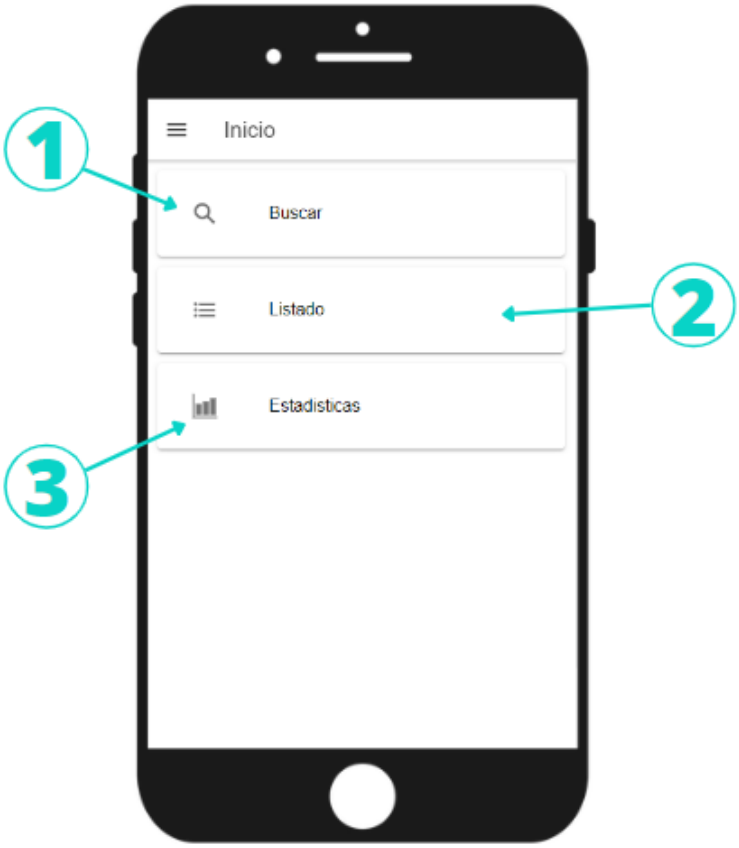

Figura 46. Interfaz de Inicio de M-Spec (2)

- Sección dedicada al buscador de medicamentos (explicada más adelante)
- En este bloque se encuentran los medicamentos que hemos escaneado y que podremos consultar su respectiva información.
- Aquí se muestra el índice de actividad llevada a cabo, para así tener un control más exhaustivo (previsto su desarrollo para usuarios *premium*).

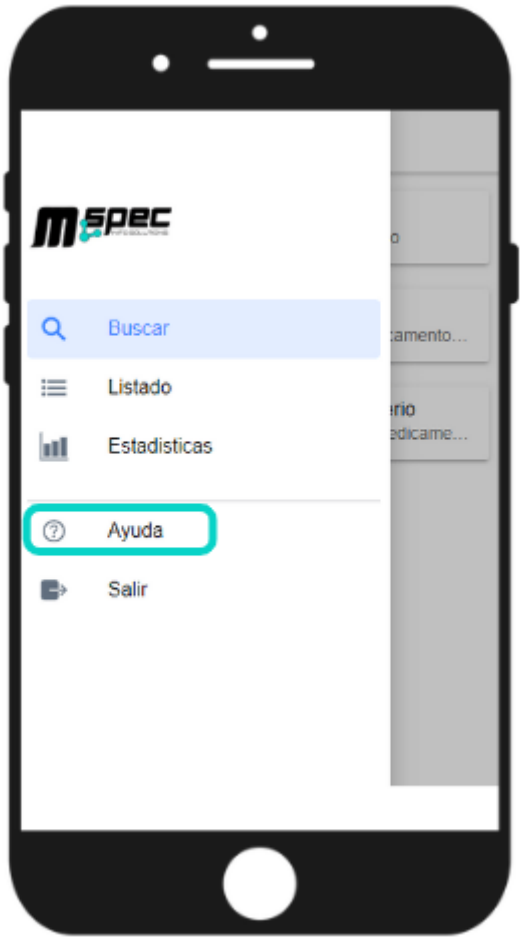

Figura 47. Menú lateral de M-Spec

Este mismo menú lo encontramos en todo momento en el lateral de nuestra aplicación, así como un sub-apartado de “ayuda” en el cual se encuentra este mismo manual de usuario.

2.2. Principales funciones de la interfaz

Dentro de cada sección descrita anteriormente, vamos a ver cada uno de sus componentes y su funcionalidad.

2.2.1. Buscador

Desde el panel de control del buscador vamos a poder seleccionar las opciones básicas de búsqueda de esta aplicación como pueden ser: el escáner, el buscador manual o el acceso a la base de datos del Ministerio.

Figura 48. Menú del Buscador de M-Spec (1)

Figura 49. Menú del Buscador de M-Spec (2)

1. Escanear medicamento: nos permite escanear el medicamento deseado, usando tecnología QR.
2. Introducir manualmente: accedemos a la información del medicamento mediante la introducción de su Código Nacional[1] en el buscador. En la propia app se encuentra en una pantalla de ayuda al usuario que especifica dónde se ubica este código en la caja del medicamento. (Figura 51)
3. Ver base de datos del ministerio: deriva a la base de datos del ministerio donde podemos consultar libremente cualquier medicamento registrado en España.

[1] Código Nacional: Los medicamentos se identifican y declaran mediante un número único dividido en tres segmentos que se conoce como el Código Nacional de Medicamentos (NDC, por sus siglas en inglés), el cual sirve como el identificador de la FDA para los fármacos.

2.2.1. Buscador manual de medicamentos

En esta pantalla, como podemos ver, añadimos manualmente el Código Nacional del medicamento a consultar, presionamos en “ESCANEAR” y obtenemos toda la información relevante de éste.

Figura 50. Buscador manual de M-Spec (resultado)

1. Espacio donde introducir el Código Nacional del medicamento.
2. Imagen del medicamento, si este dispone de ella en la base de datos.
3. Nombre del medicamento en cuestión
4. Laboratorio fabricante.
5. Prospecto digitalizado, mediante presionar en el icono señalado accedemos al prospecto completo.
6. Si el medicamento va con receta o no.

En el siguiente punto se aclara dónde podemos encontrar el Código Nacional de un medicamento registrado en España.

2.2.1.1. ¿Dónde encontrar el Código Nacional de un medicamento?

Se ha considerado de interés añadir en la propia App un botón de ayuda para indicar dónde se puede encontrar el Código Nacional del medicamento a buscar, se muestra dicho botón en la figura siguiente:

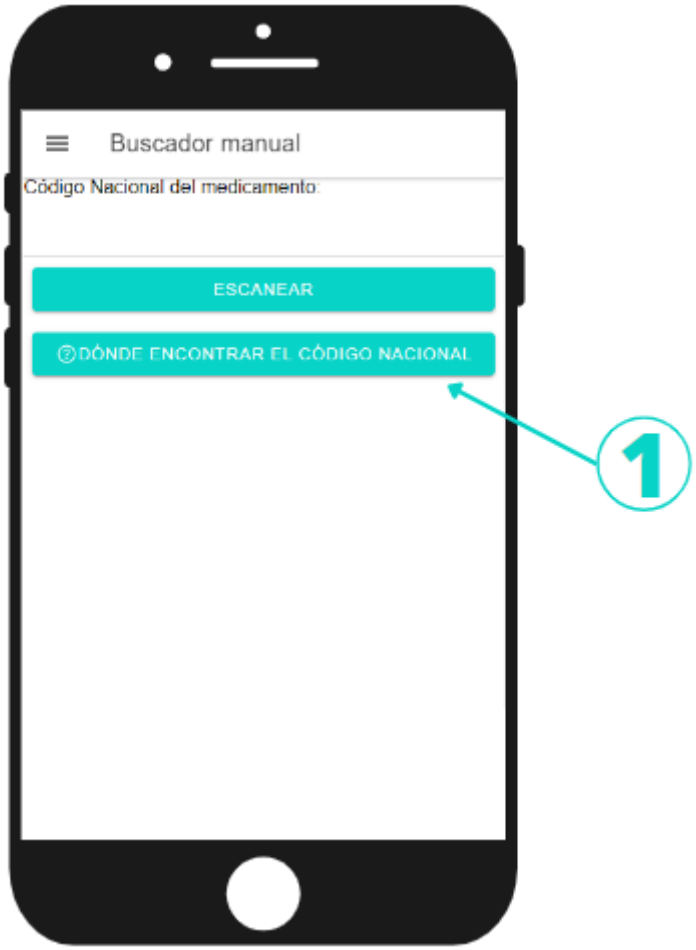

Figura 51. Ayuda al usuario para encontrar el Código Nacional

1. Botón que abre el desplegable de ayuda para indicar dónde encontrar situado el Código Nacional del medicamento.

El Código Nacional de un medicamento solemos encontrarlo en la parte superior derecha del envase del fármaco. (Figuras 52 y 53)

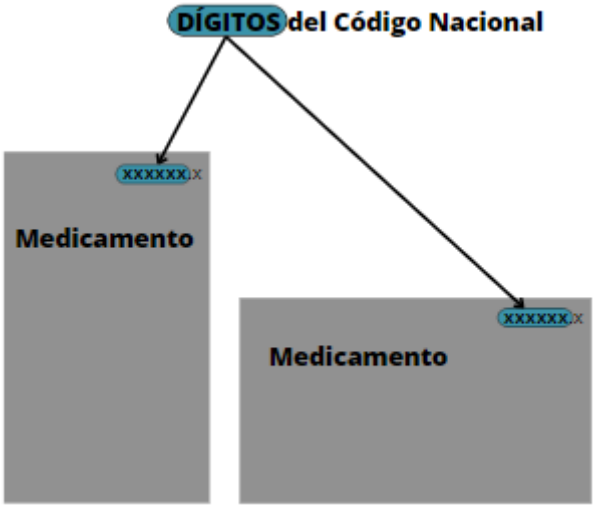

Figura 52. Ubicación del Código Nacional (1)

Figura 53. Ubicación del Código Nacional (2) (ejemplo)

2.2.2. Escáner de medicamentos

El escáner de medicamentos nos permite acceder a la información del medicamento de la manera más fácil y directa, solamente escaneando un código QR.

Figura 54. Escáner QR (1)

1. Código QR situado en el medicamento siendo escaneado por la aplicación

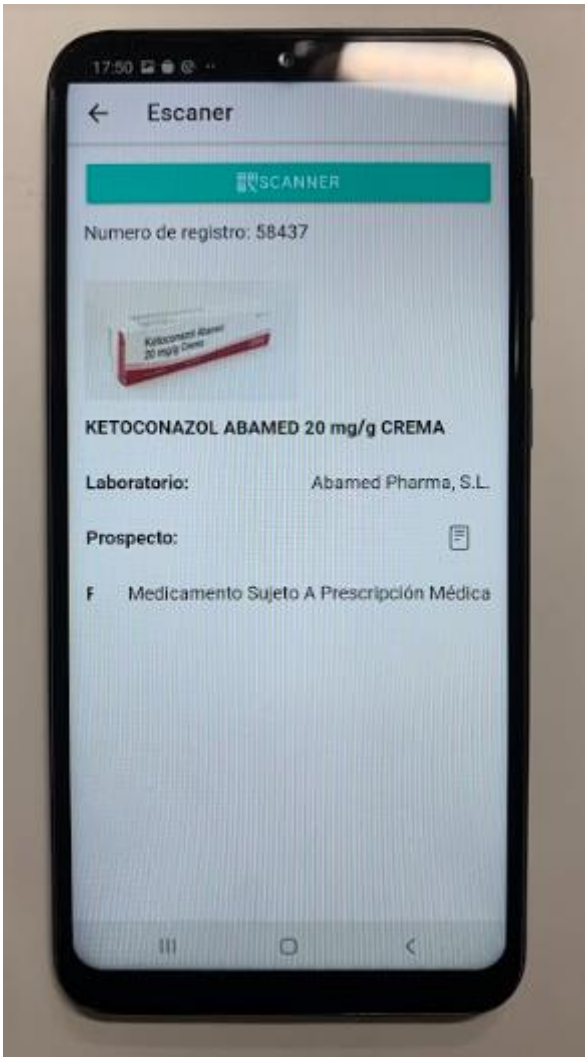

Figura 55. Escáner QR (2) (resultado)

Resultado obtenido tras haber escaneado el código QR, como observamos se trata de la misma información devuelta en el buscador manual. (apartado 2.2.1. Buscador manual de medicamentos, del manual de usuario)

2.2.3. Base de datos del ministerio

Por último, en este apartado encontraremos acceso directo a los buscadores más importantes de medicamentos de España:

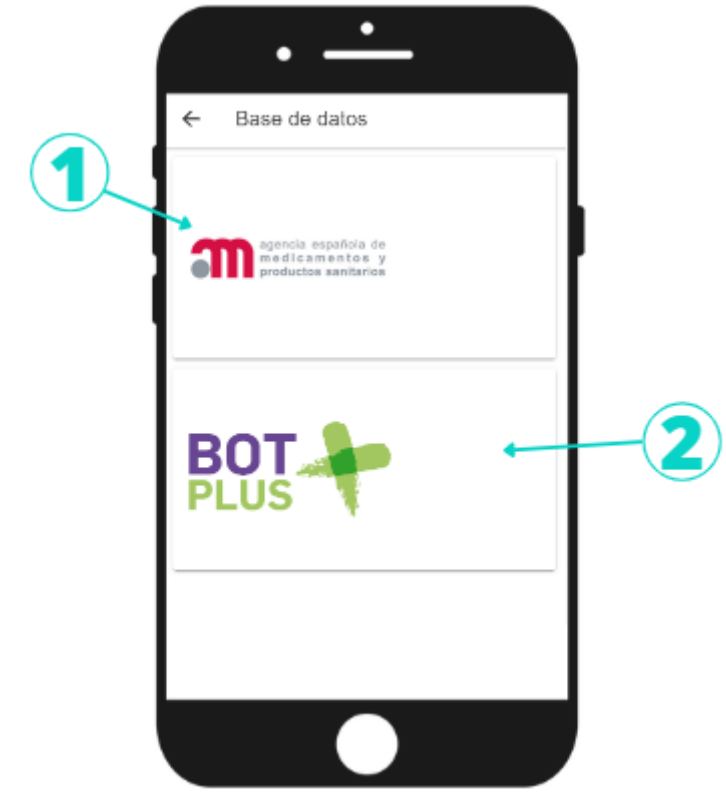

Figura 56. Buscadores nacionales de medicamentos (1)

1. CIMA: Centro de información online de medicamentos de la AEMPS - CIMA
2. BOT PLUS: Acceso para farmacéuticos. Se requiere clave, DNI / CIF y el número de la factura / pedido de compra de la Colección CONSEJO o de BOT PLUS web.

Figura 57. Buscadores nacionales de medicamentos (2)