Universitat Oberta de Catalunya

uoc.edu

Desarrollo de aplicación Android: EasyCook!

Antonio Cantos Pérez
Máster Universitario en Ingeniería Informática
Desarrollo de Aplicaciones sobre Dispositivos Móviles (M1.318)

Jordi Ceballos Villach
Robert Clarisó Viladrosa

08/01/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de aplicación Android: EasyCook!*
Nombre del autor:	*Antonio Cantos Pérez*
Nombre del consultor/a:	*Jordi Ceballos Villach*
Nombre del PRA:	*Robert Clarisó Viladrosa*
Fecha de entrega:	01/2020
Titulación::	*Máster Universitario en Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo de Aplicaciones sobre Dispositivos Móviles (M1.318)*
Idioma del trabajo:	*Español*
Palabras clave	*Android, Cocina, REST*

Resumen del Trabajo:

Debido a la revolución sufrida a principios de siglo con la invención de los llamados teléfonos inteligentes (Smarthpones), la sociedad actual ha sufrido cambios muy variados afectando a los diferentes planos de la sociedad. Este hecho unido a la escasez de tiempo, ha provocado la aparición de gran multitud de aplicaciones de comida rápida o a domicilio, siendo su principal nicho de mercado los jóvenes, muchos más familiarizados con los dispositivos comentados anteriormente.

Esta situación ha provocado que los hábitos alimentarios cambien, conllevando en muchos casos, unas rutinas poco saludables. Es por este motivo que nace EasyCook!, con el objetivo de acercar hábitos saludables y conocimientos culinarios al sector más joven de la población.

Para su correcto desarrollo, se propone un modelo en cascada con las fases de requisitos, diseño, implementación, verificación y mantenimiento, aunque con una modificación, durante la fase de diseño de la interfaz gráfica, se usará un modelo iterativo a fin de mejorar su adaptación a las fases de diseño centrado en el usuario.

En cuanto a la implementación, se ha optado por el desarrollo de una aplicación nativa de Android bajo Java, y Python para el servidor que provee de datos a la misma, mediante el uso de interfaces REST. Para ello, fue necesario el uso de librerías externas que facilitaron la integración del producto.

Finalmente, se obtuvo un producto usable, robusto y amigable, que cumplía con los objetivos del proyecto y, durante su desarrollo, aportó lecciones interesantes que aplicar en el futuro.

Abstract:

Due to the revolution suffered at the beginning of the century with the invention of the so-called smartphones, today's society has undergone very varied changes affecting the different levels of society. This fact coupled with the shortage of time, has led to the emergence of many applications of fast food or home delivery, being its main niche market young people, many more familiar with the devices discussed above.

This situation has caused eating habits to change, leading in many cases to unhealthy routines. It is for this reason that EasyCook! has born, with the aim of bringing healthy habits and culinary knowledge to the youngest sector of the population.

For its correct development, a cascade model is proposed with the requirements, design, implementation, verification and maintenance phases, although with an modification, during the design phase of the graphical interface, an iterative model will be used in order to improve its adaptation to user-centered design phases.

In terms of implementation, it has been decided to develop a native Android application under Java, and Python for the server that provides data to it, through the use of REST interfaces. For this, it was necessary to use external libraries that facilitated the integration of the product.

Finally, a usable, robust and friendly product was obtained that met the goals of the project and, during its development, provided interesting lessons to apply in the future.

Índice

1. Introducción

En este capítulo se comenzará detallando todos los aspectos introductorios del proyecto, que permitan al lector comprender la motivación y necesidad de este proyecto. A continuación, se presentarán los objetivos perseguidos y cómo se alcanzarán, finalizando con la planificación temporal, los productos obtenidos y un breve resumen de los capítulos posteriores.

1.1 Contexto y justificación del Trabajo

La revolución vivida a finales de la década del 2000, con la salida de los primeros modelos de los llamados **teléfonos inteligentes** (*Smartphones*, en adelante) ha provocado un **cambio drástico en diversos ámbitos**, siendo mucho más que una simple revolución tecnológica. Este fenómeno ha llegado a aspectos tan variados como las relaciones personales, el transporte e incluso, ha provocado la aparición de nuevas profesiones. Estos dispositivos, tal y como demuestran las estadísticas, **se han vuelto parte indispensable en la vida diaria** de una gran cantidad de personas: En España la tasa de penetración era del 88% en 2017 (Statista, 2019) y, como se observa en el siguiente gráfico, se ha convertido en el dispositivo más usado para el acceso a Internet por el 94.6% de los españoles (Ditrendia, 2017), cifras que aumentan en la población joven:

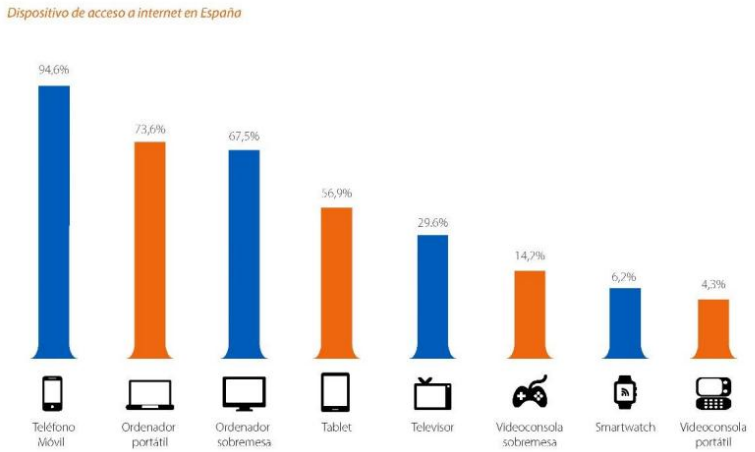


Ilustración 1. Dispositivo usado para acceso a Internet en España (Fuente: Ditrendia Mobile 2017)

El cambio comentado en el párrafo anterior, **ha afectado en gran medida a la forma en que nos alimentamos**, siendo cada vez más populares las aplicaciones de comida a domicilio. Este auge viene provocado por ciertos patrones: gran interacción del sector joven con los dispositivos móviles, escaso conocimiento sobre nutrición saludable y bajo nivel de conocimiento sobre cocina. Es aquí donde una **solución innovadora**, tecnológica, "amigable" y que aproveche estos patrones, proporcionará y cubrirá la necesidad de una alimentación equilibrada por parte de este sector de la sociedad, complementado al resto de sectores.

Revisando las alternativas posibles, tanto aplicaciones móviles como fuera del ámbito móvil, se encuentran las siguientes:

- Thermomix: Considerado el robot de cocina líder del mercado, intuitivo y fácil de usar gracias a recetas guiadas paso a paso cuidadas al detalle. Sin embargo, por contra, su elevado precio (alrededor de 1.200€) y el alto coste de

recetas (adquiridas por subscripción anual o en soporte físico), hacen que no esté dirigida al sector de interés comentado anteriormente.

- Recetarios Online: Existen multitud de webs dedicadas a las recetas online, donde éstas se muestran, mayoritariamente, de forma textual. Estas webs a menudo incluyen la posibilidad de compartir recetas propias, filtrar por tipo de comida (por ejemplo, italiana o española) o incluso escribir comentarios. En cambio, éstas son usadas escasamente por el sector estudiado pues a menudo, transmiten sensación de "vejez" o incomodidad, por su aspecto poco cuidado y/o trabajado.
- Aplicación "Recetas de Comida Casera": Aplicación visualmente poco trabajada y escasamente usable. Su funcionalidad se limita a un listado escaso de recetas sin posibilidad de búsquedas (ni simples ni complejas) ni de seguimiento paso a paso de la elaboración.
- Aplicación "Recetas fáciles de cocina y tu lista de la compra - Recetix": Aplicación de interfaz simple, ofrece gran variedad de recetas agrupadas en categorías como por ejemplo, por el producto principal que contienen. En cuanto a su funcionalidad, permite la búsqueda basada en dificultad y tiempo de elaboración. Sin embargo, no dispone del seguimiento paso a paso de la receta pues, extrae éstas de internet, abriendo la web de la que fue integrada para realizarlas.
- Aplicación "Recetas de cocina gratis - tu comunidad de cocina": Aplicación simple desde la perspectiva visual. Respecto a su funcionalidad, exige de registro para el seguimiento paso a paso de una receta y dado que éstas son aportadas por los propios usuarios de la aplicación, la calidad de las mismas dependerá en gran medida del "detallismo" con el que fue elaborada. En cuanto a las búsquedas, simplemente son permitidas a través del nombre del plato.
- Aplicación "Nestlé Cocina. Recetas y Menús": Desde el punto de vista visual, posee una interfaz bastante cuidada y elaborada lo que la hace cómoda e intuitiva. Pasando al plano funcional, carece de la funcionalidad de seguir paso a paso una receta (y por tanto, de usar comandos de voz) así como, de realizar búsquedas por aporte calórico de las mismas. Finalmente, un punto reseñable reside en el patrocinio de "Nestlé", lo que hace que la mayoría de recetas contentan y se enuncien con productos de la marca, pudiendo alejar a ciertos usuarios que no deseen usarla.
- Aplicación "Tasty": Con una interfaz atractiva aunque algo simple, dispone de una gran funcionalidad como búsqueda mediante filtros de dificultad o tipo de comida. Una de sus capacidades principales reside en la posibilidad de ocultar comidas según tu elección alimentaria (como aquellas con carne si se es vegetariano). Aunque dispone de la posibilidad de realizar la receta paso a paso con contenido multimedia muy expresivo, carece de la posibilidad de usar la voz para pasar o retroceder los pasos.
- Aplicación "My Cookbook (Mis Recetas)": Aunque por el nombre podría pertenecer al mismo grupo que la aplicación aquí planteada, su planteamiento es más como un "recetario tecnológico". Por defecto, carece de cualquier receta, esperando que el usuario añada las que considere, permitiendo su importación desde Internet. Respecto a su interfaz gráfica, está bastante cuidada y trabajada siendo cómoda tanto en el plano visual como en la usabilidad.
- Aplicación "Recetas Kiwilimón": El punto fuerte de esta aplicación reside en su estética, con un diseño tanto funcional como artístico bastante atractivo. Respecto a la funcionalidad, aunque dispone de gran cantidad de recetas categorizadas por temporalidad (verano, *halloween*, etc.) no dispone de un filtro por aportación alimentaria o duración.

- Aplicación "Libro de cocina: RecetteTek": Aunque atractiva desde el punto de vista esquemático, su interfaz monocolor la hace algo cansada a la vista así como, des-focaliza el nicho de mercado del proyecto en este documento detallado. Desde el punto de vista funcional, aunque dispone de muchas opciones interesantes (como la importación, exportación, etc.) carece de base de datos propia de recetas y su funcionamiento se basa en la extracción de Internet, lo que suele conllevar fallos.

Una vez realizada la prospección anterior, parece evidente que aunque se dispone actualmente de una oferta relativamente amplia, **ninguna termina de lograr los objetivos marcados** en este proyecto. Igualmente, en el plano funcional, aunque ciertas opciones logran un grado alto de operatividad, tampoco consiguen cubrir el espectro marcado. Por tanto, la aplicación aquí desarrollada, logrará integrar características tecnológicas atractivas para el nicho indicado anteriormente, las cuales serán diferenciales a la hora de atraer el interés de los usuarios y conseguir una cuota de mercado considerable, aportando viabilidad y mejora continua a este proyecto.

1.2 Objetivos del Trabajo

En este apartado se definirán los objetivos perseguidos con la elaboración de este trabajo, los requisitos tanto funcionales como no funcionales a través de los cuales se cumplirán así como, los productos entregados al final del mismo.

El **objetivo general** perseguido por este trabajo consistirá en elaborar una aplicación software para dispositivos móviles Android innovadora, atractiva y con posibilidades futuras de mejora, a través de la cual mejorar los hábitos alimenticios del sector más joven de la sociedad, consiguiendo desmitificar la dificultad de la cocina. De forma más concreta, se persiguen los siguientes objetivos:

- OBJ001 - Poner en práctica todo lo aprendido en el Máster de Ingeniería Informática.
- OBJ002 - Ser un proyecto basado en la realidad.
- OBJ003 - Aportar la mayor sencillez de uso posible.
- OBJ004 - Acercar la cocina a las nuevas generaciones.
- OBJ005 - Posibilitar futuros cambios y/o mejoras, como la monetización, el multi-lenguaje o la adaptación a una mayor gama de dispositivos.
- OBJ006 - Permitir la escalabilidad según el número de usuarios.
- OBJ007 - Inculcar hábitos saludables a la población.
- OBJ008 - Copiar y mejorar los competidores actuales.

Para cumplir con estos objetivos marcados, se han concretado los siguientes requisitos funcionales:

- RF001 - La aplicación contará con al menos 10 recetas.
- RF002 - Las recetas se han de poder seguir paso a paso.
- RF003 - Cada receta mostrará al inicio un listado de ingredientes y/o utensilios necesarios.
- RF004 - En caso de actividades de larga duración, se establecerá un temporizador para avisar al usuario una vez finalice.
- RF005 - Si fuera posible, se podrá seguir con la receta mientras se realiza un paso de larga duración (paralelización de tareas).
- RF006 - Se podrá avanzar/retroceder mediante comandos de voz.
- RF007 - Se mostrará el aporte nutricional de la receta (calorías, grasas, etc.).
- RF008 - Envío de estadísticas al servidor central para posibles analíticas de negocio.

- RF009 - Búsqueda mediante filtro de aporte calórico.
- RF010 - Búsqueda mediante filtro de duración de la receta.
- RF011 - Búsqueda mediante filtro de dificultad de la receta.
- RF012 - Búsqueda mediante filtro de tipo de comida (fría, caliente, vegetariana, vegana, etc.).
- RF013 - Se permitirá la gestión de usuarios (*login*, *logoff* y *sing up*).
- RF014 - Uso de imágenes explicativas allá donde sea posible.
- RF015 - Se optará por únicamente el lenguaje español, aunque se preparará su posible traducción posterior.

Seguidamente, se establecen los siguientes requisitos no funcionales, más cercanos al "cómo" de la aplicación:

- RNF001 - La aplicación estará desarrollada en Android Nativo (Java SDK).
- RNF002 - La aplicación abarcará la mayoría de mercado Android mediante la compatibilidad con un alto porcentaje (>50%) de las *APIs* actuales.
- RNF003 - La aplicación estará preparada para dispositivos *Smartphone*, teniendo en cuenta su posible adaptación a *tablets* de forma posterior.
- RNF004 - La aplicación constará de un *Back-End* simple y un *Front-End.*
- RNF005 - La seguridad quedará fuera de alcance en el *Back-End*, aunque se tratará de seguir buenas prácticas generales.
- RNF006 - Se buscará la reutilización de código mediante *frameworks*.
- RNF007 - Se potenciará el uso de tecnologías gratuitas y abiertas.
- RNF008 - Se seguirán las buenas prácticas de programación general.
- RNF009 - Se seguirán las buenas prácticas aplicadas específicamente a programación móvil (Android).
- RNF010 - *Debugging* realizado sobre emuladores (*AVDs*), aunque basado en un terminal de 5,6 pulgadas.
- RNF011 - Prueba final sobre dispositivo real.
- RNF012 - Se desarrollará/n algún/os test/s de usabilidad para medir el grado de usabilidad y *User Experience* de la aplicación
- RNF013 - Se tratará de ofrecer la mejor experiencia de usuario (UX) posible, siempre centrada en el nicho de población objetivo, sin dejar de lado el resto de población.
- RNF014 - Las comunicaciones entre *Back-End* y *Front-End* se realizarán siguiendo un modelo *API REST*.
- RNF015 - Interfaz atractiva visualmente.
- RNF016 - Uso de test automatizados de pruebas y validación.
- RNF017 - Se tratará y codificará como si se preparara un lanzamiento comercial posterior.
- RNF018 - El sistema será robusto y fiable.

Para finalizar este apartado, se presenta una matriz de relación entre los objetivos propuestos para el proyecto y los requisitos funcionales establecidos. Con este elemento gráfico se pretende mostrar cómo, a través del cumplimiento de dichos requisitos, se será capaz cumplir y validar los objetivos planteados al inicio del proyecto, aportando una relación directa y visual entre ambos:

Requisitos	OBJ001	OBJ002	OBJ003	OBJ004	OBJ005	OBJ006	OBJ007	OBJ008
RF001			X					
RF002			X					
RF003		X	X					X
RF004	X	X	X					X
RF005			X					X
RF006	X			X				X
RF007		X					X	X
RF008	X				X	X		
RF009							X	
RF010		X		X				X
RF011		X		X				X
RF012		X		X				X
RF013			X					X
RF014	X		X	X				X
RF015					X			

Ilustración 2. Relación Objetivos-Requisitos Funcionales (Elaboración propia)

1.3 Enfoque y método seguido

Como se enunció en el capítulo dedicado al contexto y la justificación de la presente memoria, existen en la actualidad un amplio abanico de aplicaciones relacionadas con la cocina y la alimentación. Pese a este hecho, ninguna logra cumplir los objetivos planteados para este proyecto pues, aunque en ocasiones se cubren parcialmente en algunas de las alternativas, contienen elementos contrarios a otros. Este hecho podría llevar a pensar en la reutilización de una de estas aplicaciones como base para la generación de la aquí expuesta, sin embargo, su carácter de código cerrado imposibilita este hecho.

Así, por lo expuesto en el párrafo anterior, **se ha decidido realizar la aplicación partiendo de cero**, aunque siempre, tomando ideas de las ya existentes en el mercado y ampliándolas con características tecnológicas tan avanzadas como el uso de interfaz vocal o el filtrado selectivo, dándole el **carácter diferencial** que atraiga usuarios de éstas e incluso, nuevos.

Para este proyecto, aunque actualmente en el mercado móvil son usadas mayoritariamente tecnologías de desarrollo software *agile*, se ha preferido usar un modelo más tradicional, **el desarrollo en cascada** (aunque como se detallará más adelante, se han añadido unas modificaciones). Los **motivos principales** que han llevado a esta elección son:

- Los requisitos y objetivos son conocidos con anterioridad, permitiendo este modelo y "anulando" una de las grandes ventajas de las metodologías ágiles.
- El "equipo de trabajo" está compuesto por un único individuo, lo que provocaría que muchas de las ceremonias presentes en estas metodologías, carecieran de sentido perdiendo su aporte.
- Posee un modelo mucho más simple, lo que evitará una carga añadida de gestión en el proyecto.
- Al ser una metodología ampliamente extendida y usada, está bastante trabajada y optimizada.
- Sus fases, coinciden con las entregas parciales propuestas en la asignatura, facilitando su integración en la planificación.

Entrando en detalle, la metodología a usar, puede verse en la imagen siguiente:

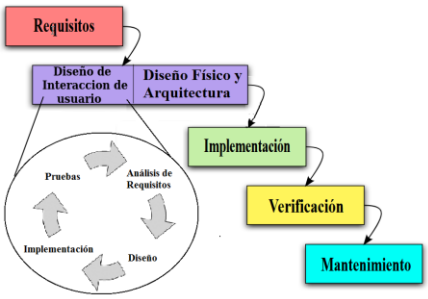


Ilustración 3. Metodología de Proyecto Usada (Elaboración Propia)

Como se observa, las fases principales de la metodología están presentes, aunque existiendo una pequeña modificación en el apartado de diseño:

- Requisitos: Durante esta fase se establecen y esclarecen todos los requisitos tanto funcionales, como no funcionales que tendrá la aplicación software.
- Diseño: En este punto se elaborarán todos aquellos aspectos relativos al diseño, como son la arquitectura físicay lógica o la interfaz gráfica. Respecto a este último elemento, para asegurar el cumplimiento de los principios básicos de la metodología de **diseño centrado en el usuario, se realizará a través de un modelo de desarrollo iterativo** donde sus fases serán: Análisis de requisitos, diseño de la solución, implementación y evaluación.
- Implementación: En esta etapa se desarrollará la aplicación en base al diseño establecido en el punto anterior.
- Verificación: Periodo durante el cual se probará que el producto (aplicación) elaborado en la fase anterior cumple con el funcionamiento esperado. Para ello, se elaborarará un plan de pruebas que establezca aquellos puntos clave a comprobar así como, la metodología de verificación a llevar a cabo.
- Mantenimiento: Durante esta fase se realizará la entrega del producto al cliente, asegurando su aceptación.

1.4 Planificación del Trabajo

En este apartado se definen aquellos aspectos más cercanos a la gestión del proyecto. Así, se define la planificación del proyecto (lo que facilitará un mayor control sobre el avance del mismo y su estado), el equipo de proyecto (diferenciando los roles de cada los integrantes), el calendario laboral y los recursos disponibles.

En primer lugar, respecto al equipo de proyecto, está compuesto por tres integrantes:

- El autor y ejecutor de este trabajo, Antonio Cantos Pérez, alumno del Máster Universitario en Ingeniería Informática.
- El profesor consultor del área de Desarrollo de Aplicaciones sobre Dispositivos Móvil, Jordi Ceballos Villach.
- El profesor consultor, más cercano al apartado de usabilidad y diseño centrado en el usuario, Jordi Almirall López.

Respecto a los recursos físicos disponibles, se dispone de:

- Equipo portátil Asus, con chipset Intel Core i7 (Ivy Bridge), 16 gigabytes de memoria RAM y 500 gigabytes de disco duro de estado sólido (SSD). El sistema operativo usado será Ubuntu 18.04, aunque en determinadas ocasiones, se acudirá a máquinas virtuales con sistema operativo Windows 7.
- Smartphone Samsung Galaxy J6 2018 con procesador de 8 núcleos, 3 gigabytes de memoria RAM, 32 gigabytes de memoria ROM y pantalla de 5,6 pulgadas. En el apartado software, cuenta con Android 9 (Pie) con capa de personalización de fabricante One UI.

Igualmente, en el plano software se dispone de:

- Paquete Office: Del cual se usará el editor de textos Word, con el fin de elaborar la presente memoria, así como el editor de presentaciones Powerpoint, para la elaboración de la presentación final.
- ProjectLibre: Software para la elaboración de planificaciones de proyecto usado para el diagrama Gantt que se mostrará posteriormente.
- Android Studio: Entorno de desarrollo integrado (IDE) usado para la codificación de la aplicación Android.
- Atom: Editor de textos. Será usado para la generación del código relativo al B*ack-End* de la aplicación.
- Git: Software de control de versiones. Su uso tendrá por objetivo asegurar la posibilidad de marcha atrás en caso de detección de errores insalvables o la generación de las distintas versiones de la implementación.
- Camtasia: Software de edición de video. Con él, se elaborará el video presentación final de este proyecto.

El plan de trabajo propuesto, debido a la innegable orientación académica de este proyecto, se ha generado en base al modelo de evaluación continua propuesto por la universidad. Así, se han establecido las siguientes cinco fases:

- F001-Elaboración de Propuesta
    - TAR001-Elección de idea y perfeccionamiento
    - TAR002-Definición del alcance del proyecto
    - TAR003-Elaboración de la Propuesta

  En esta fase se lanza el proyecto estableciendo y fijando los límites del mismo. Finalizará con el hito "H001-Entrega de Propuesta de Proyecto".

- F002-Elaboración de Plan de Trabajo
    - TAR004-Contexto y Justificación
    - TAR005-Objetivos
    - TAR006-Enfoque y Método Elegido
    - TAR007-Planificación
    - TAR008-Resumen Productos Obtenidos
    - TAR009-Descripción Resto de Capítulos de la Memoria
    - TAR010-Creación de Aplicaciones Base
    - TAR011-Refinamiento

  En esta fase se elabora toda la documentación inicial del proyecto la cual asegure el buen devenir del mismo. Terminará con el hito "H002-Entrega de Prueba 1 de Evaluación Continua".

- F003-Diseño y Arquitectura
    - TAR012-Diseño Centrado en usuario
        - TAR012.1-Contexto de Uso

        - TAR012.2-Indagación y perfilado de usuarios
        - TAR012.3-Flujos de interacción
        - TAR012.4-Prototipado
        - TAR012.5-Evaluación
    - TAR013-Generación de casos de uso
    - TAR014-Diseño de la arquitectura de la solución
    - TAR015-Documentación
    - TAR016-Refinamiento

  En este punto se diseñará al completo la solución asegurando el cumplimiento de los requisitos establecidos en la fase anterior. Finalizará con el hito "H003-Entrega de Prueba 2 de Evaluación Continua".

- F004-Implementación
    - TAR017-Elección y justificación tecnológica del *Front-End* (aplicación Android)
    - TAR018-Codificación del *Front-End* (aplicación Android)
    - TAR019-Elección y justificación tecnológica del *Back-End* (*Script* Python y BBDD)
    - TAR020-Codificación del *Back-End* (Script Python y BBDD)
    - TAR021-Definición del plan de pruebas del *Front-End* (aplicación Android)
    - TAR022-Ejecución de pruebas del *Front-End* (aplicación Android)
    - TAR023-Definición del plan de pruebas del *Back-End* (*Script* Python y BBDD)
    - TAR024-Ejecución de pruebas del *Back-End* (*Script* Python y BBDD)
    - TAR025-Documentación
    - TAR026-Refinamiento

  Llegados a esta fase, se desarrollarán todos los componentes diseñados en el apartado anterior, de forma que se asegure el funcionamiento de la misma. Terminará con el hito "H004-Entrega de Prueba 3 de Evaluación Continua".

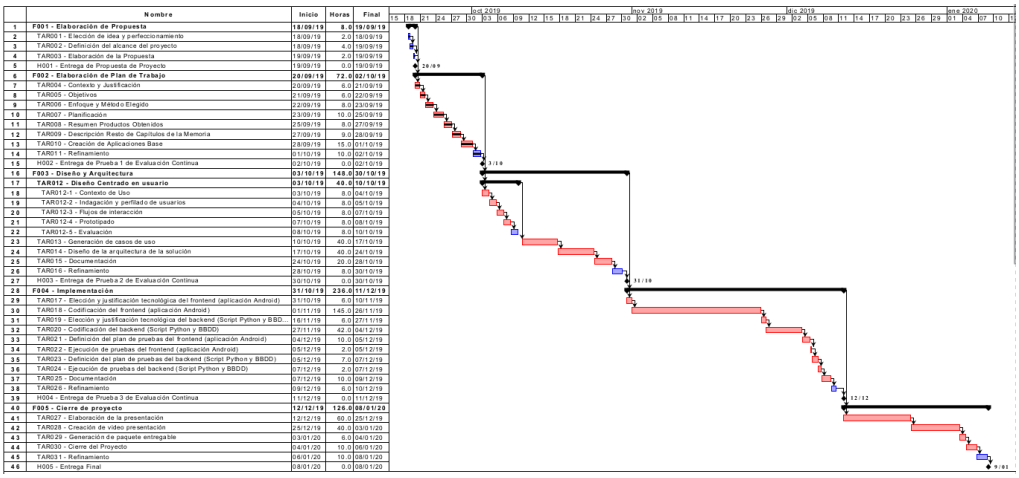
- F005-Cierre de proyecto
    - TAR027-Elaboración de la presentación
    - TAR028-Creación de video presentación
    - TAR029-Generación de paquete entregable
    - TAR030-Cierre del Proyecto
    - TAR031-Refinamiento

  Para finalizar el proyecto, se generarán aquellos elementos adicionales del proyecto con el fin de que el resultado final sea puesto en valor por el cliente y se muestre el trabajo realizado. Finalizará con el hito "H005-Entrega Final".

Seguidamente, en cuanto a la planificación temporal, en primera instancia será necesario conocer el **calendario laboral** a fin de establecer posteriormente la temporalidad correcta para cada tarea. Así, tras una breve reflexión se estableció lo siguiente:

- De lunes a viernes, se dispondrá de una media de 4 horas.
- En sábado, domingo y festivos el tiempo disponible será de 10 horas.
- Un fin de semana al mes, debido a traslado, el tiempo total disponible será de 15 horas.

Tras estas aproximaciones y teniendo en cuenta que la **fecha de inicio** del proyecto es dieciocho de septiembre de dos mil diecinueve y la **fecha de finalización** es ocho de enero de dos mil veinte, se elaboró el siguiente calendario de proyecto el cual se puede observar en formato de diagrama Gantt:



	Nombre	Inicio	Horas	Final
1	**F001 - Elaboración de Propuesta**	**18/09/19**	**8.0**	**19/09/19**
2	TAR001 - Elección de idea y perfeccionamiento	18/09/19	2.0	18/09/19
3	TAR002 - Definición del alcance del proyecto	18/09/19	4.0	18/09/19
4	TAR003 - Elaboración de la Propuesta	19/09/19	2.0	19/09/19
5	H001 - Entrega de Propuesta de Proyecto	19/09/19	0.0	19/09/19
6	**F002 - Elaboración de Plan de Trabajo**	**20/09/19**	**72.0**	**02/10/19**
7	TAR004 - Contexto y Justificación	20/09/19	6.0	21/09/19
8	TAR005 - Objetivos	21/09/19	6.0	22/09/19
9	TAR006 - Enfoque y Método Elegido	22/09/19	8.0	23/09/19
10	TAR007 - Planificación	23/09/19	10.0	25/09/19
11	TAR008 - Resumen Productos Obtenidos	25/09/19	8.0	27/09/19
12	TAR009 - Descripción Resto de Capítulos de la Memoria	27/09/19	9.0	28/09/19
13	TAR010 - Creación de Aplicaciones Base	28/09/19	15.0	01/10/19
14	TAR011 - Refinamiento	01/10/19	10.0	02/10/19
15	H002 - Entrega de Prueba 1 de Evaluación Continua	02/10/19	0.0	02/10/19
16	**F003 - Diseño y Arquitectura**	**03/10/19**	**148.0**	**30/10/19**
17	**TAR012 - Diseño Centrado en usuario**	**03/10/19**	**40.0**	**10/10/19**
18	TAR012-1 - Contexto de Uso	03/10/19	8.0	04/10/19
19	TAR012-2 - Indagación y perfilado de usuarios	04/10/19	8.0	05/10/19
20	TAR012-3 - Flujos de interacción	05/10/19	8.0	07/10/19
21	TAR012-4 - Prototipado	07/10/19	8.0	08/10/19
22	TAR012-5 - Evaluación	08/10/19	8.0	10/10/19
23	TAR013 - Generación de casos de uso	10/10/19	40.0	17/10/19
24	TAR014 - Diseño de la arquitectura de la solución	17/10/19	40.0	24/10/19
25	TAR015 - Documentación	24/10/19	20.0	28/10/19
26	TAR016 - Refinamiento	28/10/19	8.0	30/10/19
27	H003 - Entrega de Prueba 2 de Evaluación Continua	30/10/19	0.0	30/10/19
28	**F004 - Implementación**	**31/10/19**	**236.0**	**11/12/19**
29	TAR017 - Elección y justificación tecnológica del frontend (aplicación Android)	31/10/19	6.0	10/11/19
30	TAR018 - Codificación del frontend (aplicación Android)	01/11/19	145.0	26/11/19
31	TAR019 - Elección y justificación tecnológica del backend (Script Python y BBD...	16/11/19	6.0	27/11/19
32	TAR020 - Codificación del backend (Script Python y BBDD)	27/11/19	42.0	04/12/19
33	TAR021 - Definición del plan de pruebas del frontend (aplicación Android)	04/12/19	10.0	05/12/19
34	TAR022 - Ejecución de pruebas del frontend (aplicación Android)	05/12/19	2.0	05/12/19
35	TAR023 - Definición del plan de pruebas del backend (Script Python y BBDD)	05/12/19	7.0	07/12/19
36	TAR024 - Ejecución de pruebas del backend (Script Python y BBDD)	07/12/19	2.0	07/12/19
37	TAR025 - Documentación	07/12/19	10.0	09/12/19
38	TAR026 - Refinamiento	09/12/19	6.0	10/12/19
39	H004 - Entrega de Prueba 3 de Evaluación Continua	11/12/19	0.0	11/12/19
40	**F005 - Cierre de proyecto**	**12/12/19**	**126.0**	**08/01/20**
41	TAR027 - Elaboración de la presentación	12/12/19	60.0	25/12/19
42	TAR028 - Creación de video presentación	25/12/19	40.0	03/01/20
43	TAR029 - Generación de paquete entregable	03/01/20	6.0	04/01/20
44	TAR030 - Cierre del Proyecto	04/01/20	10.0	08/01/20
45	TAR031 - Refinamiento	08/01/20	10.0	08/01/20
46	H005 - Entrega Final	08/01/20	0.0	08/01/20

Ilustración 4. Diagrama Gantt del Proyecto (Elaboración Propia)

Como se muestra, el **tiempo de trabajo total** ascenderá aproximadamente a 590 horas durante un total de 113 días.

Finalmente, dado que como se ha mencionado anteriormente, el trabajo fin de máster es un proyecto más, existen riesgos que de no ser anticipados, pueden incidir negativamente en la planificación temporal del mismo, afectando a las entregas parciales y, en el peor de los casos, provocar el fracaso del mismo. Por este motivo, se cree interesante la adición de un análisis de riesgos, detallando aquellos detectados, además de las acciones mitigadoras sobre los mismos:

Riesgo	Descripción	Proba-bilidad	Impacto	Tipo	Acciones Mitigadoras
RG001-Traslados de domicilio temporales (viajes)	Debido a residir fuera del domicilio familiar, mensualmente, de forma planificada y durante un fin de semana se acudirá a éste, lo que reducirá el tiempo dedicado al proyecto.	Alta	Bajo	Riesgo menor	Dedicar un esfuerzo extra los días anteriores a fin de recuperar ese déficit de horas.
RG002-Enfermedad del autor	Dado que el proyecto se desarrollará durante los meses de invierno, el autor podrá caer enfermo.	Media	Bajo-Medio	Riesgo menor	- Evitar las principales casuísticas de enfermedad (por ejemplo, el frío). - Establecer medidas preventivas (vacunas) a las enfermedades comunes.
RG003-Falta de conocimiento técnico	Debido al uso de un abanico considerable de tecnologías, se podría detectar un déficit en el conocimiento necesario para su manejo.	Media	Medio	Riesgo significativo	- Búsqueda de bibliografía y recursos de calidad que ayuden a adoptar dicha tecnología. - Acceso a terceros que puedan prestar la ayuda/consejo requerido.
RG004-Error insalvable de diseño del producto	Fallo a la hora de establecer las lineas maestras de diseño de la aplicación que provoque un error con una afectación severa al funcionamiento de la misma.	Media	Alto	Riesgo Mayor	- Dedicación del tiempo necesario a establecer el diseño de la solución. - Uso de prototipos de

					alta fidelidad. - Búsqueda de información y/o consejo.
RG005-Pérdida accidental de información	Pérdida parcial o total de recursos elaborados durante el proyecto como podría ser la documentación o el código.	Baja	Alto	Riesgo Significativo	- Uso de backup en nube varias veces en semana. - Uso de backup en soporte físico de forma semanal. - Uso de herramientas de versionado.
RG006-Estimación temporal optimista	Establecimiento erróneo en la duración de una o varias tareas que añadan retraso en la consecución de los hitos marcados en el proyecto.	Media	Medio	Riesgo Significativo	- Seguimiento concienzudo de la planificación de forma semanal para detectar desviaciones. - Esfuerzo extra en caso necesario para recuperar la desviación.
RG007-Pérdida de recursos físicos	Fallo en alguno de los elementos usados para el desarrollo del proyecto.	Baja	Alto	Riesgo menor	- Disponer de recursos de backup por si fuera necesario.

Ilustración 5. Análisis de Riesgos
(Elaboración Propia)

1.5 Breve sumario de productos obtenidos

Una vez finalizado el presente proyecto, deberán de haberse obtenido los siguientes productos:

- Documentación: Se indicarán los aspectos más relevantes del plan expuesto para, en la medida de lo posible, justificarlos y detallarlos. Así mismo, se añadirán los anexos independientes que se consideren necesarios.
- Aplicación Android Nativa: Se proporcionará tanto el código como el instalable (fichero con extensión APK) de la aplicación desarrollada.
- Servidor *Back-End*: Se entregará el código ejecutable del servidor además de todo lo necesario para que su puesto en funcionamiento sea lo más simple posible, incluyendo si fuera necesario, datos de ejemplo.
- Presentación de diapositivas: Se proporcionará una presentación en formato Powerpoint con los detalles más reseñables del proyecto.
- Video presentación: Video que incluirá la presentación anterior con la *voz en off* del autor detallándola así como, aportando las justificaciones y/o conceptos que por su extensión, no hayan tenido cabida.

1.6 Breve descripción de los otros capítulos de la memoria

En este apartado se definen, con el fin de facilitar la lectura del presente documento, los capítulos presentados a continuación. Como se ha mencionado en apartados anteriores, el proyecto contará con tres fases principales (diseño, implementación y cierre), por lo que se presentarán **tres capítulos**, a los que se añadirán algunos apartados más, donde se incluirán bibliografía, anexos y glosario de términos y acrónimos.

Así, en primer lugar, dispondremos del **capítulo dedicado al diseño de la aplicación**, el cual podrá ser subdivido en el diseño físico/lógico y el diseño gráfico. Respecto al primero, se proporcionará la visión de todos los artefactos usados en el mismo, así como la justificación de uso de los mismos y posibles cambios de

tecnología que pudieran darse. En el apartado gráfico, se detallará todo el proceso de diseño centrado en el usuario, comenzando por los requisitos detectados, siguiendo con los primeros prototipos y su evaluación, para finalizar con el producto final, tratando de mostrar todo este proceso y llevando al lector a comprender todas las decisiones tomadas en este campo.

Seguidamente, en el **capítulo dedicado a la implementación**, se tratará de mostrar de forma cercana, todo el proceso de construcción de la solución, desde el proceso de elección de tecnologías hasta las pruebas finales efectuadas sobre el aplicativo.

En cuanto al cierre, se dedicará un apartado a las **conclusiones** de la elaboración de este proyecto, destacando aquellos puntos clave de su elaboración. Además, se incluirá una breve reflexión sobre líneas de actuación futuras.

Finalmente, en los últimos apartados se incluirá un **glosario de términos y acrónimos** con el fin de facilitar la lectura a aquellos de perfil técnico más bajo, bibliografía donde se detallarán todas las fuentes usadas para la elaboración de este documento y los anexos necesarios para comprender y apreciar mejor este proyecto.

2. Diseño

Una vez establecidos todos los aspectos cercanos a la gestión mostrados en el capítulo anterior, llegó el momento de diseñar la solución a la aplicación propuesta. En este apartado, se comenzará detallando el proceso de toma de requisitos, diseño, implementación y pruebas de la aplicación móvil a través de DCU, tras el cual quedarán definidos las “líneas maestras” del diseño gráfico e interactivo. Tras este punto, se finalizará el apartado mostrando la arquitectura de la solución completa a distintos niveles.

2.1 Diseño centrado en el usuario (DCU)

Tal y como se detalló en el apartado "1.3 Enfoque y método seguido", el proceso de DCU, destinado a ofrecer un diseño gráfico atractivo y con un alto grado de usabilidad, constará de 4 fases iterativas con el propósito de realizar un proceso de adaptación y mejora progresiva. A continuación, se presentan estas fases, mostrando el proceso de ejecución de las mismas así como, las conclusiones obtenidas.

2.1.1 Análisis de requisitos

Tal y como se detalló anteriormente esta primera fase está destinada a la obtención de toda aquella información relevante de los posibles usuarios finales (como hábitos, habilidades dificultades o deseos).

2.1.1.1 Métodos de indagación y contexto

Para la realización de esta investigación, y dado el carácter iterativo del proceso, **se creyó conveniente diferenciar entre el proceso a seguir para la primera iteración,** donde la cantidad de información a obtener será la base para construir los primeros modelos, **y el resto de iteraciones**, donde se obtendrán en principio, detalles y puntos de mejora y/o cambio respecto a la base.

Así, tras una profunda investigación sobre los métodos disponibles, se obtuvieron los siguientes:

- Observación/*Shadowing*: Método consistente en estudiar al usuario en su entorno, para así, obtener información relativa a sus costumbres, dificultades o necesidades. Según la bibliografía consultada (Jordi Almirall, 2013), la aplicación de este método puede ser de escaso interés en el caso de dispositivos móviles debido a la naturaleza portátil de los mismos, cambiando continuamente de contexto de uso.
- Entrevistas: Realización de una serie de preguntas a los usuarios finales con el objetivo de extraer el máximo de información posible respecto a las tareas o procesos de interés. Su mayor desventaja es que al ser el usuario el que aporte la información, éste puede obviar puntos importantes para el diseño final.
- Dinámicas de grupo: Parecido al método anterior, aunque con una figura moderadora cuyo fin es guiar el debate abierto a aquellos puntos de especial interés.

- Encuestas: Realización de cuestionarios a los usuarios finales con el objetivo de extraer el máximo de información posible respecto a las tareas o procesos de interés. Al igual que las entrevistas, la delegación en el usuario de la extracción de información puede ser un riesgo.
- *Logging*: Seguimiento informatizado de la interacción del usuario final con la aplicación. Aunque de aplicación relativamente sencilla, su mayor desventaja radica en el aislamiento total o parcial del usuario respecto a su contexto de uso habitual, lo que puede obviar detalles de alta importancia.
- Análisis competitivo: Uso y análisis de alternativas parecidas a la aplicación a elaborar para la extracción de posibles puntos de fallo o mejora. La mayor desventaja de este método radica en que el usuario final queda algo más "alejado" del proceso.

Una vez comprendidos los métodos y puestos en común con el tiempo de ejecución del proyecto y el presupuesto del mismo, **se decidió optar por el uso del método de análisis competitivo** en la primera iteración, **además del método de entrevistas**, que será usado en todas las iteraciones como se detallará más adelante. Los motivos que llevaron a esta elección son:

- Posibilidad de obtener una base bastante estable en lo que a diseño se refiere.
- Conocer a los competidores posibilitando la imitación de sus puntos fuertes y la mejora de los débiles.
- Seguir los convencionalismos del sector en lo que al apartado gráfico se refiere para una fácil transición entre aplicaciones (lo que facilitaría la llegada de usuarios tradicionales de dichas aplicaciones).
- Evitar errores detectados en aplicaciones competidoras.

Una vez claros los objetivos, se procedió a la aplicación del método a los competidores analizados en el apartado "1.1 Contexto y justificación del Trabajo", obteniendo las siguientes conclusiones:

- Se debe evitar la publicidad de marcas de alimentación.
- La importación de recetas, aunque atractiva, puede cometer errores que imposibiliten el seguimiento de una receta
- El uso de una interfaz moderna, atractiva y usable es un punto débil a resolver en la mayoría de competidores
- La mayoría optan por un modelo visual basado en "tarjetas" con un menú lateral deslizante para navegar entre las distintas opciones, el cual a priori, parece intuitivo y usable.
- La mayoría carece de opciones atractivas tecnológicamente que atraigan al sector más joven.
- Las interfaces suelen tener una paleta de colores transmisora de sentimientos relacionados con la edad adulta y no joven.
- Muchos de los competidores no hacen uso de la posibilidad de seguir paso a paso una receta, lo que le resta información sobre cómo proceder a ciertos usuarios cuyo conocimiento culinario sea reducido (como el nicho de mercado a abarcar, el joven).

Respecto al método general usado en todas las iteraciones, será el método basado en **entrevistas**. El motivo principal que llevó a su elección es que permitirá la adaptación de las cuestiones en cada iteración, obteniendo un posible perfilado de usuarios en las primeras iteraciones (donde el método elegido, análisis de competidores, está más alejado del usuario final) y, si fuera necesario, incidir en otras informaciones con posibilidad de ser relevantes en sucesivos ciclos.

Una vez claro el método a seguir, se elaboró el siguiente guion:

Guión de entrevista a usuarios
INFORMACIÓN DE USUARIO
Nombre
Edad
Estudios
Estado civil
Número de hijos
Ciudad Origen
Ciudad Actual
Profesión
HÁBITOS GENERALES
¿Cuanto tiempo dedica al móvil diariamente? (Horas)
¿Qué nivel de manejo considera que tiene sobre el dispositivo móvil?
¿Cuáles son los principales inconvenientes que se encuentra?
¿Para que lo usa mayoritariamente?
¿Qué valora más en una aplicación?
HÁBITOS CULINARIOS
¿Le gusta la cocina?
¿Vigila su alimentación tratando de que sea sana y equilibrada?
¿Qué porcentaje de consumo mensual atribuiría a los siguientes tipos de comida? Casera, a domicilio, en restaurantes, comida rápida.
¿Cuáles diría que son los mayores inconvenientes a la hora de preparar comida casera?
¿Conoce un amplio listado de recetas? ¿Le gustaría conocer más?
USO DE APLICACIONES DE ALIMENTACIÓN
¿Dónde acude cuando no conoce una determinada receta?¿En qué situación/es suele buscar esa información?
¿Usa aplicaciones como Just Eat, Uber Eats, Glovo, etc.? ¿Cuales son los principales motivos que le llevan a ello?
¿Conoce alguna aplicación de recetas?¿Y web? ¿Cuales?
¿Considera difícil el seguimiento de recetas extraídas/seguidas de/en Internet? ¿Por qué?

Ilustración 6. Guión de entrevista
(Elaboración propia)

Como se observa en la imagen anterior, la entrevista total, contiene un total de 22 preguntas divididas en cuatro apartados:

- Información de usuario: Sección dedicada a la obtención de información personal del usuario (se decidió esta denominación por la oposición que pudiera ofrecer el término "personal"). Esta información resultó útil para el perfilado posterior de los usuarios.
- Hábitos generales: El objetivo aquí es obtener patrones de consumo diario de aplicaciones móviles, con el objetivo, como en el caso anterior, de posteriormente elaborar perfiles de usuario.
- Hábitos culinarios: En este apartado, se comienza con la aproximación al sector de interés. En él, se busca además de añadir información de interés a los perfiles, comprender las necesidades y dificultades encontradas por los usuarios a la hora de cocinar.
- Uso de aplicaciones de alimentación: Sección intermedia entre las dos anteriores con la que se desea conocer el nivel de uso de aplicaciones móviles a la hora de elaborar comidas y, de paso, encontrar puntos débiles en los métodos actuales para evitar cometer esos mismos fallos.

Para la realización de la entrevista y, con el objetivo de abarcar un mayor público, se utilizó la herramienta Google Forms, herramienta dedicada a la generación de formularios y/o encuestas con funcionalidades útiles como resumen ejecutivo o exportación a hojas de cálculo.

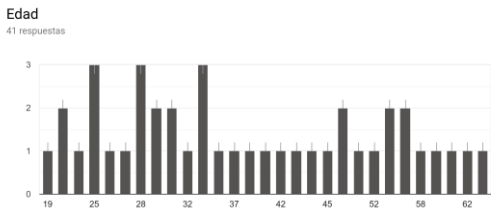


Ilustración 7. Gráfico de edad de los entrevistados
(Elaboración de *Google Forms*)

Así, a continuación, puede observarse la información obtenida a raíz de la realización de **43 entrevistas a usuarios** en un rango de edad amplio (observable en la imagen anterior). De forma general, se ha evidenciado:

- La duración de la batería es una preocupación.
- Gran atracción actual hacia la comida sana y variada.
- Se busca la sencillez y la rapidez a la hora de elaborar recetas, potenciado por la falta de tiempo general.
- Gusto general por la cocina.
- Se valora consistentemente la fluidez y la ausencia de fallos.
- La usabilidad es con diferencia, la característica más valorada.
- Se declaran problemas en el uso de las herramientas actuales, sobre todo en cuanto al seguimiento y al detalle (por ejemplo, punto de cocción del alimento).
- Preferencia por aplicaciones gratuitas, sin publicidad y sin registro obligatorio.

De forma más detallada, y dado que **la edad, es un factor a tener en cuenta** (sobre todo cuando se buscan conclusiones sobre cuestiones tecnológicas), se ha decidido **separar por grupos de edad**, siendo los dos primeros grupos usuarios objetivo y el tercero, usuarios a tener en cuenta:

- Grupo de edad entre 19 y 25 años
    - Estudiantes.
    - Sin hijos.
    - Uso intensivo del móvil (el 87% declara un uso mayor de 4 horas diarias).
    - Nativos tecnológicos: Todos los entrevistados declaran un conocimiento alto u experto de los dispositivos móviles.
    - Consumen en su mayoría comida casera (probablemente por los recursos económicos ajustados).
    - Se busca la sencillez y variedad en la alimentación.
    - Su fuente de conocimiento culinario se basa en Internet y en responsables legales y/o familiares.

- Grupo de edad entre 26 y 38 años
    - Trabajadores en activo.
    - Sin hijos mayoritariamente.
    - El uso del móvil, aunque alto, es algo más bajo que el grupo anterior (entre 1 y 4 horas).
    - Aumento de la comida no casera (a domicilio, comida rápida y restaurantes) provocado posiblemente por el aumento de recursos económicos.
    - Su fuente de conocimiento sobre cocina es Internet, aunque destacando grupos de Facebook o determinados perfiles de Instagram.

- Grupo de edad entre 40 y 63 años
    - Trabajadores en activo.
    - Con hijos.
    - Uso de móvil mucho más bajo, con una media de 1 o 2 horas diarias.
    - Se declara un conocimiento bajo o medio sobre los dispositivos móviles, probablemente por afectación parcial de la brecha digital.
    - Sensibles a los cambios y/o actualizaciones bruscas.
    - Consumen comida casera mayoritariamente y, excepcionalmente, acuden a restaurantes.
    - Se observa un uso mayor de Thermomix (potenciado por el aumento de ingresos probablemente).
    - Internet y Facebook son las principales fuentes de información.

Tras analizar en detenimiento toda la información recogida durante la indagación, se obtuvieron las siguientes **conclusiones respecto al diseño**:

- El consumo de batería de la aplicación debe ser el mínimo posible.
- La usabilidad y la fluidez deben ser pilares básicos en el diseño.
- Mostrado de alimentos y utensilios necesarios en una pantalla intermedia entre la selección de la receta y el inicio de la misma para facilitar su conocimiento a los usuarios a la hora de realizar la compra.
- El flujo principal (acceso-búsqueda-selección-ejecución de receta) debe ser lo más ágil y corto posible.
- La pantalla principal, de búsqueda y de mostrado de utensilios y/o ingredientes debe tener un contraste y colorido suficientemente destacable para ser visualizada en exterior (situaciones con mucha luz).
- La pantalla de seguimiento paso a paso deberá estar preparada para ejecución en interior.
- Destacar el aporte nutricional con etiquetas del tipo "bajo en grasa" o "bajo en azúcares" podría ayudar a atraer público interesado en la nutrición sana.
- Necesidad de destacar la dificultad y tiempo de la receta de forma clara.
- Especificar los pasos a seguir en la elaboración de forma muy detallada y clara.
- Evitar en futuras actualizaciones, cambios bruscos en la interfaz.
- Registro opcional en la aplicación, intentando atraer a los usuarios hacia el mismo mediante la explicación de las ventajas que obtendrían.

Respecto al **aspecto más cercano al negocio**, se obtuvieron conclusiones importantes que merece la pena destacar:

- Dada la tendencia actual del público, el anuncio de la aplicación en foros de Facebook y su publicitación en Google, podría atraer nuevos clientes de forma efectiva.
- Evitar el uso intensivo de publicidad en la aplicación.
- Gratuidad de descarga en la aplicación. Su monetización podría venir por la extensión de características o el acceso preferente a nuevas recetas.

Finalmente, y dado que el número de entrevistas fue elevado y su adición directa a este documento desvirtuaría su objetivo, se han almacenado en la nube y pueden consultarse a través del este enlace o bien en el Anexo III - Respuestas Entrevista.

2.1.1.2 Perfilado de usuarios

En este apartado se mostrarán los perfiles de usuario identificados tras efectuar los métodos de indagación descritos anteriormente. Estas agrupaciones tienen por objetivo aglutinar usuarios con unas determinadas características (pudiendo ser éstas de diversa índole) de forma que las fases DCU posteriores, se centren en ellos, tratando de satisfacer todas sus necesidades. Así, se han distinguido tres perfiles distintos cuyas características se muestran a continuación:

PRFL001 - Usuario olvidadizo				
Edad:	Mayor de 18 años		Estado Civil:	Cualquiera
Situación Laboral:	Cualquiera		Hijos:	0 o más
Hábitos tecnológicos	• Uso del dispositivo móvil variado. • Conocimiento de la tecnología variado. • Puede o no tener conocimiento de la oferta actual en cuanto a aplicaciones y posibilidades.			
Hábitos culinarios:	• Conoce recetas pero desconoce cómo elaborarlas. • **Elabora comida sin prestar atención a la misma.** • **Suele olvidar recetas ya realizadas**. • Preocupado por llevar una alimentación saludable.			
Innovación culinaria:	• Prefiere acudir a recetas conocidas. • Raramente tratará de buscar recetas nuevas.			
Contexto de Uso:	Este perfil usará la aplicación tanto en su hogar como fuera del mismo con la intención de buscar una receta concreta para la cual ha olvidado los pasos de elaboración y los ingredientes necesarios. Por este motivo, es probable que realice sus búsquedas o bien antes de ir a la compra o bien en el momento de cocinar.			
Tareas:	• Registro de usuario. • Inicio de sesión. • Búsqueda de recetas por nombre. • Acceso al resumen de la receta. • Acceso a los ingredientes y utensilios necesarios. • Acceso al paso a paso de la receta.			
Listado de Características:	• Opcionalidad en el registro de usuario. • Instrucciones claras y precisas de elaboración de recetas. • Mostrado de aporte nutritivo de la receta. • Mostrado de utensilios e ingredientes de la receta.			

Ilustración 8. Perfil usuario olvidadizo
(Elaboración propia)

PRFL002 - Usuario curioso				
Edad:	Mayor de 39 años		Estado Civil:	Casado
Situación Laboral:	Trabajador		Hijos:	1 o más
Hábitos tecnológicos	• Uso bajo-medio del dispositivo móvil (entre 1 y 2 horas) • Cercano a la tecnología pero con reticencias • Desconoce todas las posibilidades y aplicaciones disponibles.			
Hábitos culinarios:	• Conoce una amplia gama de recetas. • Elabora comida habitual sin necesidad de guía. • **Gran desempeño en la cocina.** • Gusto por la cocina. • Concienciado acerca de la alimentación saludable.			

Innovación culinaria:	• **Regularmente busca innovar en la cocina.** • Su innovación proviene de redes sociales o Internet.
Contexto de Uso:	Este perfil usará la aplicación normalmente en casa, durante un periodo de esparcimiento en el cual tratará de buscar nuevas recetas que puedan atraerle o que haya visualizado anteriormente en redes sociales y/o internet, con el objetivo de conocer los ingredientes y su elaboración para, posteriormente, realizar la compra y finalmente, ejecutar la receta.
Tareas:	• Registro de usuario. • Inicio de sesión. • Búsqueda sin filtro o por nombre. • Acceso al resumen de la receta. • Acceso a los ingredientes y utensilios necesarios. • Acceso al paso a paso de la receta.
Listado de Características:	• Opcionalidad en el registro de usuario. • Ausencia de publicidad. • Acceso rápido a la búsqueda de recetas sin filtro. • Alta sencillez de uso. • Ficha resumen de la receta. • Mostrado de utensilios e ingredientes de la receta.

Ilustración 9. Perfil usuario curioso
(Elaboración propia)

PRFL003 - Usuario simplista				
Edad:	De 18 a 39 años		Estado Civil:	Soltero
Situación Laboral:	Estudiante/Trabajador		Hijos:	0
Hábitos tecnológicos:	• Uso alto o muy alto del dispositivo móvil (más de dos horas diarias). • Nativo digital, acostumbrado al uso de la tecnología. • Conoce la amplia oferta disponible respecto a aplicaciones y posibilidades.			
Hábitos culinarios:	• **Reticencia a la cocina**. • **Prefiere recetas simples y rápidas.** • Busca tener una alimentación saludable. • No conoce una amplia cantidad de recetas.			
Innovación culinaria:	• Baja, motivada por el descubrimiento mediante redes sociales.			
Contexto de Uso:	Probablemente este usuario realice uso de la aplicación exclusivamente dentro del hogar instantes antes de comenzar a cocinar y con una receta como objetivo. Esta receta será rápida y simple, de forma que no entrañe dificultad y conlleve un tiempo de elaboración bajo, de lo contrario, no la realizará.			
Tareas:	• Registro de usuario. • Inicio de sesión. • Búsqueda de recetas por duración. • Búsqueda de recetas por dificultad. • Búsqueda de recetas por nombre. • Acceso al resumen de la receta. • Acceso a los ingredientes y utensilios necesarios.			

	• Acceso al paso a paso de la receta.
Listado de Características:	• Especificación de dificultad de las recetas. • Especificación del tiempo necesario de elaboración de las recetas. • Fluidez. • Instrucciones claras y precisas de elaboración de recetas. • Mostrado de aporte nutritivo de la receta.

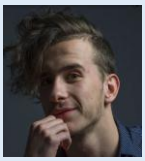
Ilustración 10. Perfil usuario simplista
(Elaboración propia)

2.1.2 Diseño conceptual

Una vez se han recogido, analizado y elaborado conclusiones de la información obtenida en la fase anterior, se comienza este apartado, donde se convertirá en pautas, decisiones y líneas maestras que permitirán en fases posteriores, crear la interfaz gráfica de la aplicación aquí presentada, teniendo en cuenta en todo momento, los usuarios finales de la misma. Para esta labor, en primer lugar se establecerán unos "personajes tipo" que representarán a cada uno de los perfiles, para posteriormente situarlos en unos escenarios de uso determinados. Finalmente, se construirá el flujo de interacción de la aplicación.

2.1.2.1 Técnica de personas o personajes

En primer lugar, para comenzar con esta fase del DCU, se decidió aplicar la técnica de personas o personajes. Este procedimiento permite caracterizar los perfiles expuestos anteriormente en forma de personas ficticias tipo con el fin de acercarlos al "plano real" con el objetivo de potenciar la inclusión del usuario en el proceso. Así, se obtuvieron los siguientes:

PJ001 – Vicente Simplón Martínez	
	Vicente Simplón es un chico de 23 años en su último año de universidad. Actualmente se encuentra finalizando 3 asignaturas por lo que ha decidido emplear parte de su tiempo en un trabajo a tiempo parcial en un pub. Vive en un piso compartido con otros estudiantes y entre los estudios y el trabajo su tiempo libre es limitado, decidiendo gastar parte de éste en una pachanga de fútbol con amigos de su facultad.
Hábitos culinarios:	Vicente tiene grandes pasiones, pero lamentablemente la cocina no es una de ellas. Aunque le gusta mantener una alimentación saludable (pues parece que ha heredado colesterol alto de su padre), a menudo acude a recetas sencillas o directamente a comida rápida. Raras veces decide probar nuevas comidas desde que en un restaurante asiático pensó que el *wasabi* era guacamole, no obstante, a veces, cuando quiere impresionar a una visita, suele tratar de hacer una receta, cuya puntuación, a menudo se queda en "comestible".
Desempeño tecnológico:	Este chico es amante de la tecnología y, aunque su *smartphone* es usado mayoritariamente para acceder a redes sociales y llamar a sus familiares (pues se ha desplazado de su ciudad natal para estudiar), controla funciones algo más avanzadas como el desbloqueo facial. Sus amigos a menudo tienen que llamarle la atención pues, usa el móvil más de lo que debería (más de 4 horas diarias). A parte de lo comentado, también consume contenido multimedia (Youtube mayoritariamente) y, en ocasiones, juega a algún juego.

Objetivos:	• Encontrar recetas sencillas para cualquier comida. • Encontrar recetas rápidas para cualquier comida. • Variedad de recetas • Saber lo sano que será el plato que va a cocinar.
Necesidades:	• Buscar recetas por nivel de dificultad. • Buscar recetas por duración. • Visualizar el aporte nutricional. • Visualizar los utensilios e ingredientes de la receta.

Ilustración 11. Personaje perfil simplista
(Elaboración propia)

PJ002 – Olvido Figueroa Sánchez	
	Olvido Figueroa es una ingeniera de 33 años que actualmente trabaja para una gran multinacional. Su puesto de trabajo actual, le permite disponer de varias horas libres por la tarde que aprovecha para ver su serie favorita, pasear o entrenar su gran pasión, el *judo*. Vive con su novio Fernando, al que conoció hace 4 años por un amigo en común.
Hábitos culinarios:	Olvido tiene muchos puntos fuertes, pero la cocina no es uno de ellos. Dado que su chico dispone de menos tiempo libre, a menudo se encarga de cocinar, para lo que recurre a recetas extraídas de redes sociales o Youtube (éste último menos usado pues le causa enfado el pausar e iniciar el video con las manos sucias). De una vez a otra, no recuerda los ingredientes, lo que le obliga a buscarlos en Internet o recurrir a su madre, además, muchas veces, se le ha pasado el punto de cocción del arroz pues se le olvidó que estaba cociéndose. A pesar de esto, algunas veces trata de sorprender a su chico con un plato nuevo.
Desempeño tecnológico:	Desde sus estudios, Olvido domina la tecnología más de lo que la ama. Suele usar el móvil entre 2 y 4 horas al día en su mayoría para acudir a redes sociales y reproducir música. Aunque ciertas características avanzadas son desconocidas para ella, sí que controla otras medias o básicas.
Objetivos:	• Encontrar una receta que probó con anterioridad. • Ser avisada de que debe apartar la comida del fuego para obtener un buen punto de cocción. • Poder pasar al siguiente paso sin necesidad de tocar el móvil. • Acceder de forma sencilla a la información básica de la receta (ingredientes, utensilios, dificultad, tiempo, etc.).
Necesidades:	• Buscar recetas por nombre. • Visualizar la dificultad y descripción de la receta. • Visualizar los ingredientes y utensilios. • Obtener el punto de cocción exacto en sus comidas. • Interfaz vocal.

Ilustración 12. Personaje perfil olvidadizo
(Elaboración propia)

PJ003 – Susana Fisgón Romero	
	Susana Fisgón es una administrativa de 44 años, divorciada y con dos hijos, Rubén y Elena de 12 y 4 años, respectivamente. Aunque las mañanas son un caos, por la tarde dispone de todo el tiempo que sus dos “terremotos” le permiten, empleándolo en su gran pasión, los puzles. Actualmente, vive en una casa con jardín en su pueblo natal, donde dispone de todo lo que necesita y tiene a sus amigas de la infancia con la que comparte momentos únicos.

Hábitos culinarios:	Susana conoce infinidad de recetas pues le encanta la cocina. Trata que sus hijos lleven una alimentación equilibrada lo que no resulta sencillo, pues odian las verduras lo que la obliga a buscar nuevas recetas en las que poder alimentarlos adecuadamente. A ella le encanta las recetas de cremas de verduras, y siempre está buscando nuevas. Igualmente, ocasionalmente, recuerda un plato de su infancia del que no recuerda su elaboración, acudiendo a menudo a Internet.
Desempeño tecnológico:	Susana ha vivido la revolución tecnológica en su esplendor y a menudo se ha sentido algo desplazada. Aunque maneja su *Smartphone* a nivel básico mayoritariamente para el acceso a redes sociales, ve que sus hijos, con menos edad, lo controlan a la perfección e incluso, en ocasiones su hijo mayor le enseña nuevas funcionalidades. Su principal preocupación es un cambio radical en las aplicaciones que usa pues a menudo deja de poder realizar acciones por desconocer su nuevo método.
Objetivos:	• Encontrar nuevas recetas para que sus hijos coman verduras. • Procedimiento sencillo para poder buscar, elegir y seguir una receta. • Visualizar la lista de ingredientes antes de comenzar para asegurar que dispone de los ingredientes necesarios. • Buscar recetas de temporada, para elaborar platos acorde al tiempo.
Necesidades:	• Búsqueda de recetas por tipo. • Navegación sencilla e intuitiva. • Mostrado de ingredientes/utensilios antes de empezar. • Información general de la receta. • Información de aporte nutricional.

Ilustración 13. Personaje perfil curioso
(Elaboración propia)

2.1.2.2 Escenarios de uso

Una vez caracterizados los distintos perfiles, en este apartado se tratan los escenarios de uso. Un escenario de uso es una descripción detallada de una situación realista y concreta en la que un usuario interacciona con el sistema con un/os objetivo/s concreto/s. Los objetivos principales de estos artefactos son los de descubrir necesidades, determinar comportamientos, generar una estructura de la aplicación y establecer flujos de interacción, información usada posteriormente para elaborar los prototipos. Así, se definen los siguientes:

EU001-Búsqueda de receta para preparación de lista de la compra	
Perfil Interventor:	Usuario olvidadizo, Usuario curioso
Personaje:	Susana Fisgón Romero
Contexto:	Susana se encuentra en la cocina de su casa sentada junto a la mesa. Está elaborando la lista de la compra pues al día siguiente tiene pensado acudir al supermercado a por provisiones. De repente, recuerda que quiere realizar salmorejo pero no está segura de sí éste lleva ajo, por ello, decide buscar la receta en la aplicación y ver si le haría falta este ingrediente.
Objetivos:	Ver la lista de ingredientes necesarios para elaborar una receta determinada.
Tareas llevadas a cabo:	Buscar receta por nombre, Acceder a receta
Necesidades de información:	Conocer el nombre de la receta
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador y escribir el nombre de la receta 4. Navegar por el listado 5. Acceder a la receta 6. Ver los ingredientes/utensilios necesarios

Ilustración 14. Escenario EU001
(Elaboración propia)

EU002-Búsqueda de receta para compra de ingredientes	
Perfil Interventor:	Usuario olvidadizo, Usuario simplista
Personaje:	Vicente Simplón Martínez
Contexto:	Vicente se encuentra en su supermercado de confianza realizando la compra para varios días y mientras pasa por la zona de frutería, recuerda que mañana quiere hacer un *smoothie* de fruta. Como no sabe qué ingredientes lleva, decide buscar la receta y acceder a estos para verlos.
Objetivos:	Ver la lista de ingredientes necesarios para elaborar una receta determinada de forma rápida.
Tareas llevadas a cabo:	Buscar receta por nombre, Acceder a receta
Necesidades de información:	Conocer el nombre de la receta
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador y escribir el nombre de la receta 4. Navegar por el listado 5. Acceder a la receta 6. Ver los ingredientes/utensilios necesarios

Ilustración 15. Escenario EU002
(Elaboración propia)

EU003-Búsqueda de receta conocida con antelación media-alta	
Perfil Interventor:	Usuario olvidadizo, Usuario simplista
Personaje:	Olvido Figueroa Sánchez
Contexto:	Olvido se encuentra viendo un capítulo de la serie *Stranger Things* con su chico en el salón de su piso cuando recuerda que mañana vendrán unos amigos a comer arroz de pollo. Como no recuerda cómo se elaboraba, decide realizar una búsqueda en la aplicación tranquilamente para saber si la receta está disponible y su información para poder planificarse.
Objetivos:	Buscar una receta por nombre
Tareas llevadas a cabo:	Buscar receta en base al nombre
Necesidades de información:	Conocer el nombre de la receta
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador y escribir el nombre de la receta 4. Navegar por el listado 5. Acceder a la receta

Ilustración 16. Escenario EU003
(Elaboración propia)

EU004-Búsqueda de receta de acuerdo a unos requisitos con antelación media-alta	
Perfil Interventor:	Usuario simplista
Personaje:	Vicente Simplón Martínez
Contexto:	Vicente se encuentra volviendo a su piso tras una tarde de futbol con los amigos. De repente decide buscar mientras va caminando, una receta sencilla, rápida y con pocas calorías para poder cenar ese mismo día.
Objetivos:	Buscar una receta en base a unos requisitos determinados
Tareas llevadas a cabo:	Buscar receta en base a unos criterios determinados
Necesidades de información:	Conocer los criterios que se desean aplicar en la búsqueda de la receta
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador 4. Establecer filtros acorde a los requisitos deseados 5. Navegar por el listado 6. Acceder a la receta

Ilustración 17. Escenario EU004
(Elaboración propia)

EU005-Búsqueda de receta innovadora con antelación media-alta	
Perfil Interventor:	Usuario curioso
Personaje:	Susana Fisgón Romero
Contexto:	Susana ha acostado a sus hijos tras un duro viernes de trabajo. Como esta semana ha comido pocas verduras, decide que el sábado a mediodía comerán un plato con ellas. Para ello empieza a buscar una receta sana que contenga verduras pero no se aprecien para evitar la aversión de sus hijos
Objetivos:	Buscar una receta no conocida en base a unos requisitos determinados o sin ellos
Tareas llevadas a cabo:	Buscar receta en base a unos criterios determinados o sin ellos

Necesidades de información:	Requisitos a aplicar en la búsqueda (Opcional)
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador 4. [Establecer filtros acorde a los requisitos deseados] 5. Navegar por el listado 6. Acceder a la receta

Ilustración 18. Escenario EU005
(Elaboración propia)

EU006-Búsqueda de receta conocida con antelación baja-muy baja	
Perfil Interventor:	Usuario olvidadizo, Usuario simplista
Personaje:	Olvido Figueroa Sánchez
Contexto:	Olvido se encuentra en la cocina decidida a realizar una exquisita crema de calabaza. Una vez se dispone a sacar los ingredientes y utensilios de cocina, se da cuenta que no recuerda bien los ingredientes por lo que decide buscar la receta para visualizarlos.
Objetivos:	Buscar una receta por nombre
Tareas llevadas a cabo:	Buscar receta en base al nombre
Necesidades de información:	Conocer el nombre de la receta
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador y escribir el nombre de la receta 4. Navegar por el listado 5. Acceder a la receta

Ilustración 19. Escenario EU006
(Elaboración propia)

EU007-Búsqueda de receta de acuerdo a unos requisitos con antelación baja-muy baja	
Perfil Interventor:	Usuario simplista
Personaje:	Vicente Simplón Martínez
Contexto:	Son las ocho de la tarde y Vicente se dispone a hacerse un plato que no le lleve más de 20 minutos realizarlo y sea sencillo, pues es tarde y está cansado después de estudiar y trabajar. Para ello acude a EasyCook! con el objetivo de buscar una receta y visualizar los ingredientes.
Objetivos:	Buscar una receta en base a unos requisitos determinados
Tareas llevadas a cabo:	Buscar receta en base a unos criterios determinados
Necesidades de información:	Conocer los criterios que se desean aplicar en la búsqueda de la receta
Funcionalidades necesarias:	Búsqueda de receta, acceso a receta
Desarrollo de tareas:	1. [Registrarse] 2. [Iniciar Sesión] 3. Acceder al buscador 4. Establecer filtros acorde a los requisitos deseados 5. Navegar por el listado 6. Acceder a la receta

Ilustración 20. Escenario EU007
(Elaboración propia)

EU008-Ejecución de receta paso a paso por interfaz táctil	
Perfil Interventor:	Usuario olvidadizo, Usuario curioso, Usuario simplista
Personaje:	Olvido Figueroa Sánchez
Contexto:	Olvido se encuentra en la cocina de su piso con su chico a punto de preparar un delicioso arroz de pollo para 4 personas. Una vez selecciona la receta, comienza con el primer paso en el que debe cortar las pechugas de pollo. Una vez termina y dado que su tiene las manos sucias, le pide a su novio que está colocando la compra que por favor, pase al siguiente paso pulsando en la pantalla.
Objetivos:	Ejecutar una receta buscada anteriormente.
Tareas llevadas a cabo:	Comenzar receta, Seguir paso a paso receta
Necesidades de información:	Ninguna
Funcionalidades necesarias:	Avanzar a paso siguiente, Retroceder a paso anterior
Desarrollo de tareas:	1. Comenzar receta 2. [Visualizar ingredientes y utensilios] 3. Realizar paso y pasar al siguiente mediante la pantalla táctil (Iterativo) 4. Finalizar receta

Ilustración 21. Escenario EU008
(Elaboración propia)

EU009-Ejecución de receta paso a paso por interfaz vocal	
Perfil Interventor:	Usuario olvidadizo, Usuario curioso, Usuario simplista
Personaje:	Susana Fisgón Romero
Contexto:	Se aproxima la hora de comer y Susana decide hacer un delicioso puré de verduras para ella y sus dos pequeños. Una vez vistos y comprobados los ingredientes, decide comenzar con el primer paso: "Cortar 100 grs de zanahoria en pequeños daditos de 1cm x 1cm". Tras finalizar, decide pasar al siguiente paso pero como tiene las manos sucias no puede pulsar la pantalla o la mancharía.
Objetivos:	Ejecutar una receta buscada anteriormente.
Tareas llevadas a cabo:	Comenzar receta, Seguir paso a paso receta
Necesidades de información:	Ninguna
Funcionalidades necesarias:	Avanzar a paso siguiente, Retroceder a paso anterior
Desarrollo de tareas:	1. Comenzar receta 2. [Visualizar ingredientes y utensilios] 3. Realizar paso y pasar al siguiente mediante la pantalla táctil (Iterativo) 4. Finalizar receta

Ilustración 22. Escenario EU009
(Elaboración propia)

EU010-Establecer cronómetro de tarea paralela	
Perfil Interventor:	Usuario olvidadizo, Usuario curioso, Usuario simplista
Personaje:	Olvido Figueroa Sánchez
Contexto:	Olvido está en la cocina con su chico. Actualmente está acabando la receta del Arroz de pollo, por lo que llega el momento de echar a hervir el arroz. Una vez lo echa, tardará en cocinarse 20 minutos.
Objetivos:	Reducir el tiempo de elaboración mediante la ejecución de tareas en paralelo con cronómetro.
Tareas llevadas a cabo:	Establecer cuenta atrás para finalización de tarea
Necesidades de información:	-
Funcionalidades necesarias:	Cuenta atrás, Paralelización de tareas, Seguimiento paso a paso de receta
Desarrollo de tareas:	1. Llegar a una tarea paralelizable de una receta ya comenzada 2. Una vez realizado el paso, pasar a siguiente paso 3. Aceptar establecimiento de cuenta atrás

Ilustración 23. Escenario EU010
(Elaboración propia)

EU011-Parar cronómetro de tarea paralela	
Perfil Interventor:	Usuario olvidadizo, Usuario curioso, Usuario simplista
Personaje:	Vicente Simplón Martínez
Contexto:	Es sábado por la tarde y Vicente se encuentra cocinando lasaña de verdura pues, esa noche vendrá a cenar una compañera del trabajo que le gusta. Actualmente, la lasaña está en el horno, Vicente está en el salón estudiando y acaban de pasar los 25 minutos que debía estar en él.
Objetivos:	Parar el aviso de una cuenta atrás establecida que avisa de finalización de una tarea paralela.
Tareas llevadas a cabo:	Parar aviso fin cuenta atrás
Necesidades de información:	-
Funcionalidades necesarias:	Cuenta atrás, Aviso/Notificación fin de cuenta atrás, Paralelización de tareas, Seguimiento paso a paso de receta
Desarrollo de tareas:	1. Haber establecido una cuenta atrás de una tarea paralelizable de una receta ya comenzada 2. Recibir aviso/notificación de cuenta atrás finalizada 3. Parar aviso 4. Seguir en el paso en el que se estaba en el momento de recibir la notificación

Ilustración 24. Escenario EU011
(Elaboración propia)

2.1.2.3 Flujos de interacción

Para finalizar con la fase de diseño conceptual y una vez quedan claros y conceptualizados los distintos perfiles, personajes y posibles escenarios de uso, se ha de generar el flujo de interacción. Se denomina así, a un artefacto dedicado a expresar el paso a paso de una interacción que en este caso, será persona-aplicación. Con él,

se intenta detallar la interacción en busca de posibles puntos de mejora y/o ideas innovadoras. Así, para la aplicación tratada en este proyecto, se diseñó el siguiente flujo de interacción:

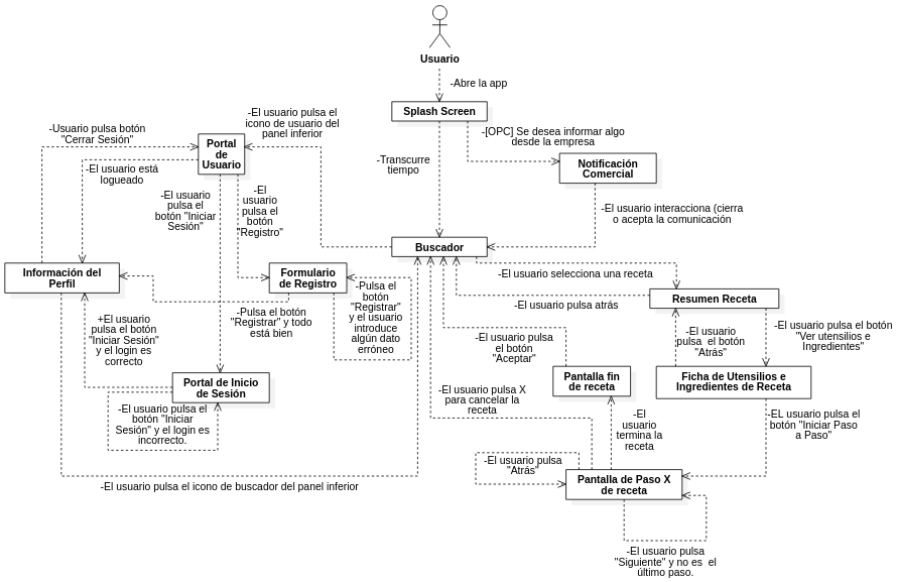


Ilustración 25. Flujo de interacción (Elaboración propia)

Por último, como puntualización, añadir que al tratarse de una aplicación Android pensada para *smartphones*, la mayoría de las pantallas aquí mostradas, permitirán la navegación inversa por el flujo, mediante el botón "Retroceso" presente en la mayoría de los dispositivos de forma física o virtualizada.

2.1.3 Implementación

Una vez se han analizado, conceptualizado y diseñado las primeras aproximaciones al diseño final en los apartados anteriores, comienza la tercera fase de DCU, la implementación. Durante esta fase, se comenzó por un prototipo de baja fidelidad (*sketch*) para poco a poco y tras un trabajo de refinamiento, obtener un prototipo de alta fidelidad que permita una evaluación consistente y realista de la experiencia de usuario en la fase 4.

2.1.3.1 Sketches

Como se comentaba anteriormente, a continuación, se muestran *sketches* de la aplicación. Los *sketches* a mano alzada son una de las opciones de prototipado de baja fidelidad, los cuales permiten una aproximación rápida al aspecto gráfico final de la aplicación, permitiendo, en caso necesario realizar modificaciones de la misma incurriendo en un coste temporal y económico casi despreciable. Así, para las distintas interfaces presentadas en el flujo de interacción, se realizaron los siguientes sketches:

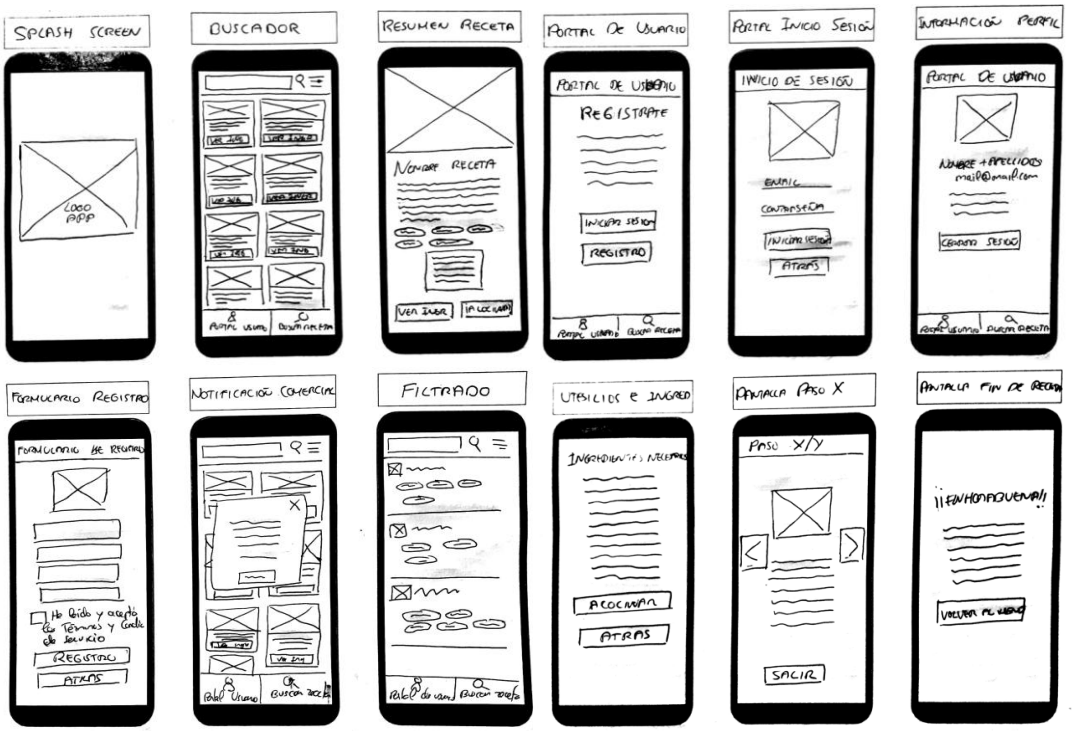
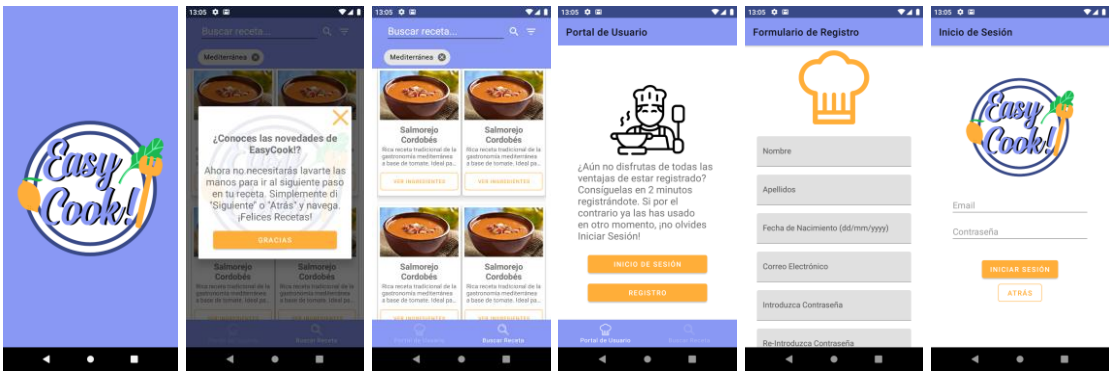


Ilustración 26. Sketches
(Elaboración propia)

2.1.3.2 Prototipos horizontales de alta fidelidad

Finalmente, y una vez se aproximó y valoró adecuadamente el diseño anterior, se realizó un prototipado de alta fidelidad. Este tipo de modelado tiene por objetivo generar una aproximación lo más realista posible del diseño final de la aplicación (en ocasiones, constituyendo éste), de forma que en la fase posterior (Pruebas) se pueda evaluar con garantías de que se obtendrán unos resultados realistas y acordes a la realidad.

Dado que la aplicación aquí presentada será nativa, se decidió usar como software de prototipado el propio entorno integrado de desarrollo de Android, Android Studio, para así evitar duplicidades posteriores en la generación de interfaces y permitir una funcionalidad simulada en concordancia con la esperada al final del proyecto. Así, se generaron las siguientes interfaces:



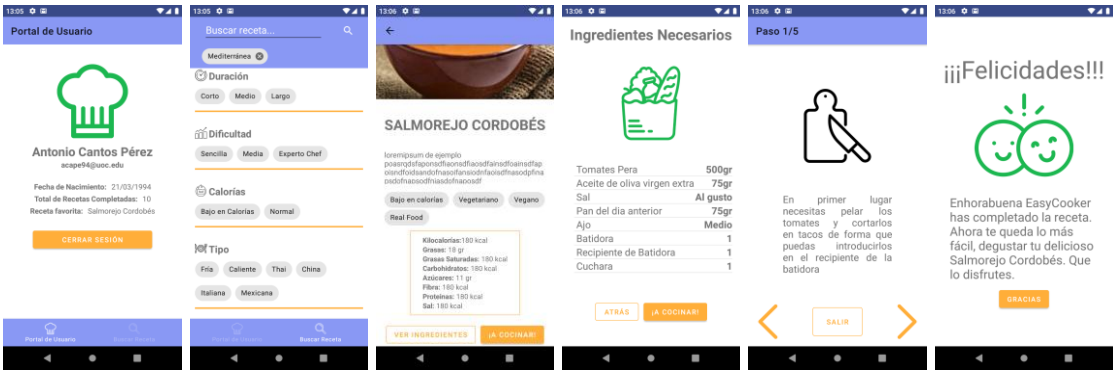


Ilustración 27. Prototipo de alta fidelidad (Elaboración propia)

A continuación, se presenta una descripción de las interfaces presentadas en la ilustración anterior, con el fin de esclarecer su funcionalidad (de izquierda a derecha y de arriba abajo):

- Splash screen: Primera pantalla mostrada al usuario. En ella se presenta el logo de la aplicación, teniendo una doble función: por un lado se le realizará una "inmersión" al usuario y por otra, se podrán precargar recursos posteriores. Su desaparición es automática tras un tiempo no extenso.
- Notificación comercial: Esta ventana será opcional. Se mostrará o no al usuario en función de futuras decisiones de negocio. Para salir de ésta, bastará con pulsar en el botón "Gracias" o un punto fuera de este diálogo.
- Buscador: Pantalla principal de la aplicación desde donde el usuario podrá buscar y filtrar recetas al gusto.
- Portal de usuario sin usuario autenticado: Pantalla que se mostrará si el usuario accede al portal de usuario y no está autenticado. En ella, se potencia y trata de concienciar al usuario de las ventajas que poseerá si decide iniciar sesión o registrarse.
- Formulario de registro: Formulario a rellenar por el usuario en caso de querer registrarse en la aplicación. En él se solicitan los datos personales clave para el registro así como, la aceptación de términos y condiciones de servicio.
- Portal de inicio de sesión: Interfaz de inicio de sesión desde la que el usuario podrá entrar en su cuenta.
- Información de perfil: Interfaz mostrada una vez el usuario ha iniciado sesión. En este apartado se muestra información básica de su perfil así como, se da la opción de cerrar sesión.
- Filtro de buscador: Se muestran al usuario los distintos filtros disponibles por tipo de comida, aporte calórico, etc.
- Resumen receta: Interfaz mostrada una vez la receta ha sido escogida por el usuario. En ella se muestra una breve descripción de la receta, una imagen, su título y el contenido nutricional. También, se da opción al usuario de ver los ingredientes y utensilios o comenzar la receta.
- Ficha de utensilios e ingredientes de receta: Pantalla donde se muestran al usuario todos los elementos necesarios para elaborar su receta, dando la opción de volver atrás o comenzar la receta.
- Pantalla de paso X de receta: Interfaz mostrada donde se indica el paso actual de la receta con una imagen y un texto descriptivo. En ella se proporciona la posibilidad de pasar al siguiente paso o retroceder de forma manual o por voz.
- Pantalla fin de receta: Pantalla de final de receta felicitando al usuario por la consecución de la receta y con un pequeño mensaje motivador. Incluye un botón que devolverá al interfaz principal (buscador).

Finalmente, conviene aclarar que el prototipo aquí mostrado constituye la **primera aproximación que se realizó al modelado final**, persiguiendo comprobar la usabilidad base de la aplicación. Por otro lado, algunas de las decisiones de diseño más relevantes tomadas son las siguientes:

- Aunque en la toma de requisitos los usuarios dejaron clara su postura en contra de la publicidad, la interfaz de comunicación comercial podría, gracias a su facilidad de supresión, constituir un buen canal de transmisión al usuario de nuevas funcionalidades, ventajas o cambios de forma, con el objetivo de que aquellos usuarios más sensibles a éstos, tengan una transición más suavizada.
- Se ha decidido la opcionalidad del registro e inicio de sesión para así, en primera instancia, atraer usuarios de forma más sencilla (por la aversión al registro transmitida en fases anteriores) y, con un trabajo de fidelización posterior, generarles la "necesidad" de registro.
- Aunque la barra inferior de navegación está desaconsejada cuando el número de pestañas es inferior a 3, su uso en aplicaciones de redes sociales como Facebook, Instagram o Twitter, harán que el usuario relacione conceptos y, de esta forma, se pueda explotar el conocimiento y hábitos adquiridos. Por otro lado, constituye una buena decisión a futuro si se deseara ampliar la funcionalidad de la aplicación.
- El uso de tarjetas ha sido necesario para mostrar al usuario una imagen del plato una vez realizado, para así, proporcionarle una idealización de lo que podría realizar siguiendo los pasos y conseguir que comience su elaboración.
- Se ha incluido la gg

2.1.4 Pruebas

Una vez completadas las fases de análisis, diseño e implementación, llega la hora de comprobar que la solución propuesta se adapta al público objetivo. Por este motivo, se diseñó el test con usuarios que se definirá en los siguientes subapartados, con el que se tratará de encontrar puntos fuertes y carencias de la interfaz generada, de forma que en las distintas iteraciones del proceso DCU puedan implementarse mejoras que subsanen o suavicen las carencias y/o potencien los puntos fuertes.

2.1.4.1 Test con usuarios

Este método permite mediante una interacción directa de los potenciales usuarios finales con la aplicación, medir la usabilidad de la aplicación, obteniendo medidas objetivas y subjetivas de niveles de eficacia, eficiencia, facilidad de uso o satisfacción de usuario.

2.1.4.1.1 Documentos necesarios

Para conseguir los objetivos expuestos anteriormente, fue necesaria la elaboración de los documentos que a continuación se muestran:

- **Documento de *screening***: El objetivo de este documento es diferenciar y filtrar aquellos usuarios que serán interés de evaluación. Mediante unas breves preguntas, se conocerá su situación personal y sus hábitos, permitiendo

posteriormente aceptar o no su participación en el test. En el caso de la aplicación aquí expuesta, se considerara válido aquel usuario que cumpla con los siguientes criterios:

- Cualquier edad a partir de dieciocho años.
- Que acuda/ haya acudido a aplicaciones u otros en busca de recetas.
- Que sea capaz de comunicarse en español (único idioma de la aplicación).
- Alto consumo de comida casera.

- **Documento pre-test**: Documento cuyo objetivo es conocer datos sociodemográficos del usuario que realizará el test. Con estos datos, se podrán categorizar los usuarios en distintos grupos de forma que se facilite el posterior análisis.

- **Documento de escenario:** Documento que será entregado al usuario al inicio del test y donde se detallará un hipotético escenario con varias tareas. Es importante que en éste, se aporte un alto nivel de detalle de la situación ficticia para así facilitar al usuario su puesta en contexto pues en definitiva, este contexto de uso es agente diferenciador a la hora del uso de la aplicación. En este documento, las distintas tareas cubren la totalidad de flujos posibles, permitiendo bajo un único test, evaluar todas las posibilidades de la aplicación. Además de este documento, se generaron los distintos formularios a usar por el evaluador de la prueba a la hora de obtener información de la misma.

- **Documento post-test**: Cuestionario entregado al usuario tras la ejecución del test con la finalidad de obtener las impresiones de éste acerca del desarrollo del test y extraer su opinión sobre el elemento analizado en el test.

- **Otros documentos**: Aunque los documentos principales de un test con usuarios son los detallados anteriormente, con el fin de asegurar un marco legal estable y seguro de cara a la ejecución del test, se hace necesario disponer de los siguientes documentos:
  - Documento de confidencialidad: Documento legal dirigido a mantener la confidencialidad del elemento evaluado, evitando filtraciones a terceras partes (por ejemplo, rivales de negocio).
  - Documento de consentimiento: Documento legal cuyo fin es obtener el consentimiento explícito firmado por el usuario para ceder los datos generados por la interacción con el elemento sometido a test (multimedia, documentos, etc.) bajo las condiciones indicadas en el mismo.

Estos documentos fueron generados y se encuentran anexados al presente documento en el Anexo I – Documentos de Evaluación.

2.1.4.1.2 Desarrollo

El proceso para la puesta en ejecución de un test con usuarios consta de las siguientes fases:

1. Elaboración de la documentación necesaria: Para la puesta en funcionamiento del test, será necesario elaborar los siguientes documentos: Documento de screening, acuerdo de confidencialidad, documento de consentimiento, documento pre-test, documento de escenario y documento post-test.

2. Obtención de candidatos: Fase dirigida al filtrado y selección de usuarios objetivo. Esta labor, es facilitada mediante el documento de *screening*.
3. Adecuación del entorno: Para el test, se preparará un entorno lo más amigable posible, de forma que el usuario esté en un estado cercano al escenario de uso de la aplicación. Además, se dispondrán los medios necesarios para la grabación del mismo, permitiendo la recogida de, por ejemplo, expresiones faciales o sensaciones.
4. Firma de responsabilidades: En esta fase se solicitará al usuario la firma del acuerdo de confidencialidad y del consentimiento de cesión de datos personales.
5. Adecuación pre-test: Primera parte del test. En este punto se tranquiliza al usuario con una conversación amigable para posteriormente proporcionarle el documento pre-test con lo que se obtendrá información relativa al usuario y relativa a la interacción de este con artefactos parecidos al que es objeto de evaluación.
6. Test: En esta fase se desarrolla el test. En primer lugar, se le proporciona al usuario el documento de escenario, donde se expone una situación con el suficiente nivel de detalle para que el usuario ejecute ciertas tareas con el artefacto mientras es supervisado por un evaluador. La función de éste último, será tomar nota de todo lo considerado relevante del test y guiar al usuario a través del test, tratando de fomentar en la medida de lo posible, el pensamiento manifiesto.
7. Post-test: Una vez finalizado el test, se pedirá al usuario rellenar un breve formulario acerca del desarrollo del test, para posteriormente agradecerle su participación y dejarle marchar.

2.1.4.1.3 Resultados

Una vez el producto estuvo en una fase avanzada de desarrollo que permitiera poner en valor todas las características de éste y, donde las carencias de falta de desarrollo no fueran determinantes en la evaluación del mismo, se procedió a ejecutar el test con usuarios mediante el protocolo comentado en los apartados anteriores, obteniendo los resultados que se expondrán a continuación. Así mismo, los documentos completados con la información, pueden visualizarse en el Anexo II – Documentos rellenos tras test de usuarios.

2.1.4.1.3.1 Contexto del test

Para una mejor comprensión del test, será necesario contextualizarlo adecuadamente, de forma que la evaluación posterior pueda realizarse consecuentemente. Así, para este caso, se ha ejecutado el test bajo el siguiente contexto:

- La usuaria entra dentro del público objetivo de la aplicación.
- La evaluación se realizó en dos estancias amigables, salón y cocina del hogar, adecuando éstas al proceso de test.
- Se realizó de forma seguida, tratando de estimular al usuario para contextualizarlo dentro de cada tarea.
- El dispositivo usado para la evaluación fue un smarthpone, modelo Samsung Galaxy J6.

En cuanto al participante, mediante los documentos de screening y pre-test se deduce lo siguiente:

- Joven, sin cargas familiares, soltera, con pareja y estudios universitarios (Ingeniería).
- Usa bastante el móvil (más de 4 horas diarias), mayoritariamente para interactuar en redes sociales y considera tener un alto nivel de manejo sobre el mismo.
- Habla español de forma nativa.
- Trata de cuidar su alimentación e innovar, siendo sus fuentes de conocimiento Youtube y aplicaciones móviles (como Tasty o Instagram).
- Alto consumo de comida casera.
- Destaca la mala explicación y rapidez de ejecución (obligando a pausas manuales) de las recetas de internet.

Así, y según lo expuesto en el apartado 2.1.1.2 Perfilado de usuarios se trata de un usuario con perfil simplista, aunque con matices de perfile curioso.

2.1.4.1.3.2 Ejecución

Ilustración 28. Fotografía de la ejecución del Test (Elaboración pròpia)

La ejecución del test transcurrió sin incidentes destacados, salvo que aunque en la tarea 1 se especificaba la búsqueda de la receta "Potaje de habichuelas", ésta no constaba entre el listado incluido, por lo que se cambió por "Taquitos de cerdo en salsa de almendra".

En cuanto al aspecto propio del test, **el usuario fue capaz de completar todas las tareas**, obteniendo unos tiempos de ejecución aceptables.

2.1.4.1.3.3 Puntos destacados por el usuario

Una vez fue realizado el test, se le hizo rellenar al usuario el cuestionario post-test, del que se deducen los siguientes puntos:

- El nivel de satisfacción general lo sitúa en 4.
- Destaca la ausencia de anuncios, la rapidez, sencillez, intuitividad y diseño de la aplicación.
- Confusión a la hora de filtrar tras desaparecer el botón de acceso a filtros debido a fallo de conexión momentáneo.

- Duda a la hora de conocer cómo establecer los filtros.
- Molestia al producirse el apagado de la pantalla en la ejecución de la receta.
- Desconocimiento de la cantidad de raciones generadas a raíz de la cantidad de ingredientes mostrada.
- Algo de nerviosismo en el usuario debido a la evaluación.

2.1.4.1.3.4 Puntos detectados por el evaluador

De igual forma que en el punto anterior, tras el test, se extrajeron los siguientes puntos destacados desde la figura de evaluador:

- No se expresa correctamente el fin de la búsqueda cuando no existen recetas con los requisitos especificados.
- Desconoce si las cantidades especificadas de ingredientes son para una o dos raciones.
- La opción filtrar es mejorable, al no esclarecerse que hacer una vez seleccionados los filtros.
- En caso de fallo de red, el botón de acceso a filtros desaparece desconcertando al usuario.
- No se informa al usuario de la opción de usar la voz para pasar al siguiente o anterior paso de la receta.
- El apagado de pantalla durante la ejecución de la receta, anula la ventaja de uso de voz.
- El menú inferior de navegación, cuando la opción no está seleccionada, no dispone del suficiente contraste obligando a fijar la visión en este campo.

2.1.4.1.3.5 Mejoras implementadas

Finalmente, mediante la naturaleza iterativa de DCU y una vez se analizaron los puntos expuestos en los dos apartados anteriores, se establecieron las mejoras expuestas a continuación:

- Se añadió un texto en la interfaz de búsqueda de forma que cuando no hubiera recetas de acuerdo a los criterios establecidos, indicara este hecho al usuario de forma clara e inequívoca.
- El botón de acceso a filtros ahora no desaparece en caso de fallo de red, sino que al pulsar sobre él, muestra una notificación indicando que hubo un problema de red y se reintente más tarde.
- Se añade un botón con el texto "Filtrar" en la interfaz de filtros para que el usuario tenga claro como proceder para filtrar.
- Se configura la interfaz de paso a paso para mantener la pantalla encendida mientras se elabora la receta.
- Se añade un texto "Para dos personas:" en los ingredientes para que el usuario conozca la cantidad de raciones que se elaborarán.
- Se añade un diálogo la primera vez que se accede al paso a paso con doble utilidad: La primera, informar de la posibilidad de uso de voz para moverse entre los pasos y la segunda, pedir consentimiento al usuario para poder activar el micrófono.
- Se cambia el color de la opción no seleccionada en el menú inferior para que el contraste sea mayor.

Por último, aunque no fue detectada esta necesidad durante el test, durante la evaluación de la segunda entrega parcial, el tutor Jordi Almirall indicó la necesidad de establecer en la interfaz de inicio de sesión la opción de "Recordar contraseña" para que la sesión quedara iniciada tras un reinicio de la aplicación. Dados los altos conocimientos en la materia de este tutor, ésto, podría considerarse una evaluación sin usuarios a cargo de un experto, por lo que se tomó en cuenta como mejora.

Así, tras implementar estas mejoras, la interfaz modificada quedó como se muestra a continuación:

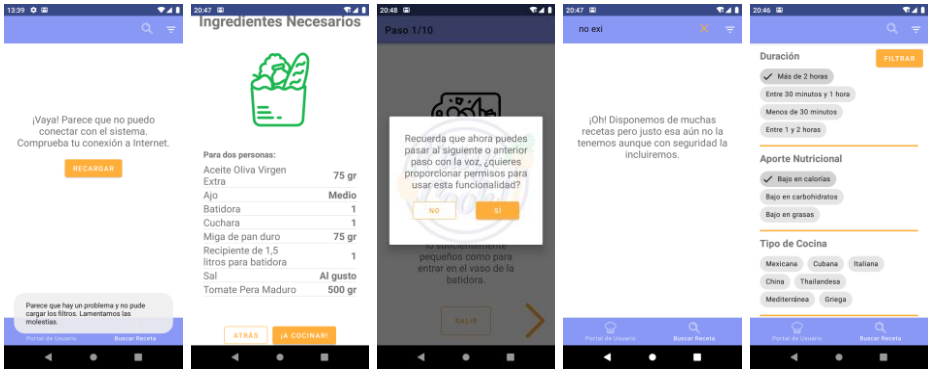


Ilustración 29. Interfaz modificada tras DCU (Elaboración propia)

2.2 Diseño técnico

Una vez finalizado el proceso DCU y tras haber extraído, analizado y conceptualizado toda la información relativa al diseño de la interfaz, se comenzó con el diseño técnico de la solución.

Durante esta fase, se siguió un proceso parecido al del diseño gráfico, partiendo de información y definiciones algo más abstractas (diagramas de casos de uso) hacia conceptualizaciones concretas de la solución (arquitectura lógica, física y de datos) pasando por puntos intermedios de definición y concreción como es la definición de casos de uso. A continuación, se presenta tanto este proceso como los resultados obtenidos.

2.2.1 Diagramas UML

Como se comenta anteriormente, el primer paso para concretar un diseño adecuado para la solución será la "destilación" de los distintos requisitos funcionales presentados en el apartado 1.2 Objetivos del Trabajo para la obtención de los diagramas de casos de uso. Un diagrama de caso de uso consiste en una representación en lenguaje unificado de modelado (UML) de la funcionalidad del sistema a modelar y su interacción con los usuarios principales.

Así, en primer lugar se definieron a los usuarios que harán uso del sistema o actores, obteniéndose lo siguiente:

Identificador:	ACT001 - Usuario
Descripción:	Dentro de este actor se englobarán los tres perfiles propuestos en el apartado <<<hiper a los perfiles>>>>, es decir, cualquier usuario de la aplicación.

Ilustración 30. Actor de la aplicación
(Elaboración propia)

El motivo que lleva a la **decisión de incluir un único actor** es que aunque dependiendo del perfil el usuario tendrá predisposición al uso de ciertas acciones o cambiará la forma de usarlas, **el nivel de abstracción actual no requiere de este tipo de detalles**. Por otro lado, la ausencia de niveles de subscripción o roles (que permitan el acceso a ciertas características reservadas) evita la necesidad de definir actores diferentes.

Una vez establecido el actor, se procedió al empaquetado de los distintos diagramas de casos de uso, es decir, la agrupación de los distintos diagramas según su finalidad y/o función. Así, se obtuvieron los siguientes paquetes:

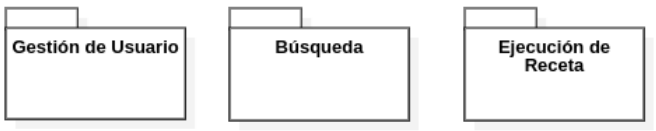


Ilustración 31. Paquetes de casos de uso
(Elaboración propia)

Estos tres paquetes incluirán en su interior casos de uso de acuerdo a las siguientes especificaciones:

- Paquete "Gestión de usuario": Casos de uso relacionados con el ciclo de vida del usuario dentro de la aplicación, es decir, registro de usuario, abrir sesión, etc.
- Paquete "Búsqueda": Casos de uso vinculados al objetivo de encontrar una receta a realizar.
- Paquete "Ejecución de receta": Casos de uso relacionados con el proceso de inicio, realización y fin de una receta.

Aunque el disponer de únicamente un actor en este sistema provoca que la relación entre paquetes y actores sea obvia, a continuación, puede observarse esta relación:

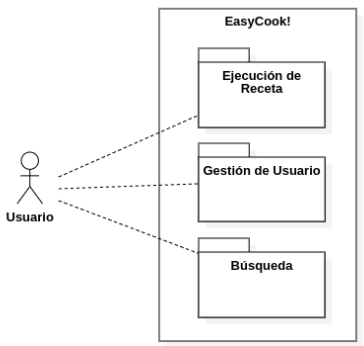


Ilustración 32. Relación actor-paquetes
(Elaboración propia)

Seguidamente, se muestran los distintos casos de uso incluidos en cada paquete:

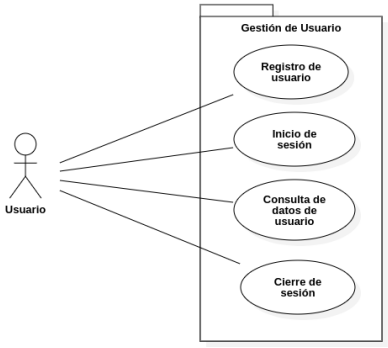


Ilustración 33. Paquete “Gestión de Usuario”
(Elaboración propia)

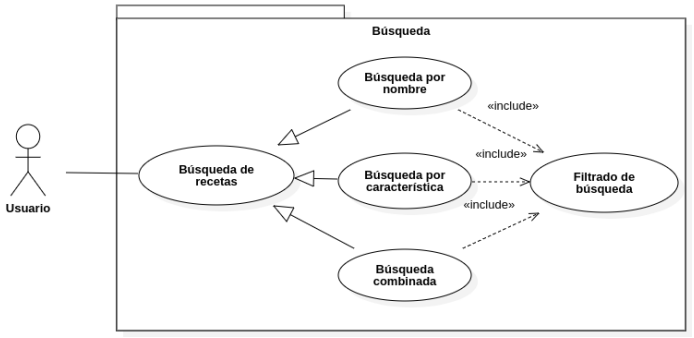


Ilustración 34. Paquete "Búsqueda de recetas"
(Elaboración propia)

En este paquete, como se puede observar en la imagen anterior, se hace uso de generalizaciones e inclusiones. Las primeras, son utilizadas para la concreción en las distintas posibilidades del caso de uso abstracto "Búsqueda de recetas". Respecto a la segunda, su uso provoca la inclusión de las distintas especificaciones de búsqueda dentro del filtrado de recetas, ya que, se tendrán en cuenta a la hora de exponer los resultados.

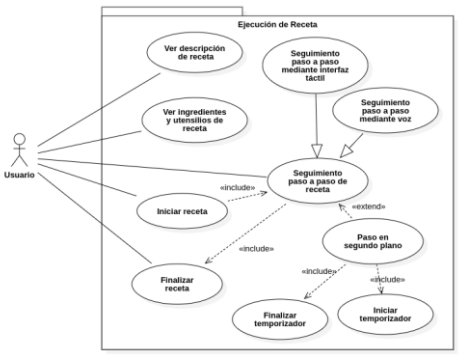


Ilustración 35. Paquete "Ejecución de receta"
(Elaboración propia)

Finalmente en este paquete se observan igualmente relaciones de generalización, extensión e inclusión. En este caso, las primeras se usan para expresar la especificación del caso de uso de seguimiento paso a paso de una receta

en el seguimiento paso a paso mediante interfaz vocal mientras que la de extensión es usada para reflejar un caso particular del seguimiento paso a paso que es un paso en segundo plano. Por último, las de inclusión son usadas para expresar el uso iniciar receta del seguimiento paso a paso de la misma, y este a su vez hace uso de la finalización de la receta y de la gestión de paso en segundo plano, además de la finalización de la receta.

2.2.2 Casos de uso

Así, para finalizar este apartado y tras la definición de cada uno de los casos de uso, a continuación pueden observarse la descripción detallada de los casos de uso expuestos en los diagramas anteriores:

CU001	Registro de usuario
Prioridad:	Baja
Descripción:	Se permite el registro de usuarios para la creación de una cuenta en el sistema que permita generar datos asignados al usuario, para ofrecerle ciertos servicios a éste.
Actores:	Usuario
Precondiciones:	- Que el email usado no esté ya registrado - Que la contraseña a usar cumpla ciertos requisitos de seguridad - Que no haya sesión activa de otro usuario en el mismo dispositivo
Iniciado por:	Usuario
Flujo:	1. Iniciar la aplicación 1.1. En caso de interés del negocio, se muestra un *banner* con información para el usuario. 2. Acceder al Panel de Usuario 3. Entrar en la opción registro de usuario 4. Rellenar los datos de usuario (Nombre, apellidos, email, contraseña, repetición de contraseña y fecha de nacimiento) 5. Pulsar sobre el botón “Registrar” 5.1. Si faltan datos obligatorios y/o éstos son incorrectos, se muestra error explícita y detalladamente. 6. El *back-end* de la aplicación recibe la petición y se encarga de ejecutar aquellas comprobaciones que no fueron posibles en el lado del cliente (por ejemplo, que el email no había sido usado con anterioridad). 6.1 Si existen datos incorrectos, se responde con un error e información del error que carga la aplicación. 7. El *back-end* ejecuta las sentencias necesarias para persistir los datos de registro y responde con un *OK*. 8. La aplicación muestra aviso de registro satisfactorio y realiza auto-logon del usuario. Mostrando la información del perfil.
Postcondiciones:	- Nuevo usuario registrado en el sistema - El usuario está logueado
Notas:	El Registro de usuario es opcional en la aplicación debido al rechazo detectado en la fase de requisitos.

Ilustración 36. Caso de uso CU001
(Elaboración propia)

CU002	Inicio de sesión
Prioridad:	Baja
Descripción:	El usuario inicia sesión en la aplicación mediante unas credenciales conocidas.
Actores:	Usuario
Precondiciones:	- El actor ha realizado en algún momento el caso de uso CU001 - Se dispone de las credenciales de acceso
Iniciado por:	Usuario
Flujo:	1. Iniciar la aplicación 1.1. En caso de interés del negocio, se muestra un *banner* con información para el usuario. 2. Acceder al Panel de Usuario 3. Entrar en la opción “Inicio de Sesión” 4. Introducir las credenciales de acceso (email y contraseña) 5. El usuario pulsa el botón “Iniciar Sesión” 5.1 En caso de que falte alguna credencial, se hace visible al usuario. 6. Se envía petición al *back-end* el cual comprueba que éstas coinciden con las almacenadas en base de datos. 6.1 Si no coincidieran, se devolvería un error con información que se procesaría y mostraría por la aplicación. 7. Se responde a la petición con *OK* e información del usuario la cual es mostrada por la aplicación. 8. La aplicación muestra el perfil del usuario
Postcondiciones:	- El usuario está logueado
Notas:	El inicio de sesión de usuario es opcional debido al rechazo detectado en la fase de requisitos.

Ilustración 37. Caso de uso CU002
(Elaboración propia)

CU003	Cierre de sesión
Prioridad:	Baja
Descripción:	El usuario inicia sesión en la aplicación mediante unas credenciales conocidas.
Actores:	Usuario
Precondiciones:	- El actor ha realizado en algún momento el caso de uso CU001 - Se dispone de las credenciales de acceso
Iniciado por:	Usuario
Flujo:	1. Iniciar la aplicación 1.1. En caso de interés del negocio, se muestra un *banner* con información para el usuario. 2. Acceder al Panel de Usuario 3. Entrar en la opción “Inicio de Sesión” 4. Introducir las credenciales de acceso (email y contraseña) 5. El usuario pulsa el botón “Iniciar Sesión” 5.1 En caso de que falte alguna credencial, se hace visible al usuario. 6. Se envía petición al *back-end* el cual comprueba que éstas coinciden con las almacenadas en base de datos. 6.1 Si no coincidieran, se devolvería un error con información que se procesaría y mostraría por la aplicación. 7. Se responde a la petición con *OK* e información del usuario la cual es mostrada por la aplicación. 8. La aplicación muestra el perfil del usuario
Postcondiciones:	- El usuario está logueado
Notas:	El inicio de sesión de usuario es opcional debido al rechazo detectado en la fase de requisitos.

Ilustración 38. Caso de uso CU003
(Elaboración propia)

CU004	Consulta de datos de usuario
Prioridad:	Baja
Descripción:	El usuario desea consultar alguna información de su interés relacionada con su perfil
Actores:	Usuario
Precondiciones:	- El actor ha realizado en algún momento el caso de uso CU002 - Su sesión no ha caducado
Iniciado por:	Usuario
Flujo:	1. Iniciar la aplicación 1.1. En caso de interés del negocio, se muestra un *banner* con información para el usuario. 2. Acceder al Panel de Usuario 3. Visualizar los datos
Postcondiciones:	-
Notas:	Este panel solo estará visible si el usuario está logueado con una sesión válida, en caso contrario, se muestran las opciones de iniciar sesión y registro.

Ilustración 39. Caso de uso CU004
(Elaboración propia)

CU005	Búsqueda de recetas
Prioridad:	Alta
Descripción:	El usuario desea buscar una receta en base a un nombre y/o característica/s concreta/s.
Actores:	Usuario
Precondiciones:	- El usuario debe conocer el nombre y/o característica/s de la receta que desea buscar. - La receta debe estar en el catálogo de la aplicación
Iniciado por:	Usuario
Flujo:	1. Iniciar la aplicación 1.1. En caso de interés del negocio, se muestra un *banner* con información para el usuario. 2. Se muestra un listado de recetas al azar. 3. En función de cómo desea realizar la búsqueda: 3.1 Si el usuario desea buscar por nombre, realiza el caso de uso CU006. 3.2 Si el usuario desea buscar en base a unas características determinadas, realiza el caso de uso CU007. 3.3 Si el usuario desea combinar la búsqueda por nombre y unas características concretas, realiza el caso de uso CU008. 4. Se pulsa sobre el botón buscar. 5. Se hace la petición a back-end, el cual se encarga de encontrar las recetas que cumplen los requisitos fijados. 6. Se muestran los resultados.
Postcondiciones:	- El usuario encuentra las recetas que se ajustan a sus requisitos.
Notas:	- Por defecto, el panel mostrado es el de búsquedas, para acelerar en la medida de lo posible este proceso.

Ilustración 40. Caso de uso CU005
(Elaboración propia)

CU006	Búsqueda por nombre
Prioridad:	Alta
Descripción:	El usuario desea buscar una receta concreta mediante su nombre.
Actores:	Usuario
Precondiciones:	- El usuario debe conocer el nombre de la receta. - La receta debe estar en el catálogo de la aplicación. - Haber iniciado el caso de uso CU005.
Iniciado por:	Usuario
Flujo:	1. Se introduce el nombre de la receta en el buscador. 2. Se ejecuta el caso de uso CU009.
Postcondiciones:	- El usuario encuentra la receta que deseaba - El usuario ve la información de la receta
Notas:	- Por defecto, el panel mostrado es el de búsquedas, para acelerar en la medida de lo posible este proceso.

Ilustración 41. Caso de uso CU006
(Elaboración propia)

CU007	Búsqueda por característica
Prioridad:	Media
Descripción:	El usuario desea buscar recetas que se ajusten a unas características determinadas (duración, dificultad, tipo y/o valor nutricional).
Actores:	Usuario
Precondiciones:	- El usuario debe conocer por qué característica/s quiere filtrar. - Debe haber recetas que cumplan el criterio. - Haber iniciado el caso de uso CU005.
Iniciado por:	Usuario
Flujo:	1. Pulsar el botón de filtro y se selecciona el/los filtro/s que sea/n de su interés. 3. Se pulsa el botón “Aceptar”. 4. Se ejecuta el caso de uso CU009
Postcondiciones:	- El usuario encuentra recetas en base a su criterio
Notas:	- Por defecto, el panel mostrado es el de búsquedas, para acelerar en la medida de lo posible este proceso.

Ilustración 42. Caso de uso CU007
(Elaboración propia)

CU008	Búsqueda combinada
Prioridad:	Baja
Descripción:	El usuario desea buscar recetas que se ajusten a una combinación de los casos de uso CU006 y CU007.
Actores:	Usuario
Precondiciones:	- El usuario debe conocer los criterios a aplicar. - Debe haber recetas que cumplan el criterio. - Haber iniciado el caso de uso CU005.
Iniciado por:	Usuario
Flujo:	1. Se realiza el caso de uso CU006 2. Se realiza el caso de uso CU007 3. Se ejecuta el caso de uso CU008.
Postcondiciones:	- El usuario encuentra recetas en base a su criterio
Notas:	- Por defecto, el panel mostrado es el de búsquedas, para acelerar en la medida de lo posible este proceso.

Ilustración 43. Caso de uso CU008
(Elaboración propia)

CU009	Filtrado de búsqueda
Prioridad:	Alta
Descripción:	El sistema debe establecer los filtros que el usuario necesita para encontrar una receta.
Actores:	Usuario
Precondiciones:	- Haber ejecutado el CU006, CU007 o CU008
Iniciado por:	Usuario
Flujo:	1. Se aplican los filtros deseados por el usuario 2. Se muestran las recetas que cumplan con los filtros establecidos.
Postcondiciones:	- Se muestran recetas que cumplen con los requisitos establecidos.
Notas:	

Ilustración 44. Caso de uso CU009
(Elaboración propia)

CU010	Ver descripción de receta
Prioridad:	Alta
Descripción:	El usuario desea visualizar la información básica de una receta como puede ser la descripción, la imagen, las características de la misma y el aporte nutricional.
Actores:	Usuario
Precondiciones:	- [OPC] Haber realizado el caso de uso CU005. - Haber elegido una receta de su interés de las posibles.
Iniciado por:	Usuario
Flujo:	1. El usuario pulsará en la receta de su interés. 2. Se mostrará la información obtenida mediante la petición a *back-end.*
Postcondiciones:	- El usuario podrá ver la información de la receta.
Notas:	

Ilustración 45. Caso de uso CU010
(Elaboración propia)

CU011	Ver ingredientes y utensilios de receta
Prioridad:	Alta
Descripción:	El usuario desea conocer los ingredientes y/o utensilios de una receta determinada
Actores:	Usuario
Precondiciones:	- [OPC] Haber realizado el caso de uso CU005 - [OPC] Haber ejecutado el caso de uso CU010
Iniciado por:	Usuario
Flujo:	1. El usuario puede acceder a esta opción por dos opciones: 1.1 Haber realizado el caso de uso CU010 y pulsar sobre el botón ver ingredientes. 1.2 Desde el menú principal, tras ejecutar el caso de uso CU005 o no, picar en el botón de ver Ingredientes de una receta concreta. 2. Se muestran los ingredientes de la receta concreta al usuario.
Postcondiciones:	- Se pone a disposición del usuario los ingredientes concretos de una receta.
Notas:	

Ilustración 46. Caso de uso CU011
(Elaboración propia)

CU012	Iniciar receta
Prioridad:	Alta
Descripción:	Comienzo de la receta por parte del usuario
Actores:	Usuario
Precondiciones:	- Haber realizado el caso de uso CU010 o CU011
Iniciado por:	Usuario
Flujo:	1. El usuario pulsa sobre el botón de iniciar receta 2. Se muestra el primer paso de la receta
Postcondiciones:	- El usuario se encuentra en el primer paso de la receta
Notas:	

Ilustración 47. Caso e uso CU012
(Elaboración propia)

CU013	Seguimiento paso a paso de receta
Prioridad:	Alta
Descripción:	El usuario sigue paso a paso una receta, pudiendo retroceder o continuar en la misma.
Actores:	Usuario
Precondiciones:	- Haber realizado el caso de uso CU012
Iniciado por:	Usuario
Flujo:	1. El usuario ejecuta el caso de uso CU014, CU015 o CU019: 1.1 En caso de acción de continuar (CU014 ò CU015): 1.1.1 Si es un paso intermedio, se lleva al siguiente 1.1.2 Si es el último paso, se comienza el caso de uso CU019 1.2 Si se efectúa acción de retroceso (CU014 ò CU015): 1.2.1 Si es el primer paso, se ejecuta el caso de uso CU019 1.2.2 Si es otro paso, se vuelve al anterior. 1.3 Se pulsa sobre el botón salir, y se dirige al usuario al caso de uso CU0019
Postcondiciones:	El usuario se encuentra en otro paso o interfaz.
Notas:	

Ilustración 48. Caso de uso CU013
(Elaboración propia)

CU014	Seguimiento paso a paso mediante interfaz táctil
Prioridad:	Alta
Descripción:	El usuario decide usar la pantalla de su Smartphone para realizar el caso de uso CU013.
Actores:	Usuario
Precondiciones:	- Estar dentro del caso de uso CU013
Iniciado por:	Usuario
Flujo:	1. El usuario pulsa el botón avanzar o retroceder mostrados en la pantalla.
Postcondiciones:	El usuario ha cambiado de paso.
Notas:	

Ilustración 49. Caso de uso CU014
(Elaboración propia)

CU015	Seguimiento paso a paso mediante voz
Prioridad:	Media
Descripción:	El usuario decide usar la voz para realizar el caso de uso CU013.
Actores:	Usuario
Precondiciones:	- Estar dentro del caso de uso CU013
Iniciado por:	Usuario
Flujo:	1. El usuario dice en voz alta y clara “Siguiente” o “Atrás”.
Postcondiciones:	El usuario ha cambiado de paso.
Notas:	

Ilustración 50. Caso de uso CU015
(Elaboración propia)

CU016	Paso en segundo plano
Prioridad:	Media
Descripción:	El usuario se encuentra en un paso que puede ejecutarse mientras se realizan otros.
Actores:	Usuario
Precondiciones:	- Estar ejecutando el caso de uso CU013 y estar en un paso paralelizable o bien haber ejecutado con anterioridad el caso de uso CU017.
Iniciado por:	Usuario
Flujo:	1. En caso de que el usuario se encuentre aquí por estar en el caso de uso CU014: 1.1 El usuario finaliza el paso a falta de esperar cierto tiempo 1.2 El usuario realiza el caso de uso CU017 2. En caso de que haya ejecutado el caso de uso CU017 anteriormente: 2.1 El usuario es avisado del fin del paso paralelizable 2.2 El usuario ejecuta el caso de uso CU019.
Postcondiciones:	El usuario inicia o finaliza tarea paralelizable.
Notas:	

Ilustración 51. Caso de uso CU016
(Elaboración propia)

CU017	Iniciar temporizador
Prioridad:	Media
Descripción:	El usuario ha finalizado un paso en el que sólo debería esperar un determinado tiempo por lo que desea ser avisado.
Actores:	Usuario
Precondiciones:	- Haber realizado el caso de uso CU016
Iniciado por:	Usuario
Flujo:	1. El usuario finaliza el paso e intenta avanzar al siguiente paso. 2. Se muestra ventana emergente indicando si desea establecer un temporizador 3. El usuario pulsa aceptar.
Postcondiciones:	- Temporizador establecido.
Notas:	Para implementar este caso de uso, será obligatorio implementar con posterioridad el caso de uso CU019

Ilustración 52. Caso de uso CU017
(Elaboración propia)

CU018	Finalización de temporizador
Prioridad:	Media
Descripción:	Un paso paralelizable iniciado con anterioridad ha finalizado y se debe avisar al usuario.
Actores:	Usuario
Precondiciones:	- Haber realizado el caso de uso CU017 - Haber finalizado el tiempo de la tarea en segundo plano.
Iniciado por:	Usuario
Flujo:	1. Se avisa al usuario mediante ventana emergente y efecto sonoro de la finalización. 2. El usuario pulsa sobre aceptar.
Postcondiciones:	- Temporizador finalizado.
Notas:	Para implementar este caso de uso, será obligatorio implementar con anterioridad el caso de uso CU018

Ilustración 53. Caso de uso CU018
(Elaboración propia)

CU019	Finalizar receta
Prioridad:	Alta
Descripción:	El usuario finaliza la receta que estaba ejecutando bien al completarla o bien porque no desea continuar.
Actores:	Usuario
Precondiciones:	- Haber ejecutado el caso de uso CU013
Iniciado por:	Usuario
Flujo:	1. El usuario pulsa sobre el botón salir o bien sobre el botón continuar encontrándose en el último paso. 2. Se muestra un mensaje en ventana emergente 3. El usuario pulsa sobre el botón aceptar y se le dirige al menú principal.
Postcondiciones:	- El usuario ha salido de la receta.
Notas:	

Ilustración 54. Caso de uso CU019
(Elaboración propia)

2.2.3 Arquitectura

Una vez se han aclarado y establecido los distintos casos de uso de la solución es el momento de comenzar a diseñar la arquitectura de la que dispondrá la aplicación. La arquitectura de un sistema como el expuesto en este trabajo consta de tres planos distintos (físico, lógico y datos), por lo que para facilitar el detalle del mismo, se ha optado por separar cada uno de ellos en un subapartado distinto como se observará a continuación.

2.2.3.1 Arquitectura física

En ésta se detallan los distintos sistemas y sus conexiones, es decir, cómo está construido el sistema en el mundo real. Así, el sistema aquí expuesto consta de la siguiente arquitectura física:

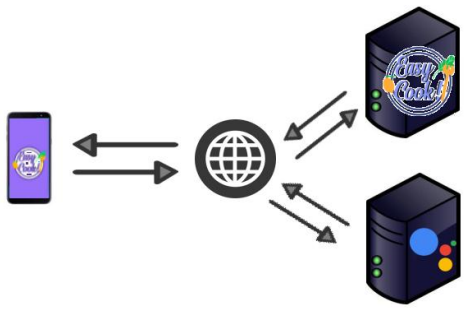


Ilustración 55. Arqutitectura física
(Elaboración propia)

Como se puede observar, el sistema se basa en un modelo **cliente-servidor,** constando de tres subsistemas:

- Aplicación Android (*front-end*): Aplicación desarrollada para Android nativo (Java) a través de la que el usuario interaccionará con el sistema.
- Servidor Específico (*back-end*): Servidor específico donde se almacenará el back-end de la aplicación, incluyendo tanto el API como la base de datos.
- Servidor Google Assistant (*back-end*): Servidor de la API de Google Assistant que transcribirá el sonido de la voz del usuario a texto.

2.2.3.2 Arquitectura lógica

La arquitectura lógica de un sistema hace referencia a cómo éste se ordena, codifica o agrupa desde el punto de vista del código. Así, existen varios patrones de diseño distintos cuyo objetivo es tratar de obtener el máximo rendimiento (tanto en términos de servicio como de codificación) a la hora de realizar un software.

Dado que en el presente proyecto existen dos subsistemas (servidor y aplicación), se ha decidido separar el detalle de cada uno. Así, en el caso del servidor *back-end* se ha optado por seguir las directrices de lo que a futuro podría ser una **arquitectura basada en microservicios**. Este tipo de arquitectura destaca por la independencia, acotación y "simpleza" de cada componente, de forma, que cada componente tiene una y solo una función (en términos generales). No obstante, dado que el objetivo principal de este trabajo consistía en el desarrollo de una aplicación móvil, se ha optado por unificar los distintos *endpoints* de la interfaz de programación de aplicaciones (API) bajo un único ejecutable Python, siempre, como se ha comentado anteriormente, manteniendo la independencia, simpleza y acotación. Respecto a la persistencia de datos, por un lado, para los recursos dinámicos (usuarios, recetas, filtros, eventos, etc.) se proporciona la base de datos presentada en el apartado 2.2.3.3 Arquitectura de datos del presente documento. Finalmente, para los recursos estáticos (imágenes principalmente), se ha implementado un *endpoint* dentro del mismo API (pero con distinta nomenclatura) desde donde se servirán los mismos. A continuación, puede observarse lo descrito, de forma gráfica:

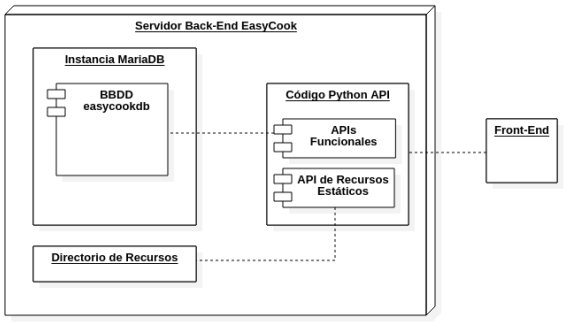


Ilustración 56. Arquitectura Lógica Back-end (Elaboración Propia)

En el caso del *front-end* (aplicación), y con el fin de respetar los requisitos no funcionales RNF003 y RNF009 establecidos para el proyecto, se optó por seguir un **patrón modelo-vista-controlador (MVC)**. Este tipo de arquitectura software destaca por la separación de los datos y su presentación de la lógica de la aplicación, definiendo para ello tres componentes:

- Modelo: Constituye la representación lógica de los datos necesarios para el funcionamiento de la aplicación. En este caso, estas representaciones servirán para almacenar los datos obtenidos mediante el API del servidor.
- Vista: Representación visual del modelo. En este proyecto la conformarán las distintas interfaces XML diseñadas.
- Controlador: Lógica de la aplicación encargada de la recepción de eventos procedentes de la interfaz, su gestión y modificación de ésta si fuera necesario. En la aplicación tratada en este documento, el controlador estará conformado por las distintas actividades, fragmentos y elementos controladores de visualizaciones (por ejemplo, *RecyclerViews*).

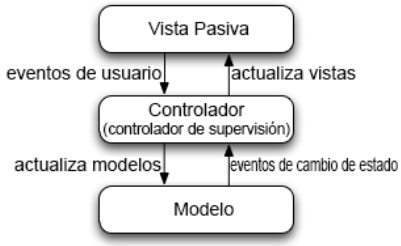


Ilustración 57. Patrón MVC
(Fuente: Wikipedia 2015)

La principal ventaja que ha llevado a la selección de este patrón es la sencillez que proporciona a la hora de elaborar nuevas representaciones del modelo, lo cual sin duda, junto al uso de fragmentos, será de gran utilidad en el futuro a la hora de su adaptación a tamaños de pantalla mayores (tal y como se establecía en el requisito no funcional antes mencionado).

Así, y de acuerdo a lo comentado anteriormente, se definió la siguiente arquitectura lógica de la aplicación:

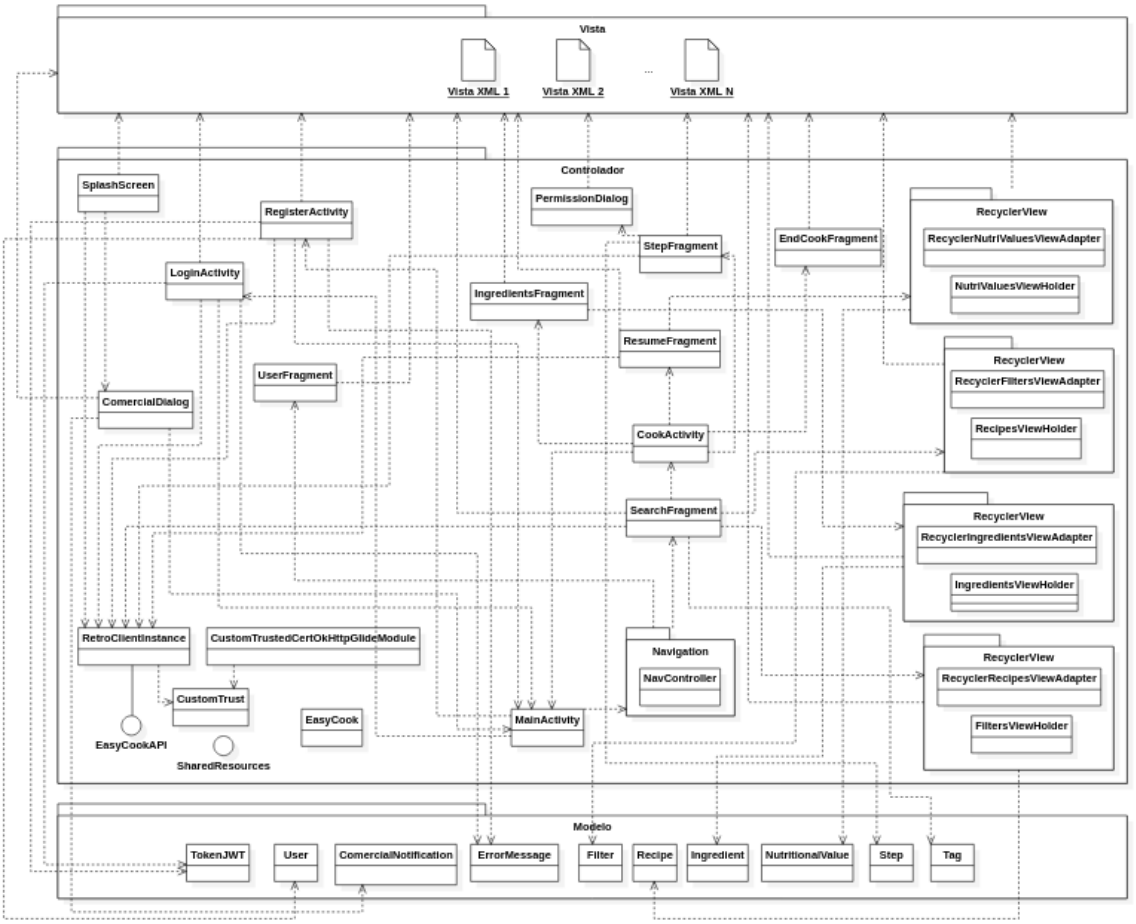


Ilustración 58. Arquitectura Lógica Front-end
(Elaboración Propia)

Como se observa, existen una interfaz, *SharedResources*, y una clase, *Easycook* que aunque no se indican relaciones, podrán relacionarse con cualquier elemento de los que aparecen en el diagrama pues están destinadas a proveer de recursos compartidos (el primer elemento) y de contexto de aplicación (el segundo).

2.2.3.3 Arquitectura de datos

Finalmente, en este punto se detallan como se estructurarán y almacenarán los datos que el sistema usará para su funcionamiento.

Dado que como se hacía mención en capítulos anteriores el sistema contará con un *back-end* encargado de ejecutar cierta parte de la lógica de la aplicación así como, persistir y/o almacenar ciertos datos. Es precisamente esta última cuestión la que provoca la necesidad de establecer una arquitectura de datos correcta que aporte la robustez y flexibilidad necesarias para una aplicación de este tipo. Así, tras evaluar y comprender los distintos casos de uso, se elaboró el siguiente diagrama lógico que soporte dichas acciones:

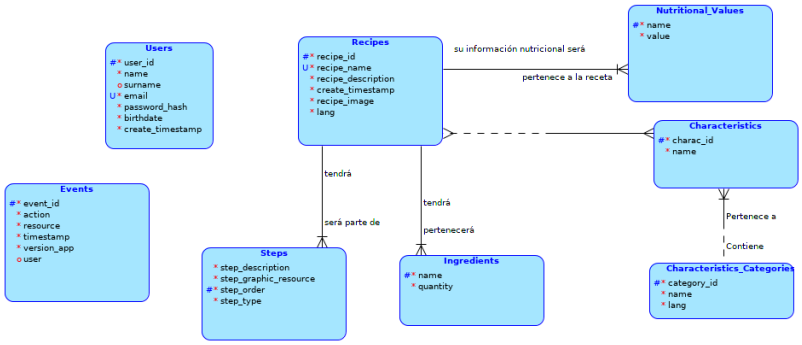


Ilustración 59. Diagrama lógico de base de datos
(Elaboración propia)

Este diagrama, a su vez, fue de utilidad para la creación del siguiente diagrama entidad-relación que será el que se usará en la base de datos de back-end de la aplicación:

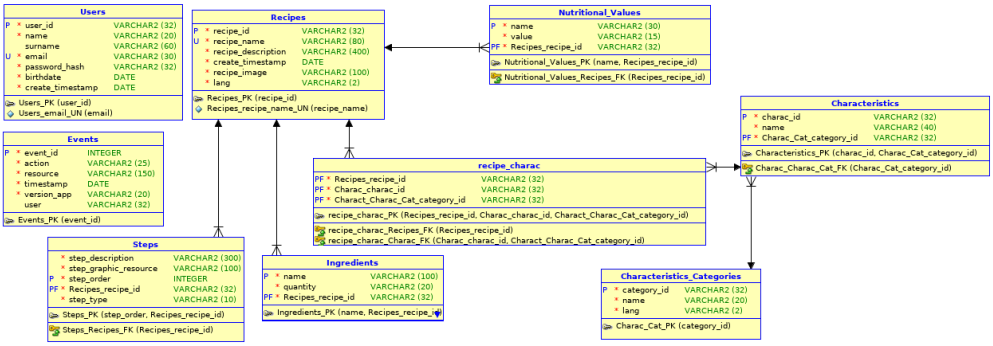


Ilustración 60. Diagrama entidad-relación de base de datos
(Elaboración propia)

Con el fin de aportar información extra sobre el diagrama, a continuación se mostrará y detallará cada clase del sistema junto con sus atributos:

- Entidad “Users”: Entidad que contendrá todos los usuarios así como, la información relativa a estos.

Users	
Atributo	**Descripción**
user_id	Identificador único de usuario (MD5 del email).
name	Nombre del usuario.
surname	Apellidos del usuario.
email	Email del usuario.
password_hash	Hash de contraseña del usuario. Así, en caso de filtración de información, se evita que los atacantes conozcan las contraseñas.
birthdate	Fecha de nacimiento del usuario.
create_timestamp	Marca de tiempo de creación del usuario.

Ilustración 61. Entidad Users
(Elaboración propia)

- Entidad “Events”: Usada para almacenar eventos fruto de la interacción del usuario con la aplicación para detectar errores y, a futuro, realizar analíticas de negocio.

Events	
Atributo	**Descripción**
event_id	Identificador único de evento.
action	Tipo de evento.
resource	Recurso sobre el que aplica.
timestamp	Marca de tiempo del evento.
Versión_app	Versión build de la aplicación con la que se realizó la acción.
user	ID de usuario que realizó la acción

Ilustración 62. Entidad Events
(Elaboración propia)

- Entidad “Recipes”: Entidad encargada de almacenar las recetas en el sistema.

Recipes	
Atributo	**Descripción**
recipe_id	Identificador único de receta (MD5 del nombre y el idioma).
recipe_name	Nombre de la receta.
recipe_description	Descripción de la receta.
create_timestamp	Marca de tiempo de creación de la receta.
recipe_image	URL a imagen de la receta.
lang	Idioma de la receta

Ilustración 63. Entidad Recipes
(Elaboración propia)

- Entidad “Ingredients”: Entidad que se encargará de almacenar los ingredientes de una determinada receta:

Ingredients	
Atributo	**Descripción**
name	Nombre del ingrediente.
quantity	Cantidad del ingrediente.
recipe_recipe_id	Identificador de receta a la que pertenece.

Ilustración 64. Entidad Ingredients
(Elaboración propia)

- Entidad "Steps": Entidad encargada de almacenar la información de un determinado paso de una determinada receta:

Steps	
Atributo	**Descripción**
step_description	Descripción con el paso a realizar.
step_graphic_resource	URL de la imagen a mostrar.
step_order	Entero que expresa el orden del paso dentro de la receta.
recipe_recipe_id	Identificador de la receta.
step_type	Campo que indicará de qué tipo de paso se trata (Tarea en segundo plano o en primer plano).

Ilustración 65. Entidad Steps
(Elaboración propia)

- Entidad "Nutritional_values": Entidad encargada de almacenar los valores nutricionales de una determinada receta:

Nutritional_values	
Atributo	**Descripción**
name	Nombre del valor nutricional.
value	Cantidad aportada del valor nutricional.
Recipe_recipe_id	Identificador de receta.

Ilustración 66. Entidad Nutritional_values
(Elaboración propia)

- Entidad "recipe_charac": Entidad encargada de almacenar la relación entre receta y características de la misma:

Recipe_charac	
Atributo	**Descripción**
Recipes_recipe_id	Identificador único de receta a la que pertenece.
Charac_charac_id	ID de la característica.
Charact_charac_cat_category_id	ID de la categoría de la característica.

Ilustración 67. Entidad Recipe_charac
(Elaboración propia)

- Entidad "Characteristics": Entidad encargada de almacenar las características disponibles en el sistema a la hora de catalogar una receta:

Characteristics	
Atributo	**Descripción**
charac_id	ID de la característica (MD5 del nombre y el ID de categoría)
name	Nombre de la característica.
Charac_Cat_category_id	ID de la categoría a la que pertenece.

Ilustración 68. Entidad Characteristics
(Elaboración propia)

- Entidad "Characteristics_Categories": Entidad encargada de almacenar las categorías permitidas de características:

Characteristics_Categories	
Atributo	**Descripción**
category_id	ID de la categoría (MD5 del nombre y el idioma)
name	Nombre de la categoría.
lang	Idioma de la categoría

Ilustración 69. Characteristics_Categories
(Elaboración propia)

3. Implementación

Una vez definidos los requisitos y detalles del proyecto en el primer capítulo de esta memoria, y diseñada la solución tecnológica acorde a los mismos en el segundo, se comenzó con la implementación de la misma. Como líneas maestras de esta fase se tomaron las siguientes:

- Los componentes deberán ser lo más desacoplados posibles para favorecer la reutilización.
- Todas aquellas acciones y/o procesos que no sean necesarios para el flujo principal, pasarán a ejecutarse en el segundo plano para agilizar el uso de la aplicación.
- Se favorecerá el uso de librerías externas fiables en detrimento del desarrollo propio en los casos que sea posible.

Una vez clarificados estos patrones se comenzó con esta fase. A continuación, en este capítulo se comenzará con las primeras etapas, donde se establecerán las tecnologías a usar para posteriormente, y tras ejecutar el proceso de codificación, destacar aquellos elementos con especial relevancia del mismo. Tras esto, se proporcionará una actualización temporal de la planificación así como, una muestra de la funcionalidad final de la aplicación. Finalmente, y para terminar este apartado, se definirá y establecerá un plan de pruebas adecuado que permita asegurar la calidad del producto desarrollado.

3.1 Justificación tecnológica

Como se mencionaba anteriormente, en este primer subapartado se presentan y justifican las tecnologías usadas en la implementación de la solución propuesta en esta memoria. Para una mayor comprensión de lo aquí detallado, se han subdividido las tecnologías en dos categorías, principales y auxiliares, y éstas a su vez en aquellas relativas a la aplicación móvil (*front-end*) y al servidor (*back-end*).

3.1.1 Tecnologías principales

En esta primera categoría se presentan aquellas tecnologías más relevantes y que por su uso, resultan necesarias para la implementación de la solución y, por tanto, de la funcionalidad final de la aplicación.

3.1.1.1 Front-end

En el caso del *front-end* (aplicación Android), su funcionamiento se basa principalmente en el uso de las tecnologías a continuación presentadas.

3.1.1.1.1 Android SDK

Kit de desarrollo software oficial de Android. Android es el sistema operativo móvil más usado en la actualidad, con una cuota de mercado del 90% en España (Xataka, 2019), lo que lo hace foco ideal para el desarrollo de aplicaciones (al disponer un número de clientes potenciales muy amplio). Desarrollado por Android Inc., está basado en kernel Linux y fue presentado en 2007.

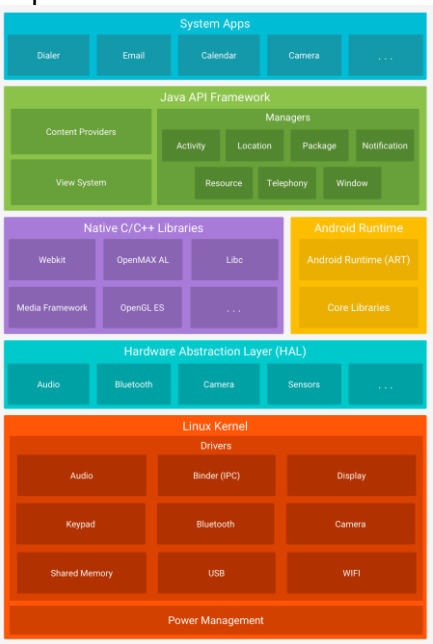


Ilustración 70. Arquitectura de Android
(Fuente: Android Developers)

En cuanto a su arquitectura (como puede observarse en la imagen) se basa en lo siguiente:

- Aplicaciones de sistema: Aplicaciones por defecto incluidas en el sistema, como navegador, app de sms, de llamadas, etc.
- Java API Framework: Capa de abstracción construida en lenguaje Java a través de la cual de forma sencilla se puede acceder a las funcionalidades del sistema operativo, permitiendo así, gestionar los ciclos de vida de las vistas, la administración de recursos, etc.
- Librerías nativas C/C++ y Android Runtime: En el caso del primero, se trata de un conjunto de bibliotecas nativas a través del Android NDK, cuyo desarrollo hace de base para la capa HAL. Finalmente, Android Runtime es el entorno de ejecución, donde cada aplicación ejecuta sus procesos de forma aislada.
- Hardware Abstraction Layer (HAL): Conjunto de APIs expuestas para el acceso a un alto nivel de abstracción a las capacidades hardware del dispositivo.
- Linux Kernel: Base de la plataforma Android. Se permite su personalización para adaptarlo a distinto hardware base.

Respecto a la versión del SDK, **se eligió como versión mínima la 23, siendo la máxima la versión 28** debido a que, según se estableció en el **requisito no funcional RNF002, se quería abarcar más del 50% de la cuota de dispositivos Android** (con esta elección se cubre el 74.8%, como puede verse en la imagen).

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	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Ilustración 71. Distribución de cuota de mercado de APIs Android (Fuente: Android Developers)

3.1.1.1.2 Android Studio

Entorno de desarrollo integrado (IDE) oficial para Android. Está basado en IntelliJ IDEA y sustituyó a Eclipse en 2013 como entorno oficial para el desarrollo de aplicaciones Android. Este IDE provee de una gran integración con el SDK de Android, incluyendo Android Virtual Devices (AVD), y con herramientas externas, como Git.

La elección de este IDE respecto a otros, como por ejemplo, Eclipse reside en su fácil instalación, su gran integración con herramientas y, principalmente, por la **oficialidad de éste**, lo que hace que disponga de una gran base de documentación.

3.1.1.1.3 Java

Lenguaje de programación multiplataforma y orientado a objetos creado por Sun Microsystems. Como se observa en la imagen mostrada a continuación, este lenguaje pertenece al Top 3 en uso actual, lo que provoca que exista una gran cantidad de documentación disponible.

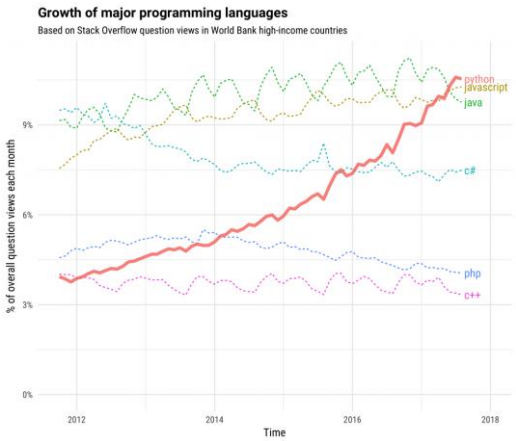


Ilustración 72. Crecimiento de uso de los lenguajes de programación (Fuente: GrapeCity, 2018)

En cuanto a su uso en este proyecto, ha de tenerse en cuenta que a día de hoy, existen dos alternativas oficiales para la implementación de aplicaciones nativas Android, Kotlin, un lenguaje de tipado estático, y Java. Debido al uso de Java durante el programa formativo y a su conocimiento anterior, se decidió usar esta opción pues, usar la contraria podría haber provocado la generación de una deuda técnica demasiado amplia, díficil de resolver en los plazos de este proyecto.

Finalmente, a día de hoy, el uso de Java en Android está mucho más extendido, por lo que la documentación existente es mayor, facilitando la resolución de inconvenientes o problemas que pudieran surgir durante la fase de implementación.

3.1.1.2 Back-end

De forma análoga, el servidor back-end de la aplicación, hace uso de las tecnologías mostradas a continuación

3.1.1.2.1 Python (Versión 3)

Lenguaje de programación interpretado y multiparadigma administrado por la *Python Software Foundation*. De forma resumida, su filosofía tiene por finalidad la ágil lectura del código, la simpleza y la compatibilidad inter-plataformas. Como se puede observar en la Ilustración 76, su crecimiento en los últimos años, ha seguido una línea ascendente con una pendiente elevada, motivado en parte, por su aplicación a los servicios web, el análisis de datos o la inteligencia artificial, campos en crecimiento.

Respecto a su elección, se motiva en los siguientes aspectos:

- Facilidad de aprendizaje: Como se mencionaba anteriormente, uno de sus pilares es la claridad, lo que provoca que su curva de aprendizaje sea bastante amena.
- Uso previo: Como se indicaba en el apartado correspondiente, uno de los requisitos no funcionales establecidos fue el uso de tecnologías ya trabajadas previamente en el máster.
- Amplia documentación: Dada la expansión sufrida por Python, dispone de un amplío número de recursos bibliográficos disponibles para su consulta de forma gratuita.
- Multiplataforma: Disponible en los principales sistemas operativos (Windows, OS y Linux)
- Gran abanico de librerías: Dispone de una gran cantidad de librerías de diversos propósitos, desde creación de servidores web a analítica de datos, facilitando la reutilización de código (establecido como requisito no funcional del presente proyecto).
- Bajo uso de recursos: Al ser un lenguaje interpretado, su uso de recursos CPU, RAM o disco es bastante bajo, facilitando su traslado a plataformas con recursos limitados. Este aspecto fue determinante a la hora de descartar Java, otro de los lenguajes ampliamente extendidos.

Finalmente, en cuanto al uso de la versión 3.x.x (3.6.8 en este proyecto), fue elegida debido a que la versión 2.x.x es una versión en vías de obsolescencia (fin de vida, *EOL* por sus siglas en inglés, programado para enero de 2020. Fuente: Python).

3.1.1.2.2 Librería Flask

Flask es un framework ligero destinado a la creación de *Web Server Gateway Interfaces (WSGI)* disponible para Python. Su alto nivel de abstracción, facilita la creación de APIs simplemente con la adición de anotaciones, permitiendo la construcción tanto de aplicaciones sencillas como algo más complejas, incluyendo autenticación y traceo de la petición.

Respecto a su elección, radica principalmente en su uso anterior en otros proyectos personales y en la ligereza en términos de uso de recursos respecto a otra de las opciones, Django, pues éste, aunque provee mayores capacidades (adaptación a MVC, mapeo de objetos en base de datos, autenticación por defecto, etc.) está recomendado para aplicaciones web de mayor envergadura donde se provea de un portal web interno.

3.1.1.2.3 MariaDB

MariaDB es un sistema de gestión de base de datos SQL desarrollado bajo las mismas directrices que se usaron en el desarrollo de MySQL, tras la compra de éste por parte de Oracle.

La elección del uso de un motor SQL tradicional respecto a otras opciones más novedosas como NoSQL o modelos híbridos, reside en que se explota el uso del modelo relacional, por ejemplo, en las recetas con sus ingredientes, además de la amplia estandarización y documentación disponible en la web.

Así, la elección de MariaDB como base de datos de la aplicación, entre otras opciones SQL como MySQL o Microsoft SQL Server, radica en su licencia GPL, la cual permite su uso libre, cumpliendo así con uno de los requisitos funcionales establecidos para el presente proyecto, y en el alto rendimiento que puede entregar.

3.1.1.2.4 Ubuntu Server

Sistema operativo basado en Debian con licencia de código abierto. Su uso en el proyecto fue la de ejecutar y alojar el código Python desarrollado junto a la base de datos MariaDB. Los motivos que llevaron a su elección fueron:

- Uso amplio en servidores web (se cree que posee una cuota superior al 50% en el mercado Linux).
- Licencia libre, lo que apoya uno de los requisitos funcionales establecidos.
- Gran cantidad de documentación disponible, gracias en parte, a su expansión.
- Uso de recursos relativamente bajos.

3.1.2 Tecnologías auxiliares

Una vez presentadas las tecnologías principales de este proyecto, en esta segunda categoría se presentan aquellas tecnologías que por su uso, aunque diferenciador, simplemente han actuado como catalizadores y facilitadores del

desarrollo y, por tanto, su omisión únicamente habría afectado a la facilidad y/o eficiencia de ejecución de esta fase.

3.1.2.1 Front-end

De nuevo, en el caso del front-end (aplicación Android), se hizo apoyo de las siguientes tecnologías para complementar y/o facilitar la implementación del mismo:

- **Retrofit**: Librería para la creación rápida y simple de un cliente HTTP para Android Java. Con esta librería se consiguió disminuir considerablemente los tiempos de desarrollo en lo que a la integración entre back-end y front-end se refiere, pues su alto nivel de abstracción provocó que la inclusión de nuevas llamadas a la API o el procesamiento de resultados de las mismas, quedara considerablemente simplificado.
  En cuanto a su elección, se basó en cumplir uno de los requisitos funcionales establecidos en el proyecto además de tratar de simplificar la implementación de la integración y así poder poner el foco en la usabilidad de la misma, aspecto especialmente relevante de este trabajo.

- **Auth0**: Librería usada para la verificación y validación del origen del token JWT usado en el proceso de autenticación, de forma que se valide el emisor de dicho token así como, su integridad para en caso de detectar alguna irregularidad (servidor origen falso, token manipulado mediante man-in-the-middle, etc.), cancelar el login del usuario.

- **Git:** sofware de versionado diseñado por Linus Torvalds. Su objetivo es simplificar el desarrollo de implementaciones con una gran cantidad de archivos fuente mediante el registro de cambios y la coordinación del trabajo de varias personas sobre ficheros compartidos. Su uso en este proyecto fue el de ir implementando mejoras a través de la rama "*dev*" e ir consolidándolas en la rama "*master"*, a la vez que, tras cada entrega, se iban generando nuevas ramas indicando la *release* de la que se trataba, de forma que, ante un bug, se pudiera volver a este punto y solventarlo de forma previa a la entrega.

- **Glide:** Librería para la gestión y carga de imágenes desde Internet en Android. Aunque en un principio se había establecido el uso de Picasso, otra librería similar a ésta en funcionalidad, tras un proceso de investigación, se observaron una serie de mejoras clave en varios aspectos de Glide respecto a Picasso, provocando su elección: Uso de cacheado de imágenes, consumo de memoria del 50% respecto a Picasso, tiempos de carga sin caché parecidos y método de uso parecido.

- **MaterialIO**: Librería para el uso de componentes de *Material Design* en Android. *Material Design* es una normativa establecida por Google a partir del API 23 (*Lollipop*) de Android, para la visualización de su sistema operativo además de otras plataformas. Este concepto simplifica la creación de interfaces gráficas atractivas y usables para los usuarios, principal fin de este proyecto, por lo que su uso fue de gran utilidad a la hora de proveer a la aplicación de un buen interfaz.

3.1.2.2 Back-end

De forma análoga, el servidor *back-end* de la aplicación, hace uso de las siguientes tecnologías auxiliares:

- **Editor Atom.io**: Editor de texto desarrollado por Github con un gran abanico de extensiones disponible así como, integración con versionado por Git. En este proyecto fue usado para implementar el código del servidor principal así como, anotar ciertos aspectos relevantes mostrados en los siguientes apartados de esta documentación.

- **Lenguaje Bash**: Lenguaje de consola e intérprete de comandos ampliamente usado en las distribuciones Linux. Su uso en el proyecto fue el de manejar el sistema operativo del servidor, acceder mediante el cliente correspondiente a la base de datos, tareas de configuración y finalmente, desarrollo de scripts de pruebas sobre el servidor.

- **Paquete Unix cURL**: Proyecto software para la ejecución de peticiones bajo distintos protocolos, entre ellos, HTTP y HTTPS, disponible para Linux, MAC OS y Windows. En este proyecto se usó para el desarrollo de pruebas unitarias sobre los distintos endpoints del servidor.

- **Paquete Unix jq**: Lenguaje de programación de alto nivel para la gestión de flujos con objetos JSON. En el presente proyecto, fue usado para el tratamiento de los resultados devueltos por el servidor y su evaluación posterior.

- **Paquete Unix libmysqlclient-dev**: Cliente de Linux para el acceso a la base de datos MariaDB. Fue usado para la configuración inicial de la base de datos.

- **Git**: Detallado anteriormente en el apartado 3.1.2.1Front-end.

- **Librería Base64**: Librería de Python dedicada a la codificación y descodificación de cadenas en base64. Su uso principal, fue la descodificación de las cabeceras de autorización a la hora de realizar el inicio de sesión por parte del usuario.

- **Librería Datetime**: Librería de Python para la gestión de fechas y timestamps. Su uso en este proyecto fue la de formatear y obtener las fechas de expiración de token o, el formateo de la fecha de cumpleaños del usuario.

- **Librería Gevent**: Librería de Python destinada al manejo de redes y los métodos relacionados con éstas. En el proyecto fue usada para la creación de un Web Server Gateway Interface (WGSI) que enmascarara la API desarrollada, añadiéndole la capa de seguridad con tráfico cifrado (SSL/TLS) y acercando el proyecto a lo que posteriormente podría ser un servidor en producción.

- **Librería Hashlib**: Librería de Python para la aplicación de funciones hash sobre cadenas y ficheros. En este proyecto, fue usada para la codificación a MD5 de forma que se pudieran comprobar por ejemplo, si la contraseña introducida por el usuario es correcta, pues el almacenado en base de datos, se producía cifrado para evitar que una fuga de información pudiera comprometer las cuentas de usuario.

- **Librería Json**: Librería de Python para la gestión integral de objetos JSON. En el servidor, fue usada para generar o analizar todos los objetos en esta codificación, que emitía o recibía éste.

- **Librería Os**: Librería de Python para el uso de funciones dependientes del sistema operativo de forma totalmente portable. Su uso en el servidor fue el de obtener las variables de entorno para la configuración del acceso a base de datos por parte de las APIs.

- **Librería Mysql-connector**: Librería de Python para la gestión de un cliente MariaDB/MySQL. Su uso facilitó el acceso y la gestión de los datos contenidos en base de datos, pudiendo así, obtener o persistir la información necesaria reclamada o enviada por el cliente.

- **Librería Time**: Librería de Python destinada al manejo y gestión de fechas. En el proyecto fue usada para el formateo de ciertas fechas así como, cálculos que fuera necesario realizar sobre éstas (por ejemplo, en la fecha de expiración de los tokens).

- **Paquete Unix OpenSSL**: Librería Linux ampliamente propagada para su uso en criptografía y especialmente, en la gestión de certificados TLS/SSL. Su uso en este proyecto fue la de generar el certificado del servidor y las claves pública y privada para la codificación de los tokens JWT.

- **Paquete Unix Pip (Versión 3)**: Sistema de gestión de paquetes para Python. En el proyecto se usó para la descarga e instalación de las distintas librerías expuestas en este apartado.

- **Librería Pyjwt**: Librería de Python para el manejo y gestión de Java Web Tokens (JWT). JWT es un estándar abierto basado en JSON para la gestión de identidades, permitiendo incluir distintos niveles de acceso o información disponible. El motivo por el que se tomó esta opción para el proceso de autenticación de usuario reside en la posibilidad de incluir a futuro, información adicional que permita la diferenciación de usuarios, añadiendo por ejemplo, paquetes disponibles para el mismo (subscripción).

- **Oracle SQL Data Modeler**: Herramienta gráfica gratuita destinada a simplificar y mejorar la productividad en las tareas de modelado de datos. Su uso se limitó a la creación de la estructura relacional de la base de datos y sin duda, su opción de generación automática de código SQL para esta tarea, ayudó a reducir los tiempos de desarrollo (aunque, al ser una herramienta de Oracle, no disponía de la opción de generación de código para MariaDB, teniendo que realizar algunas modificaciones posteriores sobre ciertos tipos de columnas).

- **VMWare Workstation**: Hipervisor disponible para sistemas operativos Windows y Linux. Su uso fue clave a la hora del desarrollo del back-end, pues permitió el aislamiento y prueba de creación del mismo, de forma que se pudiera certificar tanto la independencia de éste, como su posterior exportación a otros entornos.

3.2 Aspectos destacados del proceso de implementación

Una vez especificadas las tecnologías usadas en la implementación del servidor y la aplicación, conviene detallar algunos puntos destacados que caracterizaron el desarrollo de la solución. En este apartado se ponen en relevancia dichos puntos separando de nuevo el servidor *back-end* de la aplicación *front-end*.

3.2.1 Front-end

En este apartado se especifican aquellos detalles destacados en lo que a implementación de la aplicación se refiere.

3.2.1.1 Estructura del proyecto

En este caso la estructuración del proyecto ha seguido el patrón genérico ofrecido por Android Studio, donde se dispone de los siguientes directorios:

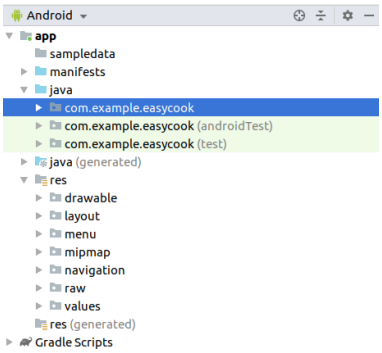


Ilustración 73. Estructura proyecto Android
(Elaboración pròpia)

Estos directorios fueron usados para los siguientes objetivos:

- Sampledata: Este directorio es usado para la inclusión de datos de ejemplo a usar en la aplicación. No usado en el proyecto.
- Manifests: Directorio dedicado a contener los manifiestos de la aplicación. En este proyecto solo se incluyó el fichero *AndroidManifest.xml* donde se incluyen detalles de la aplicación además de las actividades que contiene, los permisos que necesita o el lanzador de la misma.
- Java: Directorio destinado a contener los ficheros fuente de la aplicación. Este directorio a su vez, contiene los siguientes directorios:
    - com.example.easycook: Contiene los ficheros fuente de la aplicación. Aquellos relativos a la lógica de la aplicación (controlador) y al almacenamiento de objetos (modelo).
    - com.example.easycook (androidTest): Directorio dedicado a contener test unitarios que hagan uso de la instrumentación propia de Android. Usado para contener los test unitarios elaborados para la aplicación.
    - com.example.easycook (Test): Directorio dedicado a contener test unitarios que no hagan uso de la instrumentación propia de Android. No usado en este proyecto.

- Java (generated): Directorio donde se almacenan los ficheros fuente de ciertos elementos generados por herramientas externas. En este proyecto, se almacenaron aquí elementos de configuración como el versionado además, del modulo Glide customizado detallado más adelante.
- Res: Directorio donde se almacenan los elementos relativos la interfaz de usuario. En este proyecto, este directorio contendrá todos los layouts de la aplicación, los iconos básicos de la misma, las definiciones de menús y navegaciones, los valores de colores, strings y estilos y el sonido que se reproducirá tras la finalización de un temporaizador.
- Gradle Scripts: Directorio dedicado a contener ficheros de configuración de la herramienta de compilación de Android, Gradle. En este proyecto, fue usado para indicar las importaciones necesarias de librerías externas, indicar la versión mínima de Android necesaria y la nomenclatura de versionado.

Finalmente, con el fin de concretar aquellos puntos personalizados en la estructuración del proyecto, se elaboró la siguiente jerarquía en la organización de archivos fuente:

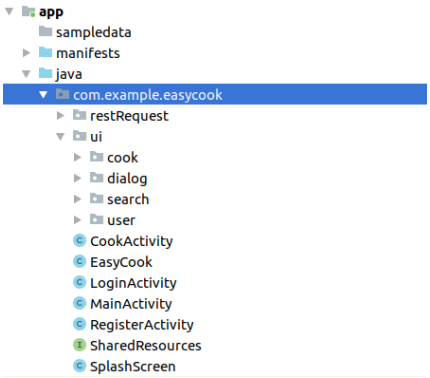


Ilustración 74. Estructuración interna del código (Elaboración pròpia)

- RestRequest: En este directorio se incluyeron todas las clases necesarias para el uso de la librería Retrofit y Glide es decir, aquellos elementos obligatorios para el correcto funcionamiento de la conexión con el servidor back-end. En este directorio, además de configuración concreta de Glide y Retrofit que se detallará más adelante, se encuentran aquellas clases de modelo, necesarias para la "objetización" de los elementos JSON enviados por el servidor.
- Ui: Directorio dedicado a contener los fragmentos usados en la aplicación, es decir, los elementos más cercanos al cliente. A su vez, se encuentran los siguientes subdirectorios:
  - Cook: Contiene los ficheros fuente de aquellos fragmentos relacionados con el proceso de seguimiento de la receta.
  - Dialog: Contiene los ficheros fuente de los diálogos (pop-ups) mostrados en la interfaz.
  - Search: Contiene tanto el fichero fuente del fragmento de búsqueda como los elementos Recycler y ViewHolder necesarios para el correcto mostrado de la información.
  - User: Dedicado a contener el fragmento de información de usuario.
- Activities, SharedResources y Easycook: En la raíz del directorio se encuentran las 5 actividades que componen esta aplicación así como, la interfaz *SharedResources* con elementos de configuración global (como la IP de back-end) y la clase *Easycook* para el uso del contexto de aplicación en el servicio de consulta a la API.

3.2.1.2 Versionado

Como se comentó en el apartado 3.1 Justificación tecnológica, durante el proceso de desarrollo se usó Git como herramienta de versionado. Así, la nomenclatura usada, corresponde con un identificador de versión con tres apartados (X.Y.Z donde X, Y y Z son números) y un nombre coloquial de versión. Éstos se definen de acuerdo a las siguientes normas:

- Primer número (X): Comenzará en 0 y tendrá tendencia ascendente cuando el cambio introducido tenga una magnitud considerable, entendiéndose por ejemplo, un cambio de arquitectura.
- Segundo número (Y): Comenzará en 0 y tendrá tendencia ascendente cuando el cambio introducido sea una nueva funcionalidad, modificación de una introducida o extensión de la misma.
- Tercer número (Z): Comenzará en 0 y tendrá tendencia ascente. Variará ante resolución de bugs de desarrollo o pequeños cambios.
- Nombre coloquial: En este caso se ha decidido usar nombres de faraones egipcios, comenzando por la A y siendo la primera versión *Akenaton*. Cambiará y pasará a ser otro faraón con la siguiente letra en orden alfabético a medida que el primer o segundo número aumente.

3.2.1.3 Precarga de información

Aunque no se ha masificado su uso debido a las restricciones temporales del proyecto, sin embargo se ha querido incluir una primera aproximación a la precarga de elementos que agilicen el uso de la aplicación por parte del cliente. Así, el diálogo informativo mostrado tras el *splash screen*, se carga mientras dura éste, de forma que su mostrado se realice sólo si éste ha podido ser cargado.

Esta estrategia podría expandirse en el futuro a otros apartados como las recetas mostradas al inicio, los ingredientes tras seleccionar una receta o algunos pasos anteriores y posteriores al actual durante el desarrollo de la receta.

3.2.1.4 Comprobación de origen y no alteración de JWT

De nuevo, como se ha mencionado anteriormente, aunque la seguridad no constituía un punto fundamental en este proyecto, se ha querido añadir un chequeo adicional que evite ataques como *man-in-the-middle* o el uso de servidores no originales en la aplicación mediante la comprobación de integridad y origen del token JWT, gracias al uso de la librería Auth0.

3.2.1.5 Uso de HTTPS con certificado autofirmado

Como se añadirá en el apartado homólogo a éste en el apartado dedicado al *back-end*, se ha optado por el uso de tráfico seguro mediante el uso de certificados autofirmados. Este hecho supuso en la aplicación un verdadero reto al obligar a generar un cliente modificado de tipo *OkHTTPClient* que aceptara este tipo de

certificados como seguros, que posteriormente fuera usado por las librerías Retrofit y Glide. Respecto a esta última, supuso la generación de la elaboración de un módulo *ad hoc* para el proyecto que forzara al uso de este cliente modificado.

3.2.1.6 Fragments autocontenidos

Durante la fase de diseño se decidió usar un modelo autocontenido en los fragmentos, es decir, cada fragmento contendrá su propia lógica de aplicación. Esta decisión fue tomada con la idea de facilitar la posible adaptación futura de la aplicación al ecosistema *tablet*, de forma, que al disponer de un tamaño de pantalla mayor, pudieran usarse distintos fragmentos en la misma interfaz con cambios de código relativamente pequeños.

3.2.1.7 Adaptación al uso de distintos idiomas

Como se mencionó en el apartado de requisitos no funcionales al inicio de este documento, la aplicación únicamente estaría disponible en español aunque se facilitaría la posterior inclusión de nuevos idiomas. Es por ello, por lo que todo texto usado en la aplicación está incluido en el fichero "*strings.xml*" del proyecto (siguiendo las buenas prácticas establecidas por Android), de forma que la adaptación a otro idioma únicamente conlleve la inclusión de un fichero similar bajo este nuevo idioma.

3.2.1.8 Interfaz *responsive*

Aunque en el apartado de requisitos antes mencionado, se comentaba que el dispositivo final de la aplicación sería el smartphone Samsung Galaxy J6 2018, se ha establecido el uso de variables y configuraciones adaptativas a los distintos tamaños de pantalla, estableciendo las alturas y anchuras dependientes del contenido, del layout padre o de medidas en base *dp* (píxeles de densidad independiente) y los textos en medidas en unidades *sp* (Píxeles de escala independiente), facilitando así su adaptación a dispositivos con pantallas de distintos tamaños y evitando la realización de una re-ingeniería para este fin.

3.2.1.9 Control por voz

El uso de control por voz ha sido realizado mediante el uso de la clase *SpeechRecognizer*. Esta clase se encarga de habilitar y deshabilitar el micrófono, transformando el audio a texto. Además en orden de lo especificado en el punto 3.2.1.7 Adaptación al uso de distintos idiomas se ha optado por indicar los comandos de voz en el fichero "*strings.xml*", de forma que no se requiera de grandes cambios arquitectónicos para la inclusión de nuevos idiomas.

3.2.1.10 Paginación de resultados

Aunque inicialmente no fue contemplado, tras un profundo análisis del mercado de dispositivos móviles, se quiso "democratizar" el uso de la aplicación aquí desarrollada, y en este aspecto, el consumo de recursos fue clave. Por este motivo, finalmente se incluyó la paginación de resultados, es decir, cuando se realiza una búsqueda de recetas, en lugar de devolver el conjunto completo de las mismas (que consumiría grandes cantidades de memoria y ralentizaría la carga), se devuelve un subconjunto de éstas. A nivel de aplicación, esto requirió de incluir un *ScrollListener* que permitiera la carga automática e incremental del conjunto de recetas además, de disponer de dicha funcionalidad en el back-end.

3.2.1.11 Actualización automática de resultados en búsqueda

Con el fin de enriquecer la experiencia de usuario, se optó por automatizar la búsqueda de recetas, de forma que, conforme el usuario fuera escribiendo en el campo de búsqueda, el sistema fuera actualizando el conjunto de resultados. Para ello, fue necesario implementar la clase correspondiente *OnQueryTextListener* además de incluir variables auxiliares que evitaran la superposición de cargas de información. De igual forma, se implementó una variable que estableciera el mínimo de caracteres necesarios para poder comenzar a mostrar resultados parciales.

3.2.2 Back-end

De igual forma, en este apartado se muestran aquellos aspectos a tener en cuenta respecto a la implementación del servidor.

3.2.2.1 Estructura del proyecto

En cuanto a la estructuración de los ficheros de proyecto, se optó por establecer la siguiente estructura:

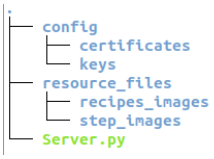


Ilustración 75. Estructura de ficheros del servidor (Elaboración propia)

Como se observa, se establecieron dos directorios junto al ejecutable Python, uno dedicado a contener parámetros de configuración necesarios para la ejecución del servidor, incluyendo en él las claves pública y privadas de firma para los token JWT y los certificados para el uso de HTTPS, y otro para contener los recursos gráficos de la aplicación, separados en imágenes de recetas y acciones de paso a paso.

Respecto al código fuente, su estructuración consta en los siguientes apartados:

- Variables globales: En este apartado se indican las variables de configuración global del servidor tales como el tiempo de validez del token, el tamaño de página devuelto o la localización de ciertos ficheros clave.
- Paths de acceso a APIs: Apartado dedicado al establecimiento de las rutas permitidas del servidor así como sus métodos.
- Métodos auxiliares: Funciones complementarias a las principales de las APIs. Esta separación ha sido hecha para la posible reutilización de las mismas por varios métodos diferentes.
- Función principal: Main de la aplicación donde se da la opción de configurar el servidor con o sin HTTPS y para entornos de desarrollo o producción.

3.2.2.2 Versionado

Se siguió la misma estrategia de versionado comentado en el apartado 3.2.1.2 sólo que en lugar de nombres de especies de roedores para el nombre coloquial de versión, se optó por nombres de especies de roedores comenzando por la A, siendo nombrada esta versión como *Arvicolina*.

3.2.2.3 Requisitos detectados

Durante la fase de integración se detectó un requisito en cuanto a la versión de la base de datos MariaDB, la cual deberá ser versión 10.2 o superior para poder usar algunas de las funcionalidades implementadas en el servidor.

3.2.2.4 Inclusión de cabeceras de información

Todas las APIs incluyen la opcionalidad de incluir y capturar dos cabeceras con información relevante de cara a su posterior análisis. Las cabeceras incluidas fueron:

- "x-version-app": Incluirá el nombre extendido de versión de la aplicación, lo que permitirá almacenar eventos con éste, para a posteriori, conocer de forma exacta el porcentaje de uso de cada una de las versiones de la aplicación.
- "Authorization": Aunque como se mencionó en el apartado correspondiente de esta memoria, el inicio de sesión es opcional e innecesario para el uso principal de la aplicación, se ha implementado de cara a posteriores enriquecimientos de la experiencia de usuario, mediante le uso de técnicas de Business Intelligence, pues de esta forma, toda acción realizadaqueda almacenada junto al identificador de usuario (si estuviera logado).

3.2.2.5 Almacenado de eventos

De la interacción con cada una de las APIs disponibles, se producen eventos indicando la acción a la que se refiere el evento, el recurso sobre el que se

interacciona, la hora exacta y el usuario (si estuviera logueado). Esto conforma el paquete de extracción de métricas básico para la realización de analíticas de negocio posteriores, además de una buena aproximación para la instalación posterior de elementos de monitorización que permitan detectar comportamientos anómalos tales como una tasa de error elevada.

3.2.2.6 Uso de HTTPS con certificados autofirmados

Como se mencionaba en el apartado de objetivos de este proyecto previamente en esta memoria, aunque la seguridad solo será tenida en cuenta en forma de conceptos básicos, se ha decidido introducir un tráfico cifrado mediante HTTPS entre los clientes y el servidor, de forma que se cumplan los requisitos básicos mencionados.

Este cifrado se realiza mediante claves extraídas de un certificado autofirmado generado *ad hoc* al proyecto, lo que requirió de configuración adicional en la aplicación como se comentó en el apartado correspondiente, pues, se considera un certificado no seguro.

3.2.2.7 Simplificación de arquitectura

Al igual que en el apartado anterior se tuvieron únicamente en cuenta las nociones básicas de seguridad, se ha realizado una simplificación de la arquitectura, optando por un modelo autocontenido, evitando el uso de otras piezas imprescindibles en un sistema productivo como son la redundancia de componentes, un API-gateway, desacoplado de eventos del flujo principal, servidor de recursos, proxy inverso o sistemas de monitorización proactiva y reactiva. Esta simplificación, ha sido motivada por el ajuste a la temporalidad del proyecto, así como la intención de poner en valor el producto en sí, sin aderezarlo con componentes externos que pudieran desvirtuar el valor real del mismo.

3.2.2.8 Adaptación a distintos idiomas

De igual forma, como se definió en el apartado homólog del *front-end*, en esta primera versión, el idioma elegido será el español, no obstante, para facilitar la futura ampliación del catálogo de idiomas disponibles, se ha optado por crear un esquema de base de datos que soporte esta acción, limitándola únicamente a la adición de filas extra para cada idioma.

3.2.2.9 Eficiencia en base de datos

Dado que muchas de las tablas de base de datos constaban en su primera versión de claves primarias compuestas de varios campos, se optó en su lugar por el uso de identificadores de 32 caracteres, fruto de la función hash MD5 sobre estos campos. De esta forma, aunque el espacio ocupado por una fila aumenta, en futuras

relaciones con otras tablas, este espacio puede reducirse, además de simplificarse esta relación gracias a disponerse de un único campo por el que efectuarla.

Esta decisión, también ayudó a la posterior integración con el código fuente, gracias a que las consultas SQL fueron más simples y rápidas que si por el contrario, se hubieran usado varias columnas.

Por último, de igual forma, un traspaso posterior a una base de datos columnar, se simplificaría gracias a esta arquitectura. Si por el contrario, se deseara optimizar el comportamiento de la actual, se podrían generar índices en campos muy concretos que mejorarían los tiempos de respuesta en la base de datos.

3.2.2.10 Variables de entorno

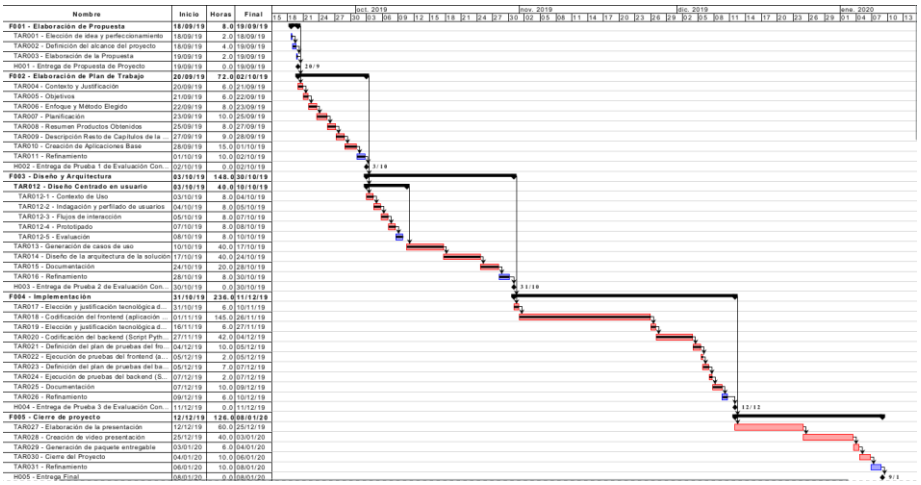
Para evitar la modificación del código fuente Python, se ha optado por el uso de variables de entorno a la hora de integrar y configurar el acceso a base de datos. Este hecho paralelamente facilitó la posterior "dockerización" del servidor, gracias a la posibilidad de fácilmente configurar su conexión con un elemento externo (contenedor) de MariaDB, simplemente alterando las variables globales del contenedor del servidor.

3.2.2.11 Soporte a paquetización de la oferta

Gracias al uso de JWT en la autenticación, se podría añadir de forma liviana el uso de paquetes preconfigurados bajo pago con acceso a ciertas recetas exclusivas, pues la modificación constaría de añadir un par de tablas más en base de datos, añadir la posibilidad de adquirir paquetes en la aplicación, chequear el acceso en el método de búsqueda y cargar los paquetes en el campo "*packg*" del token, de forma que la aplicación rápidamente pudiera saber qué paquetes tiene contratados el usuario.

3.3 Planificación actualizada

Una vez detallados los puntos clave de la fase de implementación, se realizó una actualización de la planificación del proyecto para así conocer de forma objetiva el cumplimiento de los tiempos previstos inicialmente. Así, en primer lugar, se actualizó el diagrama Gantt presentado en el apartado 1.4 Planificación del Trabajo:



Nombre	Inicio	Horas	Final
F001 - Elaboración de Propuesta	**18/09/19**	**8.0**	**19/09/19**
TAR001 - Elección de idea y perfeccionamiento	18/09/19	2.0	18/09/19
TAR002 - Definición del alcance del proyecto	18/09/19	4.0	18/09/19
TAR003 - Elaboración de la Propuesta	19/09/19	2.0	19/09/19
H001 - Entrega de Propuesta de Proyecto	19/09/19	0.0	19/09/19
F002 - Elaboración de Plan de Trabajo	**20/09/19**	**72.0**	**02/10/19**
TAR004 - Contexto y Justificación	20/09/19	6.0	21/09/19
TAR005 - Objetivos	21/09/19	6.0	22/09/19
TAR006 - Enfoque y Método Elegido	22/09/19	8.0	23/09/19
TAR007 - Planificación	23/09/19	10.0	25/09/19
TAR008 - Resumen Productos Obtenidos	25/09/19	8.0	27/09/19
TAR009 - Descripción Resto de Capítulos de la ...	27/09/19	9.0	28/09/19
TAR010 - Creación de Aplicaciones Base	28/09/19	15.0	01/10/19
TAR011 - Refinamiento	01/10/19	10.0	02/10/19
H002 - Entrega de Prueba 1 de Evaluación Con...	02/10/19	0.0	02/10/19
F003 - Diseño y Arquitectura	**03/10/19**	**148.0**	**30/10/19**
TAR012 - Diseño Centrado en usuario	**03/10/19**	**40.0**	**10/10/19**
TAR012-1 - Contexto de Uso	03/10/19	8.0	04/10/19
TAR012-2 - Indagación y perfilado de usuarios	04/10/19	8.0	05/10/19
TAR012-3 - Flujos de interacción	05/10/19	8.0	07/10/19
TAR012-4 - Prototipado	07/10/19	8.0	08/10/19
TAR012-5 - Evaluación	08/10/19	8.0	10/10/19
TAR013 - Generación de casos de uso	10/10/19	40.0	17/10/19
TAR014 - Diseño de la arquitectura de la solución	17/10/19	40.0	24/10/19
TAR015 - Documentación	24/10/19	20.0	28/10/19
TAR016 - Refinamiento	28/10/19	8.0	30/10/19
H003 - Entrega de Prueba 2 de Evaluación Con...	30/10/19	0.0	30/10/19
F004 - Implementación	**31/10/19**	**236.0**	**11/12/19**
TAR017 - Elección y justificación tecnológica d...	31/10/19	6.0	10/11/19
TAR018 - Codificación del frontend (aplicación ...	01/11/19	145.0	26/11/19
TAR019 - Elección y justificación tecnológica d...	16/11/19	6.0	27/11/19
TAR020 - Codificación del backend (Script Pyth...	27/11/19	42.0	04/12/19
TAR021 - Definición del plan de pruebas del fro...	04/12/19	10.0	05/12/19
TAR022 - Ejecución de pruebas del frontend (a...	05/12/19	2.0	05/12/19
TAR023 - Definición del plan de pruebas del ba...	05/12/19	7.0	07/12/19
TAR024 - Ejecución de pruebas del backend (S...	07/12/19	2.0	07/12/19
TAR025 - Documentación	07/12/19	10.0	09/12/19
TAR026 - Refinamiento	09/12/19	6.0	10/12/19
H004 - Entrega de Prueba 3 de Evaluación Con...	11/12/19	0.0	11/12/19
F005 - Cierre de proyecto	**12/12/19**	**126.0**	**08/01/20**
TAR027 - Elaboración de la presentación	12/12/19	60.0	25/12/19
TAR028 - Creación de video presentación	25/12/19	40.0	03/01/20
TAR029 - Generación de paquete entregable	03/01/20	6.0	04/01/20
TAR030 - Cierre del Proyecto	04/01/20	10.0	06/01/20
TAR031 - Refinamiento	06/01/20	10.0	08/01/20
H005 - Entrega Final	08/01/20	0.0	08/01/20

Ilustración 76. Planificación actualizada tras implementación (Elaboración propia)

Como se observa en la imagen, los plazos establecidos inicialmente se han cumplido, exceptuando los siguientes puntos que, aunque no han sido reflejados, merece la pena destacar para finalizar este apartado:

- Sobre-esfuerzo durante el prototipado: Durante la ejecución de la tarea TAR012-4, se optó por **tomar la decisión** tras consultar al tutor responsable, de realizar el **prototipado directamente en Android Studio**. Esta decisión, **aunque acertada a medio plazo, produjo una desviación importante en el esfuerzo necesario para completarla**, obligando a un sobre-esfuerzo de 20 horas respecto a las 8 planificadas para esta tarea, debido a la complejidad que entrañaba el uso de Android. Sin embargo, en el esfuerzo general del proyecto, redujo algo los tiempos de implementación lo que permitió añadir características no contempladas inicialmente como el uso de HTTPs.
- Retraso de la fase de evaluación DCU: Aunque durante esta fase DCU se elaboraron todos los documentos necesarios para la ejecución del test con usuarios, su **ejecución, tras consenso con el tutor responsable, se realizó durante la fase de implementación**, de forma que pudieran evaluarse todas las características de la aplicación pues, por ejemplo, el uso de comandos de voz, no estaban incluidos en el prototipo inicial. **Esta decisión, sin duda ha sido acertada** pues, por una parte ha permitido probar un prototipo totalmente funcional de la aplicación y por otra, **el riesgo de fallo grave de usabilidad quedaba reducido al diseñar la aplicación en base a los conocimientos extraídos durante la asignatura de usabilidad** del máster.

3.4 Funcionalidad de la solución

Tras justificar las tecnologías, implementar la solución tecnológica y analizar el estado actual de proyecto, en este apartado se muestran las funcionalidades de la solución resaltando aquellos puntos destacadas de las mismas.

3.4.1 Usuarios

En primer lugar, se muestran las funcionalidades más cercanas al usuario, las cuales se encargan de gestionar el ciclo de vida del mismo en la interacción con la aplicación.

3.4.1.1 Registro de usuario

Registro del usuario en la aplicación, proceso para el cual necesitará de introducir una serie de datos personales:

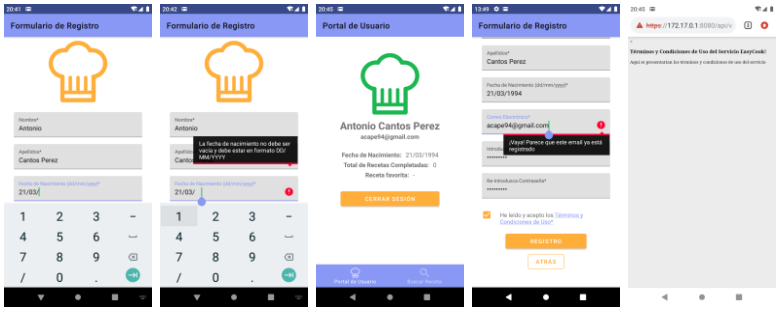


Ilustración 77. Funcionalidad Registro (Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Detección de errores proactiva en los distintos campos
- Formato automático del campo "fecha de nacimiento"
- Detección de errores reactiva en el campo "correo electrónico"
- Registro del usuario y auto-login de usuario

3.4.1.2 Login de usuario

Inicio de sesión del usuario en la aplicación mediante usuario y contraseña:

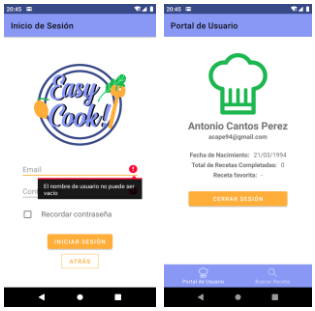


Ilustración 78. Funcionalidad Login (Ilustración propia)

En este caso, las funcionalidades aquí presentes son:

- Detección de errores reactiva
- Inicio de sesión de usuario

3.4.1.3 Consulta de información de usuario

Panel de información del usuario en el que se detallan algunos datos personales del usuario así como, una primera aproximación a datos surgidos de inteligencia de negocio, como son el número de recetas completadas o la receta favorita.

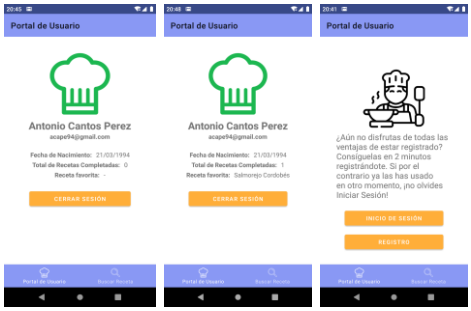


Ilustración 79. Funcionalidad de Información de usuario (Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Cierre de sesión
- Actualización automática por temporizador de los datos personales
- Caducidad de sesión

3.4.2 Recetas

Panel principal de la aplicación donde se muestra un listado de recetas y las distintas opciones de búsqueda.

3.4.2.1 Búsqueda por filtro de texto

Búsqueda en la que mediante la introducción de texto se obtiene un listado de recetas cuyo nombre coincide con la cadena introducida:

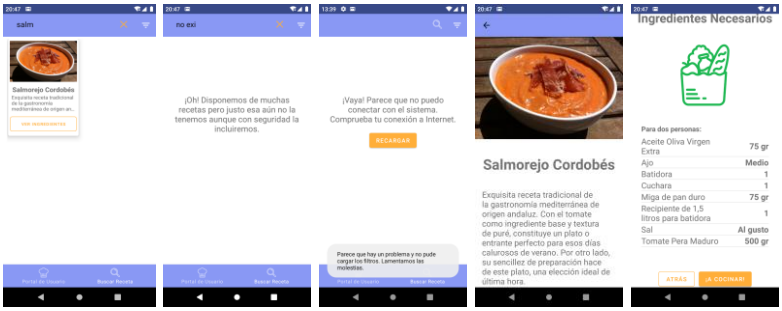


Ilustración 80. Funcionalidad filtrado de texto (Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Búsqueda parcial conforme se va escribiendo texto
- Error reactivo de falta de recetas
- Error reactivo de falta de filtros
- Error reactivo por fallo de conexión con opción a re-intento
- Carga automática e incremental
- Acceso a receta
- Acceso a ingredientes de receta

3.4.2.2 Búsqueda por filtros

Búsqueda en la que mediante la introducción de una serie de requisitos preestablecidos se obtiene un listado de recetas que los cumplen:

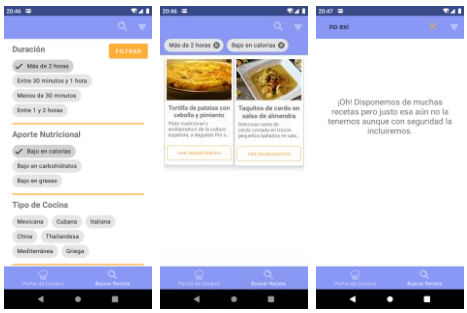


Ilustración 81. Funcionalidad de filtrado automático
(Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Búsqueda intuitiva por filtros de diversa índole
- Carga de filtros configurados en back-end
- Cancelación de filtros con adaptación de búsqueda automática
- Error reactivo de falta de recetas

3.4.2.4 Búsqueda combinada

Combinación de los casos 3.4.2.1 y 3.4.2.2.

Ilustración 82. Funcionalidad filtrado combinado
(Elaboración propia)

3.4.2.5 Resumen de Receta

Panel donde se informa al usuario con los puntos definitorios de la receta incluyendo una descripción, características que cumple y valores nutricionales de la misma:

Ilustración 83. Funcionalidad de resumen (Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Mostrado de información general de la receta
- Mostrado del aporte nutricional de la receta
- Mostrado de categorías a la que pertenece
- Acceso a ingredientes de la receta
- Posibilidad de vuelta a la interfaz de búsqueda
- Acceso al paso a paso de la receta

3.4.2.6 Ingredientes de Receta

Interfaz en la que se muestran al usuario los ingredientes y utensilios necesarios para completar la receta:

Ilustración 84. Funcionalidad Ingredientes (Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Mostrado de ingredientes de la receta
- Posibilidad de retroceso
- Acceso al paso a paso de la receta

3.4.2.7 Paso a Paso

Interfaz donde se van mostrando los distintos pasos de la receta:

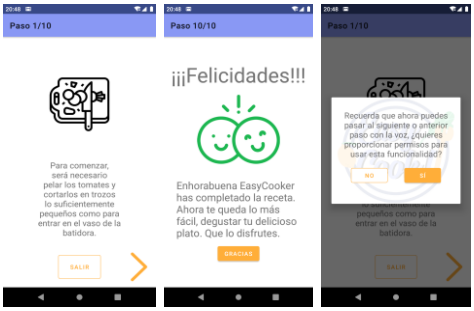


Ilustración 85. Funcionalidad de paso a paso
(Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Salida inmediata
- Configuración de temporizador
- Pantalla "*always-on*"
- Uso de voz para el avance o retroceso
- Pantalla de enhorabuena tras finalizar la receta
- Petición de acceso al micrófono con información de uso de éste

3.4.2.8 Gestión de Temporizadores

Elementos gráficos que permiten la correcta gestión del ciclo de vida completo de los temporizadores de trabajos en segundo plano:

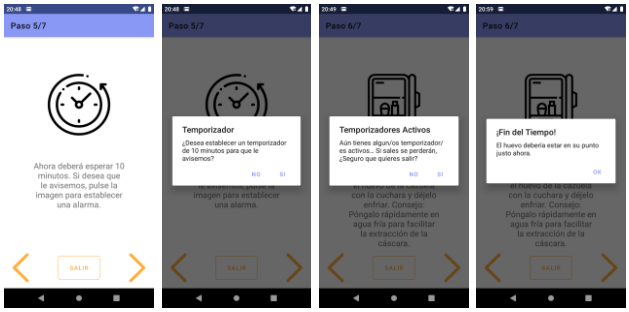


Ilustración 86. Funcionalidad temporizadores
(Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Activación con confirmación del temporizador configurado en back-end
- Chequeo en la salida de temporizadores activos
- Aviso sonoro y visual de fin de temporizador configurado en back-end

3.4.3 General

Además de lo detallado en los apartados anteriores, se establecieron también los siguientes elementos que, aunque no están directamente relacionados con los casos mostrados anteriormente, ayudan a la mejora de la experiencia del cliente.

3.4.3.1 SplashScreen

Punto de entrada a la aplicación, está constituida por un fondo en color sólido con el icono de la aplicación. Su duración está preestablecida a un tiempo fijo, aunque durante la misma, se aprovecha para realizar, como se indicaba anteriormente, la precarga del diálogo comercial:

Ilustración 87. Funcionalidad Splash
(Elaboración propia)

En este caso, las funcionalidades aquí presentes son:

- Precarga del diálogo de negocio

3.4.3.2 Diálogo Comercial

Pequeño boletín informativo mostrado al usuario con la intención de trasladar a éste información que pudiera ser relevante para él. Su modificación es totalmente dinámica en el servidor:

Ilustración 88. Funcionalidad de diálogo comercial
(Elaboración propia)

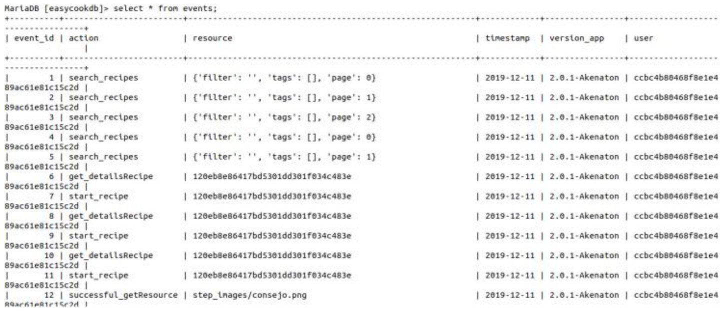
En este caso, las funcionalidades aquí presentes son:

- Mostrado de diálogo de negocio
- Configuración en back-end del diálogo

3.4.3.3 Eventos

Como se observa en la imagen, toda acción realizada en la interacción aplicacion-servidor añade un registro en el que se especifica el tipo de evento, el

recurso al que afecta, el momento en el que sucedió, la versión de aplicación que lo provocó y si procediera, el usuario logueado en ese momento:


MariaDB [easycookdb]> select * from events;

event_id	action	resource	timestamp	version_app	user
1	search_recipes	{'filter': '', 'tags': [], 'page': 0}	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
2	search_recipes	{'filter': '', 'tags': [], 'page': 1}	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
3	search_recipes	{'filter': '', 'tags': [], 'page': 2}	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
4	search_recipes	{'filter': '', 'tags': [], 'page': 0}	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
5	search_recipes	{'filter': '', 'tags': [], 'page': 1}	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
6	get_detailsRecipe	120eb8e86417bd5301dd301f034c483e	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
7	start_recipe	120eb8e86417bd5301dd301f034c483e	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
8	get_detailsRecipe	120eb8e86417bd5301dd301f034c483e	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
9	start_recipe	120eb8e86417bd5301dd301f034c483e	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
10	get_detailsRecipe	120eb8e86417bd5301dd301f034c483e	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
11	start_recipe	120eb8e86417bd5301dd301f034c483e	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d
12	successful_getResource	step_images/consejo.png	2019-12-11	2.0.1-Akenaton	ccbc4b80468f8e1e489ac61e81c15c2d

Ilustración 89. Funcionalidad de captura de eventos (Elaboración propia)

3.5 Plan de Pruebas

Una vez especificadas todas las funcionalidades de la aplicación en el apartado anterior, llega el momento de detallar como fueron validadas durante, y de forma posterior a su implementación.

Aunque la forma más adecuada hubiera sido la elaboración paralelizada de tests unitarios que fueran validando cada una de ellas, la falta de recursos (sólo un desarrollador), la ajustada planificación temporal y la deuda técnica a recuperar, obligaron a la toma de la decisión de realizar estas pruebas de forma manual con cada actualización.

Así, de forma previa a la integración de los dos elementos, y dado que el primer elemento desarrollado fue el *back-end*, se fueron realizando peticiones manuales mediante la herramienta cURL de Linux a cada uno de los endpoints comprobando el buen funcionamiento de éstos y solucionando los problemas que se fueron detectando. Por otro lado, una vez se realizó la integración, de igual forma, se probaron de forma manual, gracias a los dispositivos virtuales Android, las distintas funcionalidades mostradas, comprobando la ausencia de errores y/o inconsistencias. Sin duda, aunque esta forma de proceder no es la más ortodoxa, si que ha demostrado ser efectiva, permitiendo el cumplimiento de los plazos establecidos en la entrega y generando al final del proceso, una solución bastante depurada.

Sin embargo, con el fin de asegurar el futuro de este proyecto, y contextualizándose dentro de un entorno productivo y donde un error podría provocar una pérdida importante de reputación para la compañía o una disminución de clientes activos fruto de la desconfianza, se desea establecer las líneas base de un plan de pruebas consistente, siendo descritas en los siguientes apartados.

3.5.1 Herramientas

Dado que el "mundo" móvil es un entorno cambiante donde los intereses y funcionalidades "*core*" pueden variar prácticamente de una semana a otra, la lógica establece que una vez finalizado el proyecto con la entrega de la versión base de la aplicación, se migrará a un modelo de mantenimiento y mejora más cercano a las denominadas **metodologías agile**, donde los ciclos de desarrollo son más ajustados y

finalizan de forma general con la inclusión de una nueva funcionalidad. Así, y en base a esta afirmación, las herramientas a usar deberían ser automatizadas e incluso, integrables dentro de una metodología de integración continua.

Integración continua es el nombre recibido a un modelo informático que consiste en la disminución de plazos de certificación y prueba de nuevas funcionalidades, apoyándose para ello en automatizaciones, de forma que la subida de un elemento a producción sea lo más habitual posible, sin penalizar por ello a la integridad y/o seguridad del mismo. Así, disponer de un conjunto de pruebas consistente, permitiría que tras la inclusión de una funcionalidad, pudiera testearse de forma automática obteniendo confianza plena en la solidez de esta versión.

Una vez claros estos conceptos, se decidieron usar las siguientes herramientas para la realización automática de tests:

- *Back-end*: Aunque la materia de testing de APIs consta de un gran número de opciones a la hora de automatizar estas tareas, debido a las particularidades del producto desarrollado, las opción elegida será un desarrollo *ad hoc*. Los motivos que llevan a ello es, que aunque gran parte de los endpoints podrían certificarse con una herramienta automática como Postman, toda la funcionalidad de recogida de eventos, al ser interna e inaccesible vía API, no podría asegurarse.
- *Front-end*: En este caso, aunque se posee la particularidad del uso de la voz, asunto para el cual, el número de alternativas automáticas de test es casi inexistente, esta integración no debería sufrir grandes alteraciones, por lo que se podría utilizar una herramienta comercial general. Así, se piensa que la opción adecuada sería usar una combinación de las siguientes herramientas:
    - JUnit: Librería open-source de pruebas automáticas para el lenguaje Java ampliamente extendida. Su uso será el de construir pruebas unitarias sobre funciones Java, que validen su correcto funcionamiento.
    - Espresso: Librería open-source para la ejecución de pruebas automatizadas sobre la interfaz gráfica Android. Su uso será el de ejecutar pruebas que evalúen el correcto funcionamiento de la interfaz construida, chequeando cada una de las posibilidades de la misma y reportando posibles incongruencias que se produzcan sobre ella.

Los motivos que han llevado al uso de estas herramientas han sido principalmente su integración con el entorno de desarrollo elegido (Android Studio), lo que permitiría el uso de dispositivos virtuales Android (AVD) y el gran apoyo comunitario que poseen en Internet.

3.5.2 Pruebas

Según los actuales estándares de la industria, toda aplicación debería contener un conjunto de pruebas amplio para cada uno de los siguientes tipos:

- Pruebas unitarias: Comprueban la funcionalidad de la unidad mínima de código, es decir, cada elemento (función) de forma independiente.
- Pruebas de integración: Aseguran el funcionamiento de los componentes evaluados en las pruebas unitarias cuando éstos, son puestos a funcionar en conjunto.
- Pruebas de instrumentación: Aquellos más cercanos al uso final del cliente, pues se encargan de simular las posibles interacciones evaluando los resultados de las mismas. Aunque se encuentran diferenciadas de las

unitarias, este tipo de validaciones podrían considerarse como tales, pues, realmente, evalúan el comportamiento de cierto componente de la interfaz tras simular una interacción del usuario.

Así, una vez estuvieron claras las pruebas que deberían existir a futuro en el proyecto y con la intención de establecer un precedente estable, se desarrollaron los tests mostrados a continuación.

3.5.2.1 T001_nav_bottom_menu

El objetivo de este conjunto de test es el de comprobar que el menú de navegación inferior de la aplicación efectivamente cambia el fragment mostrado al usuario. Para ello, se implementó el siguiente test "*doNavigation()"* el cual, se movía de la pestaña de búsqueda a la de usuario y al revés:

```
@RunWith(AndroidJUnit4.class)
@SmallTest
public class T001_nav_bottom_menu {

    @Rule
    public ActivityTestRule<MainActivity> activityTestRule = new ActivityTestRule<>(MainActivity.class);

    @Test
    public void doNavigation(){

        onView(withId(R.id.navigation_user)).perform(click());
        onView(withId(R.id.textfield_account)).check(matches(isDisplayed()));
        onView(withId(R.id.navigation_search)).perform(click());
        onView(withId(R.id.recycler_view_recipe)).check(matches(isDisplayed()));
    }

}
```

Ilustración 90. Implementación T001
(Elaboración propia)

3.5.2.2 T002_checkErrorsLogin

El objetivo de este conjunto de test es el de comprobar todos los errores existentes en la interfaz de inicio de sesión. Para ello, se generaron tres tests distintos: *"checkUsernameEmpty()"*, "*checkPasswordEmpty()"* y "*checkErrorConnection"*, los cuales comprobaban el mostrado de error por campo de usuario vacío, el error por campo de contraseña vacío y la no conexión con el servidor, respectivamente.

```
/* Test encargado de chequear la aparición de errores en apartado de login*/

@RunWith(AndroidJUnit4.class)
@MediumTest
public class T002_checkErrorsLogin {

    @Rule
    public ActivityTestRule<LoginActivity> activityTestRule = new ActivityTestRule<>(LoginActivity.class);

    @Test
    public void checUsernameEmpty(){

        String userEmptyMessage = "El nombre de usuario no puede ser vacío";

        onView(withId(R.id.btn_login)).perform(click());
        onView(withId(R.id.username)).check(matches(hasErrorText(userEmptyMessage)));

        onView(withId(R.id.username)).perform(replaceText( stringToBeSet: "userTest"));

        onView(withId(R.id.btn_login)).perform(click());
    }

    @Test
    public void checkPasswordEmpty() {

        String passEmptyMessage = "La contraseña no puede ser vacía";


        onView(withId(R.id.btn_login)).perform(click());
        onView(withId(R.id.password)).check(matches(hasErrorText(passEmptyMessage)));

        onView(withId(R.id.password)).perform(replaceText( stringToBeSet: "passTest"));

        onView(withId(R.id.btn_login)).perform(click());

    }

    @Test
    public void checkErrorConnection(){

        onView(withId(R.id.username)).perform(replaceText( stringToBeSet: "userTest"));
        onView(withId(R.id.password)).perform(replaceText( stringToBeSet: "passTest"));

        onView(withId(R.id.btn_login)).perform(click());

        onView(withText("¡Vaya! Parece que no puedo conectar con el sistema. Com..."))
                .inRoot(withDecorView(not(is(activityTestRule.getActivity().getWindow().getDecorView()))))
                .check(matches(isDisplayed()));
    }


}
```

Ilustración 91. Implementación T002
(Elaboración propia)

3.5.3 Resultados

Una vez implementados, se procedió a la ejecución de los mismos, arrojando resultados satisfactorios tras su ejecución:

Ilustración 92. Resultado Tests
(Elaboración propia)

4. Conclusión y líneas futuras

Para finalizar la presente memoria y como cierre al proyecto, se presentan en este capítulo todas aquellas conclusiones obtenidas en este trabajo, tanto en el plano de gestión del proyecto como más cercanas al proceso de desarrollo, además de aquellas posibilidades de futuro de la aplicación desarrollada.

4.1 Conclusión

En este primer subapartado se presenta un análisis exhaustivo de todo el proyecto en términos generales, así como las principales conclusiones obtenidas durante la ejecución del mismo.

Para comenzar, como reflexión personal, el presente proyecto me ha servido para reforzar y afianzar muchos de los conceptos estudiados a lo largo del programa formativo del máster, aportándome la confianza necesaria para su aplicación en el entorno laboral. Como máxima lección aprendida, considero que ha sido cómo una buena planificación de proyecto y una buena iniciación son claves a la hora de aumentar las probabilidades de éxito en un proyecto, lección, que a nivel personal creo, no está afianzada en la actualidad, donde la desvirtuación de las llamadas metodologías ágiles ha provocado proyectos sin rumbo y sin un sustento estable.

Sobre este **plano más cercano a la gestión**, obviamente, tras realizar un proceso de análisis, se han observado puntos de mejora de cara a proyectos futuros:

- Error en la estimación de la fase de prototipado: Como se ha expresado anteriormente, durante esta etapa se decidió implementar el prototipo directamente en XML para asegurar su posterior integración en Android. Esta decisión aunque a medio-largo plazo resultó acertada, obligó a un sobreesfuerzo considerable en esta fase para conseguir cumplir con los plazos establecidos por el hito H003.
- Incorrección sobre la temporización de la fase de pruebas DCU: Aunque a priori, establecer la fase de pruebas antes de la implementación final parecía la decisión correcta, la ejecución del proyecto y las características del mismo, determinaron lo contrario. El hecho de que la aplicación contara con el uso de la voz como elemento diferenciador, obligaba a realizar una fase de implementación previa, para asegurar la correcta usabilidad de esta opción. Igualmente, no contemplar los conocimientos adquiridos durante la formación en el plano de usabilidad (por falta de confianza en los mismos), hicieron que no se valoraran adecuadamente como un primer análisis de usabilidad a ejecutar justo después del prototipado DCU, sirviendo como una valoración real del producto en primera versión, listo para su posterior implementación.

Respecto al **plano desarrollador**, pienso que hay dos lecciones aprendidas a destacar:

- Uso de librerías como catalizador: El uso de librerías y su establecimiento como requisito no funcional del proyecto, ayudaron a acelerar la fase de implementación permitiendo la implementación de puntos no establecidos inicialmente y el perfeccionamiento/mejora de los ya establecidos, lo que dió al producto un aspecto mucho más comercial del inicialmente pensado.
- Importancia de la automatización de pruebas: Aunque durante el proyecto no se hizo, en futuros proyectos este punto será impuesto como requisito no funcional de los mismos. Durante la ejecución de este proyecto, como se mencionó en el apartado correspondiente, las pruebas de validación han sido

ejecutadas de forma manual debido al miedo a retrasos que la deuda técnica existente en este aspecto pudiera provocar. No obstante, tras la ejecución de la fase de pruebas del proyecto, considero que esta decisión fue un error del proyecto, pues, al menos, las pruebas unitarias, podrían haberse automatizado ahorrando una gran cantidad de tiempo de implementación.

Para terminar este apartado, en **aspectos globales del proyecto**, se consideran los **objetivos del mismo cumplidos de forma satisfactoria**. Aunque la consecución de los mismos, como se ha indicado anteriormente tiene puntos de mejora, los objetivos en sí han sido conseguidos obteniendo un producto usable, eficaz y robusto. No obstante, aunque el producto ha recibido comentarios positivos del entorno, el objetivo "*OBJ007 - Inculcar hábitos saludables a la población*", necesitará de una puesta en producción a gran escala para poder ser valorado de forma adecuada.

4.2 Líneas Futuras

En este apartado se presentan las lineas futuras del producto, que como se observará a continuación, son bastante amplias, al ser un producto de grandes posibilidades:

- Monetización: Obviamente, como paso natural de cualquier producto, la obtención de beneficios del mismo debe ser un punto importante. Así, las opciones de esta aplicación, tras el estudio de usabilidad, se reducen al modelo de subscripción. Podría realizarse un modelo parecido al de competidores como Thermomix donde los usuarios pudieran adquirir por tiempo limitado o ilimitado paquetes de recetas exclusivas además de las de por defecto.
- Personalización: En la actualidad, la mayoría de softwares destinados a servicios optan por la personalización al consumidor haciendo que la usabilidad y experiencia de usuario percibida sean mucho mayores. En este caso, al haber integrado en el producto métricas de inteligencia de negocio y una primera aproximación a este comportamiento (número de recetas totales y receta favorita), se debería optar por trabajar estos datos e incluir nuevos para por ejemplo, a un usuario acostumbrado a buscar recetas vegetarianas, ofrecerle recomendaciones de este tipo de comida.
- Feedback: Facilitar mediante la inclusión de comentarios públicos o privados el envío de opiniones de los clientes, de forma que se detecten carencias (por ejemplo, una receta cuyos tiempos están mal medidos) u oportunidades de negocio (por ejemplo, nuevas recetas).
- Socialización: Con el auge actual de las redes sociales, establecer funcionalidades en esta dirección otorgaría una visión más moderna y actual de la aplicación. Estas funciones, por ejemplo, podrían ser indicar gusto por una receta, compartirla o guardarla para más tarde.
- Importación de recetas: La importación automática tras revisión manual, ayudaría a un crecimiento mucho más rápido del catálogo de recetas lo que ampliaría las posibilidades de la aplicación.
- Paso a producción: Aunque la solución desarrollada constituye un buen punto inicial, su paso a producción requerirá de una serie de tareas para asegurar la estabilidad y robustez de la misma. Estas tareas serán la securización (pues se dejó a un lado en el proyecto para simplificarlo), la división de la arquitectura (autenticador, APIs y servidor de recursos en distintos componentes), la redundancia de componentes, construcción de una solución de monitorización efectiva y aseguramiento de carga mediante pruebas de estrés.

5. Glosario

- **Android Virtual Device (AVD)**: Configuración que define las características necesarias para emular el comportamiento de cierto dispositivo móvil.

- **Application Programming Interface (API)**: Conjunto de métodos provisionados por una aplicación para su uso por terceros a modo de capa de abstracción.

- **Back-End:** Parte no visible de una aplicación. Normalmente relacionado con grandes servidores con bases de datos y apificadores.

- **Brecha Digital**: Distancia en el acceso, uso y apropiación de las tecnologías tanto a nivel geográfico, a nivel socioeconómico y otras desigualdades culturales, etc. (2019, Wikipedia).

- **Bug**: Denominación informática para un error de implementación de cierto componente.

- **Business Intelligence (BI, Inteligencia de negocio)**: Conjunto de estratégias, técnicas, metodologías y herramientas destinadas a la extracción de información de los datos obtenidos de los usuarios.

- **Endpoint**: Interfaz de comunicación expuesta al exterior desde la cual un cliente puede acceder a ciertos servicios computacionales.

- **Entorno de desarrollo integrado (IDE):** Aplicación informática que presta todos los servicios necesarios para el desarrollo de software.

- **Experiencia de usuario (*User Experience*):** Denominación dada a la percepción del usuario al manejar o utilizar un elemento.

- **Extensible Markup Language (XML)**: Lenguaje de marcado especificado por W3C usado habitualmente en la codificación de elementos cuando son enviados/recibidos a través de la red.

- **Front-End**: Parte visible de la aplicación. Normalmente relacionado con un interfaz gráfico residente en un dispositivo o web.

- **Hipervisor**: Plataforma para la aplicación de técnicas de virtualización de computadoras.

- **Lenguaje de Consulta Estructurada (SQL)**: Lenguaje de dominio específico usado para la gestión de datos en entornos de bases de datos relacionales.

- **Memoria de acceso aleatorio (RAM)**: Memoria volátil parte de un computador. Este tipo de memoria suele usarse para el cacheado de datos pues, el acceso a estos, es mucho más rápido que por ejemplo, si se accedería al disco duro.

- **Pensamiento manifiesto:** Se denomina así cuando un individuo transmite de forma comunicativa exterior parte de los pensamientos que está teniendo.

- **Representational State Transfer (REST):** Tipo de arquitectura software usada ampliamente en la web. Dispone de métodos bien definidos para la extracción, modificación, creación y borrado de objetos.

- **Smartphone**: Dispositivo móvil “inteligente”, evolución del tradicional, el cual añade multitud de funciones además de la llamada y la mensajería.

- **Software Development Kit (SDK)**: Conjunto de herramientas prestadas por una entidad para el desarrollo de software para un sistema concreto.

- **Unidad Central de Procesamiento (CPU)**: Hardware de un computador dedicado a la interpretación de instrucciones de un programa y su traducción a operaciones aritméticas, lógicas o de entrada/salida.

- **Unified Modelling Language (UML)**: Lenguaje gráfico de modelado para sistemas y/o softwares más expandido en la actualidad gracias a su versatilidad y adaptación.

- **Usabilidad**: Característica de un sistema o aplicación para ser fácil de usar e intuitiva aun cuando este no haya sido usado con anterioridad.

- **Web Server Gateway Interface (WGSI)**: Convención de llamada simple para que los servidores web reenvíen peticiones a aplicaciones web escritas en Python.

6. Bibliografía

[1] Statista (2017). "*Penetración mensual de la telefonía móvil sobre la población en España de enero de 2016 a diciembre de 2018 (líneas por cada 100 habitantes)".* Disponible en: https://es.statista.com/estadisticas/477127/tasa-penetracion-telefonia-movil-mensual-espana/ (Consultado en 02/10/2019).

[2] Ditrendia (2017). "*Informe Ditrendia Mobile en España y en el Mundo 2017*". Disponible en: https://www.amic.media/media/files/file_352_1289.pdf (Consultado en 02/10/2019).

[3] Wikipedia (2019, 1 de octubre). "*Desarrollo en Cascada*". Disponible en: https://es.wikipedia.org/wiki/Desarrollo_en_cascada#Validaci%C3%B3n_y_Verificaci%C3%B3n_del_producto_de_software (Consultado en 02/10/2019).

[4] Wikipedia (2019, 2 de octubre). *"Proceso para el desarrollo de software*". Disponible en: https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software#Modelo_de_espiral (Consultado en 02/10/2019).

[5] José Ramón Rodríguez. *Módulos académicos de la asignatura Gestión Avanzada de Proyectos TI* [Versión electrónica], Barcelona, Universitat Oberta de Catalunya. Recuperado de: https://cv.uoc.edu (Consultado en 02/10/2019).

[6] X-Wiki UOC (2013, 22 de mayo). "*Indagación". Disponible en: http://cv.uoc.edu/webapps/xwiki/wiki/matm1202es/view/Main/6.1+Indagaci%C3%B3+%5Ban%C3%A0lisi%5D (Consultado en 30/10/2019).*

[7] Juan Ramón Ruíz (2012, 4 de diciembre). "*Diseño centrado en el usuario (DCU). Diseño de escenarios y flujos de interacción*". Disponible en: http://juanramonruiz.com/diseno-centrado-en-el-usuario-dcu-diseno-de-escenarios-y-flujos-de-interaccion/ (Consultado en 30/10/2019).

[8] Wikipedia (2019, 25 de octubre). "*Brecha digital*". Disponible en: https://es.wikipedia.org/wiki/Brecha_digital (Consultado en 30/10/2019).

[9] Amaia Calvo-Fernández Rodríguez, Sergio Ortega Santamaría, Alicia Valls Saez. *Módulos académicos de la asignatura Ingeniería de la Usabilidad* [Versión electrónica], Barcelona, Universitat Oberta de Catalunya. Recuperado de: https://cv.uoc.edu (Consultado en 30/10/2019).

[10] The Pallets Projects. "Flask". Disponible en: https://www.palletsprojects.com/p/flask/ (Consultado en 10/12/2019).

[11] Python.org. "Sunsetting Python 2". Disponible en: https://www.python.org/doc/sunset-python-2/ (Consultado en 10/12/2019).

[12] Genbeta (2017, 3 de noviembre). "Python se ha convertido en el lenguaje de programación que crece más rápido". Disponible en: https://www.genbeta.com/actualidad/python-se-ha-convertido-en-el-lenguaje-de-programacion-que-crece-mas-rapido (Consultado en 10/12/2019).

[13] MariaDB Foundation. "MariaDB Server: The open source relational database". Disponible en: https://mariadb.org/ (Consultado en 10/12/2019).

[14] Atom.io. "A hackable text editor for the 21st Century". Disponible en: https://atom.io/ (Consultado en 10/12/2019).

[15] JWT.io. "JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties". Disponible en: https://jwt.io/ (Consultado en 10/12/2019).

[16] Wikipedia (2019, 29 de octubre). "Android Studio". Disponible en: https://es.wikipedia.org/wiki/Android_Studio (Consultado en 10/12/2019).

[17] Xataka Móvil (2019, 16 de abril). "Android supera el 90% de cuota en España mientras que iOS cae por debajo del 9%, según Kantar". Disponible en: https://www.xatakamovil.com/mercado/android-supera-90-cuota-espana-ios-cae-debajo-9-kantar (Consultado en 10/12/2019).

[18] Android Developers. "Arquitectura de la plataforma". Disponible en: https://developer.android.com/guide/platform?hl=es-419 (Consultado en 10/12/2019).

[19] Android Developers. "Distribution dashboard". Disponible en: https://developer.android.com/about/dashboards (Consultado en 10/12/2019)

[20] Wikipedia (2019, 28 de noviembre). "Java (lenguaje de programación)". Disponible en: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n) (Consultado en 10/12/2019).

[21] GitHub (2019, 9 de diciembre). "Retrofit". Disponible en: https://github.com/square/retrofit (Consultado en 10/12/2019).

[22] Medium (2016, 21 de octubre). "Glide vs. Picasso". Disponible en: https://medium.com/@multidots/glide-vs-picasso-930eed42b81d (Consultado en 10/12/2019).

[23] Material.io. "Material Design". Disponible en: https://material.io/ (Consultado en 10/12/2019).

[24] Oracle. "Modelado de datos con Oracle SQL Developer". Disponible en: https://www.oracle.com/es/database/technologies/appdev/datamodeler.html (Consultado en 10/12/2019).

[25] Wikipedia (2019, 15 de noviembre). "VMware Workstation". Disponible en: https://en.wikipedia.org/wiki/VMware_Workstation (Consultado en 10/12/2019).

[26] Git. "Git". Disponible en: https://git-scm.com/ (Consultado en 10/12/2019).

[27] Auth0 Docs. "Auth0.Android". Disponible en: https://auth0.com/docs/libraries/auth0-android (Consultado en 10/12/2019).

[28] Android Developers. "Documentation for app developers". Disponible en: https://developer.android.com/docs (Consultado en 10/12/2019).

[29] Python.org (2019, 10 de diciembre). "time — Time access and conversions". Disponible en: https://docs.python.org/3/library/time.html (Consultado en 10/12/2019).

7. Anexos

En este apartado, se muestran los anexos adjuntos y autocontenidos de este documento. Estos documentos son los siguientes:

- Anexo I - Documentos de Evaluación
- Anexo II - Documentos rellenos tres test de usuarios
- Anexo III - Respuestas Entrevista
- Anexo IV - Manual de Instalación EasyCook!
- Anexo V - Manual de Usuario EasyCook!