Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Allerga: Aplicación de análisis de alérgenos

Allerga: Allergen analysis application

Alejandro Medina García

La Laguna, 04 de julio de 2022

D. **Francisco Javier Rodríguez González**, con N.I.F. 43.618.712-V profesor asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Alejandro Pedro Pérez Nava**, con N.I.F. 43.821.179-S profesor asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)
Que la presente memoria titulada:
"Allerga: Aplicación de análisis de alérgenos"

ha sido realizada bajo su dirección por D. **Alejandro Medina García**, con N.I.F. 79.087.949-B.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 04 de julio de 2022.

Agradecimientos

Antes de nada, quería agradecer a mi familia y amigos, por apoyarme a lo largo de estos años, pero sobre todo a mis padres, que no han dejado de confiar en mí en todo momento. Especialmente quería también agradecer a María, por su ayuda y apoyo incondicional, y a mi tutor Javier, por toda la ayuda prestada estos últimos meses.

Licencia

Resumen

Actualmente, las personas que presentan alérgenos alimentarios tienen muchas dificultades a la hora de comprar productos en el día a día. Este problema se extiende a familiares y amigos, que desconocen aún más sus necesidades y restricciones.

Partiendo de esta situación inicial, se ha realizado un prototipo de aplicación (***Allerga***), cuya principal función es, tras configurar el/los alérgeno/os que padece la persona objetivo, informar mediante una lectura de código de barras si un alimento es apto o no para su consumo, ahorrando el tener que buscar en la etiqueta si posee dicho alérgeno o no.

Para ello, se ha hecho uso de una base de datos abierta perteneciente a **Open Food Facts**. Dicha base de datos se constituye del conocimiento generado a partir de usuarios de la página, los cuales registran datos de forma individual. En una primera instancia, se ha utilizado la API proporcionada por la misma página, pero se implementó la posibilidad de que la aplicación funcionase (con menos características y/o funcionalidades) sin conexión alguna, abordando el problema de que se dé el caso hipotético de que en un supermercado no llegue la señal de red.

La idea principal era crear una aplicación móvil, pero se acabó haciendo también una versión web, la cual cumple la función con un buscador, en vez de con el escáner.

Palabras clave: comida, alérgenos, aplicación, API, escáner.

Abstract

Currently, people with food allergens have many difficulties buying products on a day-to-day basis. This problem extends to family and friends, who are even more unaware of their needs and restrictions.

Starting from this initial situation, an application prototype (***Allerga***) has been made, whose main function is, after configuring the allergen(s) suffered by the target person, to inform by means of a barcode reading, if a food is suitable or not for its consumption, saving having to look on the label if it has said allergen.

For this task, an open database belonging to **Open Food Facts** has been used. Said database is made up of the knowledge generated from users of the page, who record data individually. In the first instance, the API provided by the same page was used, but the possibility was implemented for the application to work (with fewer features and / or functionalities) without any connection, addressing the problem of the hypothetical case of that in a supermarket the network signal does not reach.

The main idea was to create a mobile application, but a web version was also created, which fulfills the function with a search engine, instead of with the scanner.

Keywords: food, allergens, application, API, scanner.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Motivación personal

1.1.1 Casos cercanos personales

Actualmente en mi vida tengo mucho contacto con personas que padecen alergias o intolerancias alimentarias, tanto familiares como amigos, la más cercana, mi hermana. Al hacer la compra para cada uno de estos grupos, numerosas veces he tenido complicaciones a la hora de elegir qué productos puedo comprar, ya que cada artículo que se pone en la cesta de la compra tiene que pasar antes un filtro de lectura de la etiqueta. Mi madre, cansada de llevar gafas de cerca al supermercado, fue la que me dio la idea de realizar este proyecto, haciendo útil el desarrollo de este, independientemente de su posterior seguimiento.

1.1.2 Proyecto full stack

La propuesta inicial de mi tutor consistía en la realización de un **proyecto full stack**[1], con la única condición añadida de que trabajase de alguna forma con **Open Data**[2]. Al final, el tema de dicho proyecto derivó a una idea que tenía previamente a la elección de este, pero mi principal factor de decisión para escoger su propuesta, no se basaba en el desarrollo de mi propia idea, sino por el concepto de programación de full stack. A lo largo de la carrera, de todos los campos con los que he trabajado, me he dado cuenta de que este sector es el que más me gusta dentro de la informática. Al finalizar el grado, este es el camino que tenía pensado seguir, y me atraía el reto de formarme por mi cuenta en tecnologías nuevas de cara al mundo laboral, ya que nunca he tocado muchas de estas que se han acabado utilizando. Además, siempre me ha atraído el mundo del desarrollo de aplicaciones móviles, y esta era una muy buena ocasión para adentrarme en este.

1.2 Definición del problema

La problemática por resolver es la que enfrentan día a día las personas que padecen algún tipo de alergia alimentaria, simplificando tareas laboriosas, como es tener que leer cada una de las etiquetas de los productos. Como explica el artículo

[1] Proyecto que cubre cada uno de los aspectos relacionados con la creación y el mantenimiento de una aplicación web.
[2] Base de datos gratuita.

"*Alergias alimentarias: Importancia del control de alérgenos en alimentos"*[1], distintos estudios sobre la prevalencia de esta hipersensibilidad señalan que es un problema de salud pública en auge, que afecta en torno al 1-3 % de la población adulta y al 4-7 % de la población infantil. Gracias a este artículo se puede ver que el avance de las personas que padecen alérgenos alimentarios está en crecimiento, debido a muchos factores, varios de ellos comentados por la BBC en su artículo *"Por qué están aumentando las alergias alimentarias"*[2].

1.3 Actividades a realizar

- **A1. Estudio de los alérgenos:** Estudio sobre los diferentes alérgenos alimentarios existentes e importancia. Acotación del problema y verificación de necesidad de ser resuelto.

- **A2. Estudio de las bases de datos:** Búsqueda de una base de datos que cumpla los requisitos del proyecto.

- **A3. Estudio de las tecnologías:** Estudio sobre qué tecnologías utilizar en el proyecto.

- **A4. Hola mundo de la aplicación:** Primera versión funcional de la aplicación con el sistema de navegación entre pantallas.

- **A5. Pantalla principal:** Creación de la pantalla principal de la aplicación.

- **A6. Pantalla de configuración:** Creación de configuración de alérgenos consistente en una base de datos interna de la aplicación.

- **A7. Pantalla de ayuda:** Creación de un apartado de ayuda con indicaciones sobre el funcionamiento de la aplicación.

- **A8. Pantalla de escaneo:** Creación de la funcionalidad del escaneo de productos junto a su conexión a la API.

- **A9. Pantalla de historial:** Creación de la pantalla de historial para mantener durante cierto tiempo información de los productos escaneados.

- **A10. Funcionalidad offline:** Implementación de la funcionalidad demo sin conexión a internet.

- **A11. Versión de escritorio:** Ajustes para la creación de una versión demo de escritorio.

1.4 Plan de trabajo: Trabajo Fin de Grado

La fecha de inicio del desarrollo del proyecto fue el 23 de febrero de 2022, siendo la fecha de finalización el 6 de junio de 2022. Se ha realizado un **diagrama de Gantt**[4](**1.1**) gracias a **Microsoft Project**[3] según las actividades a realizar previamente explicadas.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Allerga	**104 días**	**mié 23/02/22**	**lun 06/06/22**	
Fase de investigación	**14 días**	**mié 23/02/22**	**mar 08/03/22**	
Estudio de las tecnologías	14 días	mié 23/02/22	mar 08/03/22	
Estudio de las bases de datos	14 días	mié 23/02/22	mar 08/03/22	
Plan para actualizar personalizaciones	14 días	mié 23/02/22	mar 08/03/22	
Fase de desarrollo	**90 días**	**mié 09/03/22**	**lun 06/06/22**	**2**
Hola mundo y primeros pasos	7 días	mié 09/03/22	mar 15/03/22	
Pantalla principal	7 días	mié 16/03/22	mar 22/03/22	7
Pantalla de configuración	10 días	mié 23/03/22	vie 01/04/22	8
Pantalla de ayuda	4 días	sáb 02/04/22	mar 05/04/22	9
Pantalla de escaneo	30 días	mié 06/04/22	jue 05/05/22	10
Pantalla de historial	15 días	vie 06/05/22	vie 20/05/22	11
Funcionalidad offline	14 días	sáb 21/05/22	vie 03/06/22	12
Versión de escritorio	3 días	sáb 04/06/22	lun 06/06/22	13

Tabla 1.1: Planificación del desarrollo del proyecto

[3] https://www.microsoft.com/es-es/microsoft-365/project/project-management

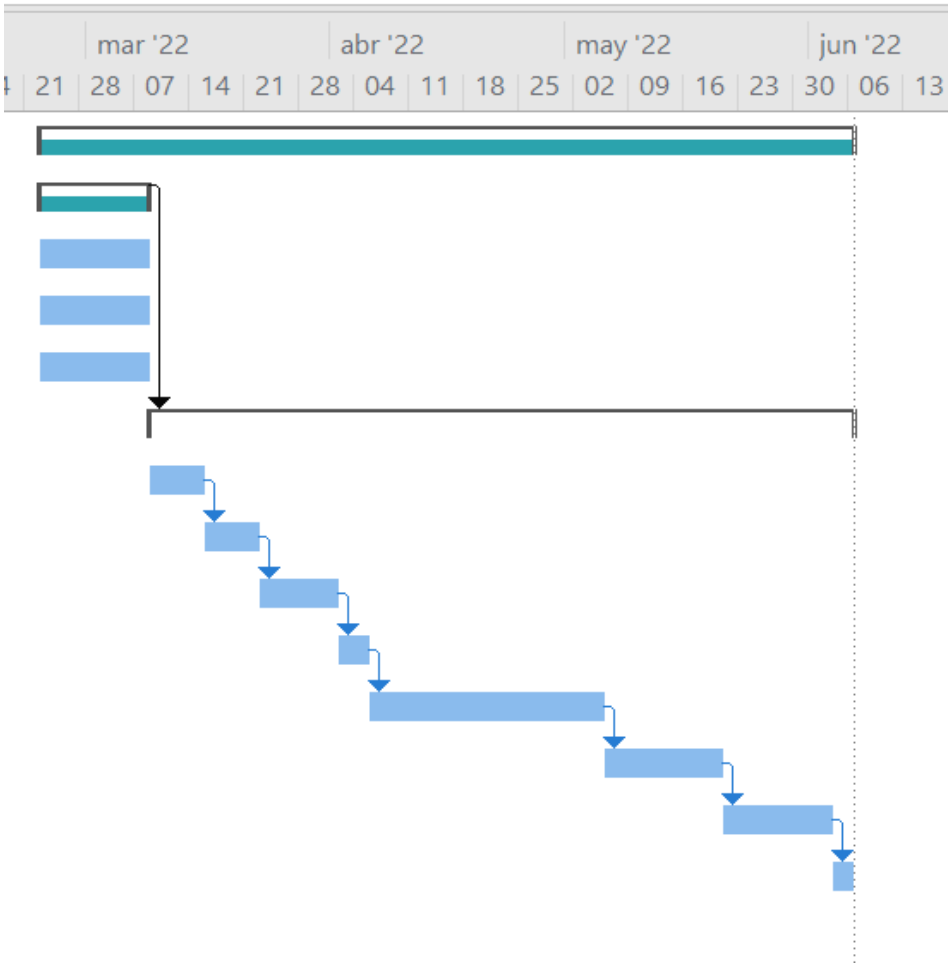

Figura 1.1: Gráfico de Gantt generado por la tabla de planificación del proyecto

Capítulo 2

Antecedentes y estudio previo

Se ha completado un estudio de las aplicaciones similares al aplicativo propuesto que existen en el mercado, junto a un estudio de incidencia de la problemática a la que se enfrenta el proyecto.

2.1 Yuka

La aplicación que más funcionalidades tiene, y la aplicación de escaneo de productos predilecta por el público que busca este tipo de contenido, es **Yuka**[4]. Surge de un proyecto similar, en el que la principal funcionalidad es, al escanear un producto, indicar lo saludable que es este, con un rango de puntuación de 0 a 100. Como comentan en un artículo del apartado de ayuda de su página[5], al principio hicieron uso, al igual que este proyecto, de la API perteneciente a **Open Food Facts**[5], que se trata de una **base de datos abierta colaborativa**, actualizada por el conocimiento aportado por los usuarios. Yuka se desvinculó de esta fuente de datos abierta para poder poner en marcha *"avanzados sistemas de control y verificación de contribuciones"*. Seguidamente, ampliaron su rango de productos para valorar la calidad de los cosméticos. Junto a sus funcionalidades, también indica los alérgenos que posee un producto, entre otra inmensa cantidad de información que puede llegar a saturar al público.

Yuka es una opción completa para todo tipo de usuarios, pero dista de la idea de este proyecto: una aplicación sencilla, configurable y accesible en gran medida, que cumpla un propósito específico sin saturar a su público.

2.2 ¡Bueno para mí!

¡Bueno para mí! posee una configuración personal de alérgenos y cumple la misma funcionalidad principal que Allerga. Sin embargo, a diferencia de Yuka, esta aplicación está muy poco pulida, tanto en accesibilidad como optimización e interfaz. Solo hay que echar un vistazo a su página web oficial[6] para observar que el proyecto

[4] https://yuka.io/es/
[5] https://es.openfoodfacts.org/
[6] http://www.buenoparami.com/

está muy lejos de ser perfecto, dando la sensación incluso de estar en gran medida incompleto.

Posee una interfaz(**2.1**) liosa y desorganizada, muy poco accesible, con colores demasiado llamativos, llena de bugs, muy lenta, con demasiada información inútil en pantalla, transiciones entre pantallas muy toscas, con funciones excesivas como poder mandar mensajes entre usuarios, sin fotos de productos, etc.

Figura 2.1: Ejemplo de escaneo de leche con lactosa siendo alérgico a esta en *¡Bueno para mí*!

2.3 Estudio de alérgenos e incidencia

Se ha realizado un estudio previo a la realización del proyecto en sí, con el objetivo de cerciorarse de que se está tratando con una necesidad real, dado un problema muy común y globalizado. Varios de los artículos consultados son los

anteriormente mencionados en el punto 1.2, siendo estos “Alergias alimentarias: Importancia del control de alérgenos en alimentos”[1] y “Por qué están aumentando las alergias alimentarias”[2].

2.3.1 RASFF

Los artículos que más se han tenido en cuenta en este estudio previo, han sido los del **RASFF**[7] (**Rapid Alert System for Food and Feed**). Se trata de una herramienta de la Comisión Europea[8], encargada de aportar información sobre productos alimenticios, junto a información de procedimiento en caso de ingesta de alérgenos. En concreto, los artículos que más se han tenido en cuenta han sido los resúmenes anuales, particularmente en el último publicado de 2020[6].

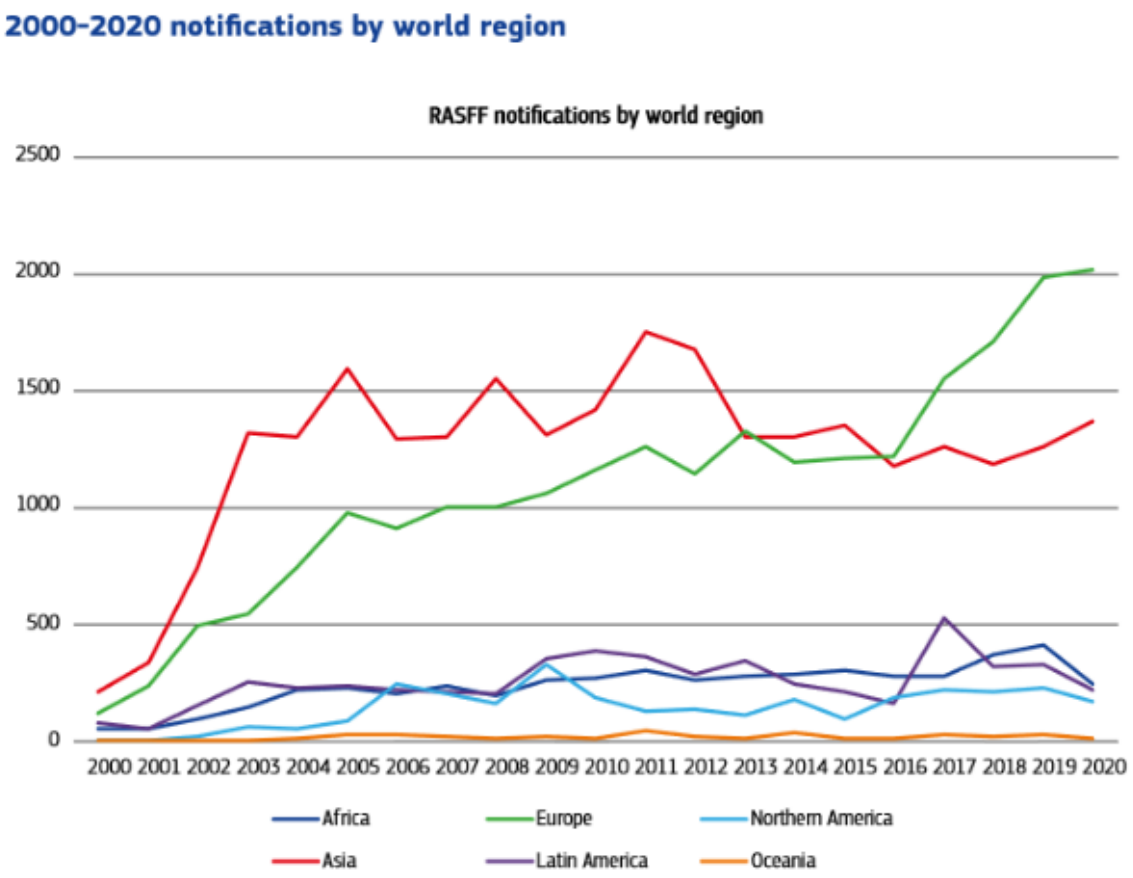

Figura 2.2: Notificaciones del RASFF en 2020 dividida por continentes

[7] https://ec.europa.eu/food/safety/rasff-food-and-feed-safety-alerts_en
[8] Una de las siete instituciones de la Unión Europea. Ostenta el poder ejecutivo y la iniciativa legislativa.

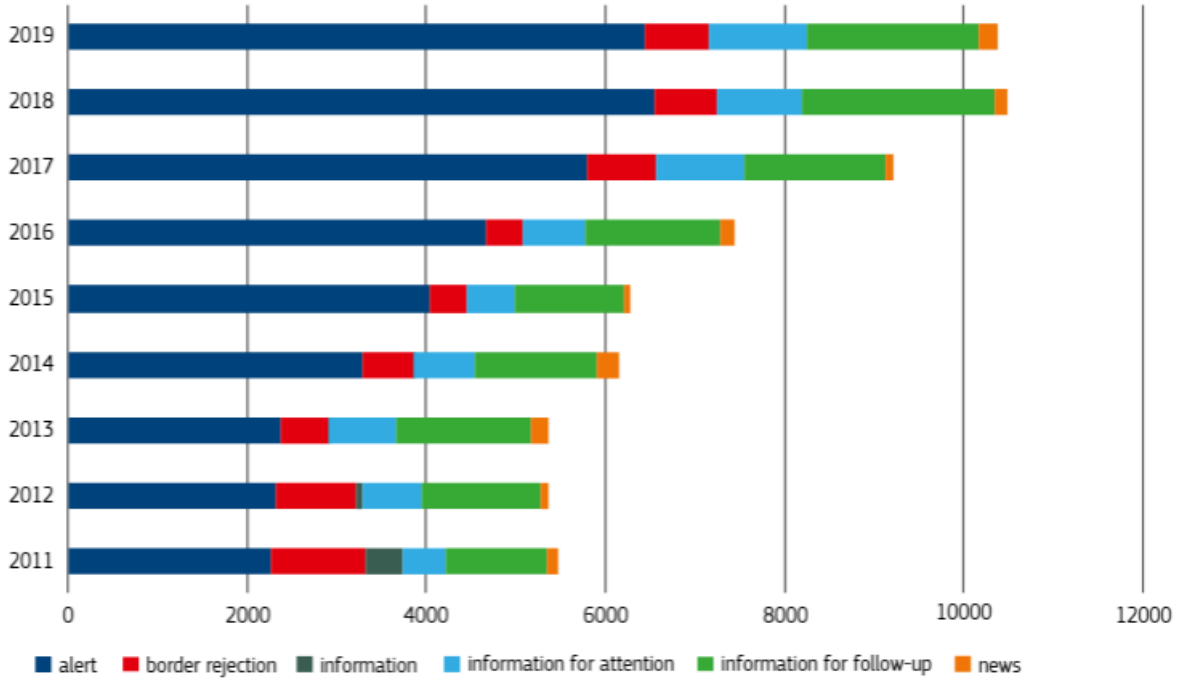

Figura 2.3: *Notificaciones del* RASFF en 2019 dividida por tipos

Los alérgenos que se han tenido en cuenta a implementar en la aplicación son los incluidos en el *anexo II del Reglamento (UE) nº 1169/2011 del Parlamento Europeo*[7], en su propósito de perseguir un alto nivel de protección de la salud de los consumidores y garantizar su derecho a la información. Estos alérgenos son:

- Leche
- Dióxido de azufre y sulfitos
- Soja
- Cereales que contengan gluten (trigo, centeno, cebada, avena, espelta o sus variedades híbridas)
- Cacahuetes
- Crustáceos
- Huevos
- Pescado
- Frutos con cáscara: almendras, avellanas, nueces, anacardos, pacanas, nueces de Brasil, pistachos o alfóncigos, macadamias o nueces de Australia
- Apio
- Mostaza
- Sésamo
- Moluscos
- Altramuces

Capítulo 3

Herramientas y tecnologías

Github 

Git[9] es una herramienta de sistema de control de versiones[10] esencial para cualquier proyecto de esta envergadura. Se ha utilizado un repositorio de **Github**[11] para el desarrollo de este, junto a la utilidad proporcionada por su tablero **Kanban**[12], con el objetivo de seguir una buena metodología **Agile**[8] .

React Native 

React Native[13] ha sido el framework[14] principal elegido para el desarrollo del proyecto. Se evaluaron las diferentes opciones disponibles, siendo entre las que más destacadas, **React Native** (perteneciente a Facebook) y **Flutter**[15] (perteneciente a Google). Aunque la decisión en sí no fue fácil, se valoraron tres factores:

- **La popularidad:** Flutter, al ser una tecnología nueva, está lejos de compararla con React Native, que le saca años de ventaja. Esto se ve reflejado, por ejemplo, en la cantidad de componentes web[9] que tienen disponibles.

- **El lenguaje:** React Native, al igual que **React**[16], se utiliza como lenguaje **Javascript**[10]. Al ser un lenguaje con el que previamente se ha trabajado en la carrera, existía una predilección notable hacia este, en vez de empezar a aprender un lenguaje desde cero, como es el caso de **Dart**[17], utilizado por Flutter.

- **Relación con React:** Debido a que este proyecto se desarrolló al mismo tiempo que las Prácticas Externas y en estas se utilizaba React como framework

[9] https://git-scm.com/
[10] Herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo.
[11] https://github.com/
[12]https://docs.github.com/es/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards
[13] https://reactnative.dev/
[14] Plataforma de software universal y reutilizable para desarrollar aplicaciones de software, productos y soluciones.
[15] https://flutter.dev/
[16] https://es.reactjs.org/
[17] https://dart.dev/

principal, haría que la curva de aprendizaje de dicho framework fuese muy grande en beneficio para ambos proyectos.

Programación funcional: React hooks

Dentro de React Native, se utilizará la programación funcional para desarrollar el proyecto en vez de utilizar programación orientada a objetos. Esto es debido a que, cada vez más, hay más artículos y documentación que recomiendan usar este paradigma de programación. Incluso el propio Facebook, ha creado recientemente los **React Hooks**[18], que adaptan este estilo de programación a la perfección.

Open Food Facts API

La principal forma de consulta de elementos escaneados es gracias a la **Open API** perteneciente a **Open Food Facts**. Dicha página está creada a base de conocimiento generado por los usuarios, por lo que, al no ser información demasiado controlada, puede llegar a tener entradas erróneas o incompletas. Aunque esta base de datos sea suficiente dentro del propio prototipo de aplicación, en el planteamiento del proyecto real se cuenta con la compra o creación de una base de datos con control sobre ella.

Expo

Expo[19] es un framework que trabaja específicamente con React Native, utilizándolo de base. Tiene tres características principales:

- **Ayuda en el desarrollo**: Gracias a su aplicación en la Play Store/App Store, **ExpoGo**[20], hace posible el desarrollo de un proyecto con hot reload[21] directamente en un dispositivo Android/iOS, sin tener que utilizar un simulador ni tener que generar un APK[22] en cada prueba.

[18] https://es.reactjs.org/docs/hooks-intro.html
[19] https://expo.io/
[20] https://expo.dev/client
[21] Evita tener que recargar la aplicación ni volver a lanzarla para ver los cambios realizados en el desarrollo.
[22] Fichero ejecutable de Android

- **Ayuda en el despliegue**: Con tan solo un comando, Expo genera un build[23] de la aplicación, pasa por una pipeline[24] el proyecto, y genera un APK, que lo mantiene en la nube en su propia página web durante 30 días.

- **Desarrollo multiplataforma**: Todo el código funciona tanto para Android, iOS y escritorio, por lo que, aunque haya que especificar condicionales para que funcionen correctamente ciertas características en cada uno de estos, agiliza enormemente el desarrollo multiplataforma.

React Native Paper 

Aunque muchos componentes utilizados son parte del propio React Native, se ha hecho uso de la ayuda de los componentes de **React Native Paper**[25], que tratan de componentes de **Material Design**[26] multiplataforma.

NPM 

Se ha hecho uso de **Node Package Manager**[27] para casi todas las tecnologías incluidas en el proyecto, junto a los componentes web utilizados.

ESLint 

Se ha configurado un linter, en este caso **ESLint**[28], para mantener un orden y estructura del código.

[23] Compilación del código de un proyecto.
[24] Cadena de procesos conectados de forma tal que la salida de cada elemento de la cadena es la entrada del próximo.
[25] https://callstack.github.io/react-native-paper/
[26] https://material.io/design
[27] https://www.npmjs.com/
[28] https://eslint.org/

Python 

Debido a que la base de datos descargada desde Open Food Facts tenía un tamaño gigantesco, se ha usado **Python**[29] para optimizarla y transformarla de CSV a JSON[30], que posteriormente se transformará en SQL.

Javascript 

Como se ha mencionado anteriormente, React Native trabaja con **Javascript**, lenguaje utilizado con anterioridad en la carrera. Más adelante, se planteó el pasar a **Typescript**[31], ya que las ventajas que tiene como lenguaje de tipado son bastante notables, pero el proyecto ya estaba demasiado avanzado, y refactorizar tal cantidad de código era una tarea que ralentizaría el proyecto de gran manera.

SQLite 

Dadas las pocas compatibilidades del sistema de bases de datos con Expo, **SQLite**[32] era la única opción viable para poder trabajar con una base de datos local, con el objetivo de su uso en la funcionalidad offline.

Microsoft Project 

Para la teorización de un supuesto del desarrollo de la aplicación comercial, se ha utilizado **Microsoft Project** para organizar, tanto el presupuesto, como organizar cada una de las fases del proyecto.

[29] https://www.python.org/
[30] Tipos de archivos contenedores de datos.
[31] Similar a Javascript, pero con tipado.
[32] https://www.sqlite.org/index.html

Capítulo 4

Metodología de desarrollo

La metodología de desarrollo del proyecto utilizada se basó en la metodología **Agile**. Cada dos semanas, los jueves, se llevaron a cabo reuniones, en las que el alumno mostraba los avances del proyecto y el tutor aportaba retroalimentación sobre los elementos desarrollados, indicaciones, recomendaciones, ideas y resolución de dudas acumuladas.

El canal de dichas reuniones fue Google Meet las cuales giraban en torno a un tablero Kanban. Dicho tablero se planteó realizarse en la plataforma **Trello**[33], pero finalmente se utilizó el del repositorio de **Github(4.1)**. Se dividió en varias columnas, que contenían cinco tipos de tarjetas de tareas:

- **To do**: Backlog de tareas acumuladas pendientes de su realización.
- **Blocked**: Tareas bloqueadas pendientes de factores externos, entre ellas, consultas con el tutor.
- **In progress**: Tareas actualmente en desarrollo.
- **Review**: Tareas finalizadas pendientes de revisión del tutor.
- **Done**: Tareas finalizadas por completo.

Cada tarjeta seguía un orden de prioridad según lo alto que esté en la lista de cada columna.

[33] https://trello.com/es

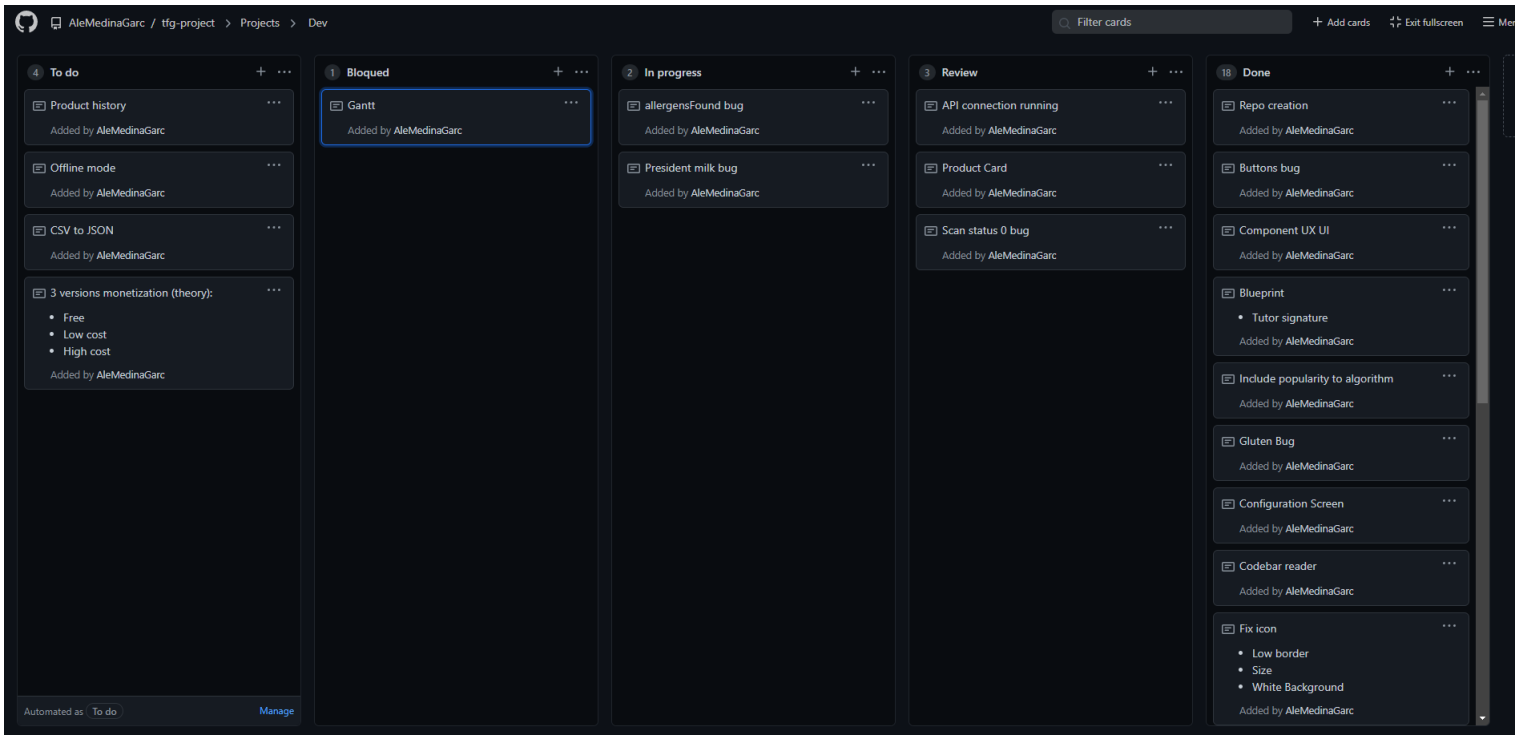

Figura 4.1: Tablero Kanban de Github

Capítulo 5

Desarrollo del prototipo de aplicación

5.1 Demostración Just In Mind

Antes de empezar el desarrollo del prototipo en React Native, se hizo un primer aproximamiento a la idea del diseño en **Just in Mind**[34], que se utilizó como referencia inicial para el desarrollo de la aplicación. Aunque fue una buena iniciativa para empezar a formar una idea de la interfaz del proyecto, el resultado final fue muy diferente a este primer planteamiento de la aplicación.

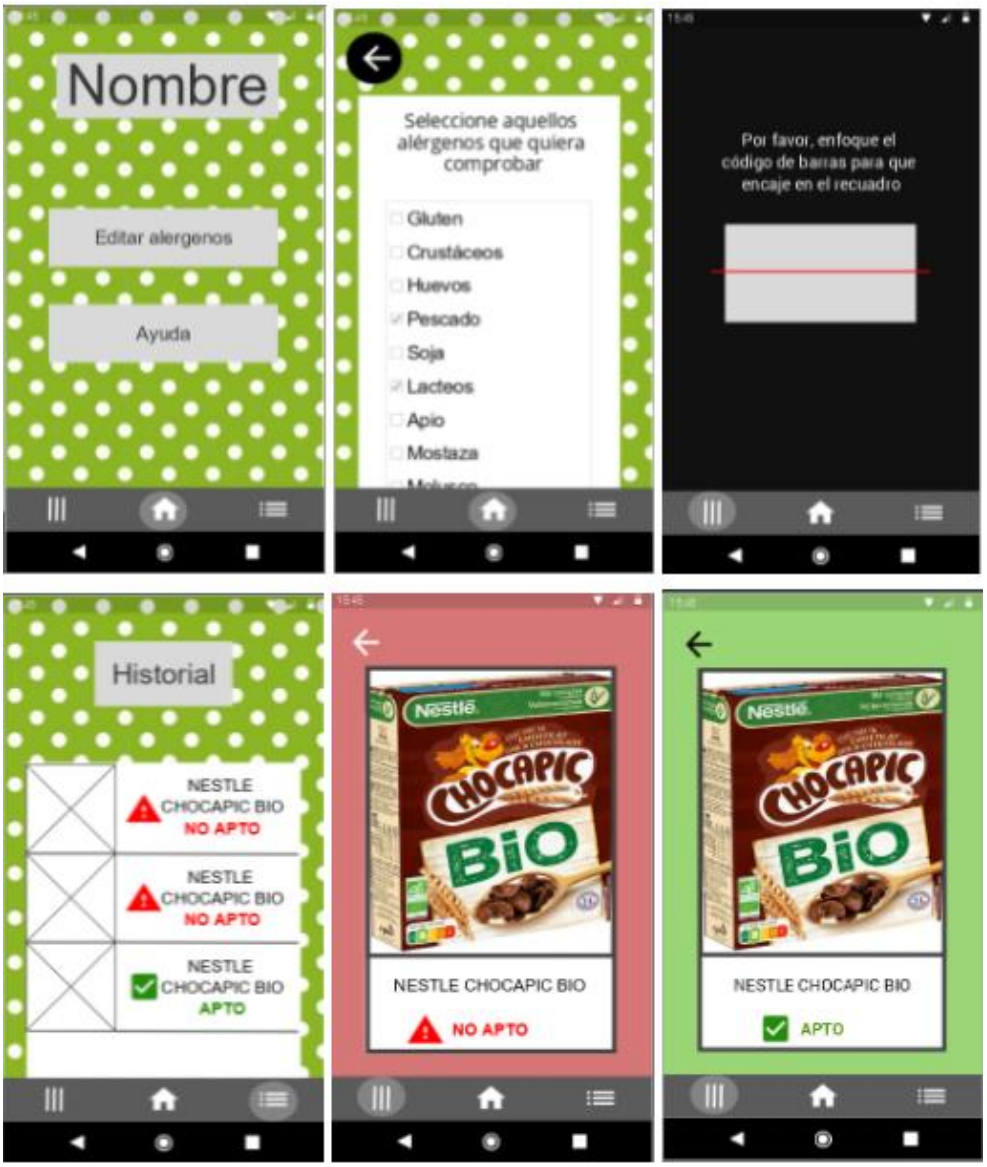

Figura 5.1: Prototipo de la aplicación

[34] https://www.justinmind.com/

5.2 Inicio de una aplicación vacía

Se utilizó como referencia a la hora de empezar la aplicación la documentación[35] publicada en la página oficial de Expo. A partir de ese momento, tanto npm como Expo se encargaron de la creación de ficheros y directorios iniciales. Más adelante, se amplió dicho sistema de directorios.

5.2.1 Navegación entre pantallas

Previamente al desarrollo del contenido de cada pantalla, se desarrolló el sistema de navegación entre las mismas. Para ello, se utilizó **React Native Navigator**[36], un componente instalado con NPM que permite tres tipos de navegación:

- **Stack navigation**: Navegación en profundidad en la aplicación. Consta de una "pila de pantallas" la cual se controla como una pila[37] genérica. Por ejemplo, para volver a la pantalla anterior, se realiza llamando a una función *pop*.

- **Tab navigation**: Navegación en anchura en la aplicación. Se utilizó para enlazar las tres pantallas principales de la aplicación, el inicio, el historial y el escáner.

- **Drawer navigation**: Navegación de menú lateral. Como no se requiere, ha sido el único sistema de navegación que no se ha implementado.

Debido a la complejidad de anidar diferentes sistemas de navegación, se utilizó el artículo "*React Navigation 5: Stack, Tab, and Drawer All in One"[11]* como herramienta de guía a la hora de implementar dicha funcionalidad.

[35]https://docs.expo.dev/get-started/installation/
[36] https://reactnavigation.org/
[37]https://es.wikipedia.org/wiki/Pila_(inform%C3%A1tica)

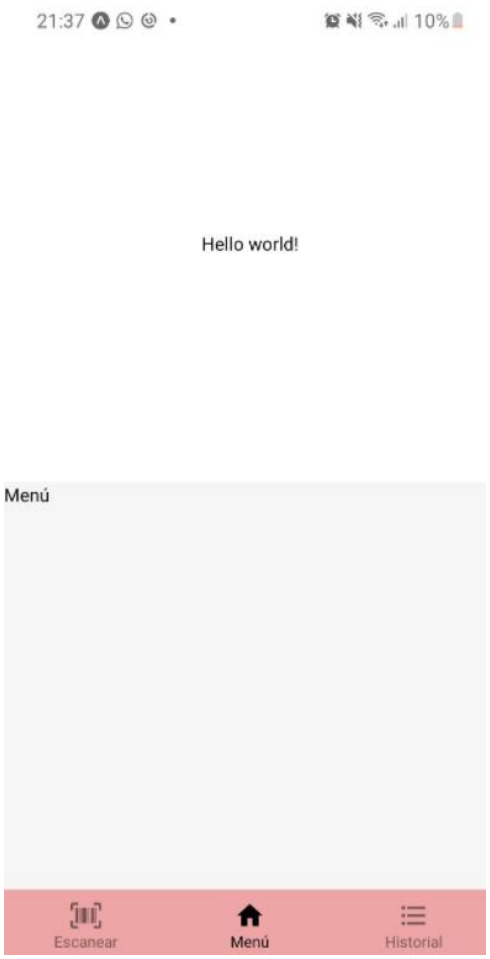

Figura 5.2: Versión 0.1.2 de la aplicación, usando tab navigation

5.2.2 Logo

El logo(**5.3**) se realizó de forma personalizada en la página de **Free Logo Design**[38]. Este se ha utilizado en el icono de la aplicación, en la imagen de carga de la aplicación, en el menú principal y en el apartado de ayuda.

Figura 5.3: Logo de la aplicación

[38] https://es.freelogodesign.org/

5.3 Página principal

La página principal(**5.4**) consta de dos botones, el primero para acceder a la configuración de los alérgenos perjudiciales para el usuario, y el segundo, una sección de ayuda, con información referente al proyecto.

Figura 5.4: Página principal final

5.4 Pantalla de configuración

El apartado de configuración(**5.5**) es donde el usuario selecciona, tocando una casilla, aquel alérgeno que quiera evitar consumir. Debido a que los alérgenos con los que trata la aplicación, y su información individual, no llega a ser una gran cantidad de datos, se descartó desde el comienzo utilizar un sistema de base de datos local para los mismos. La idea inicial consistía en que el programa editase un fichero JSON en el que el usuario cambiase los valores y así los datos guardados perduraran entre sesiones.

Sin embargo, tras un poco de investigación, se llegó a la conclusión de que usar un paquete llamado **Async Storage**[39], era una opción muchísimo más viable. No es tan completo como un sistema de base de datos, pero lo suficiente para cumplir el

[39] https://github.com/react-native-async-storage/async-storage

objetivo necesario. Async Storage permite, de forma asíncrona, editar datos internos de la aplicación que permanecen aun cerrándola, sin tener que hacer uso de un sistema de lectura y escritura de ficheros, como inicialmente se había planteado. Finalmente, se utilizó un sistema de casillas, en el que el usuario podría marcar aquel alérgeno no deseado. Debido a que como mucho, normalmente el usuario marcará no más de dos, se ha implementado que, al pulsar una casilla, automáticamente la aplicación lo devuelva a la pantalla de inicio, registrando el nuevo alérgeno.

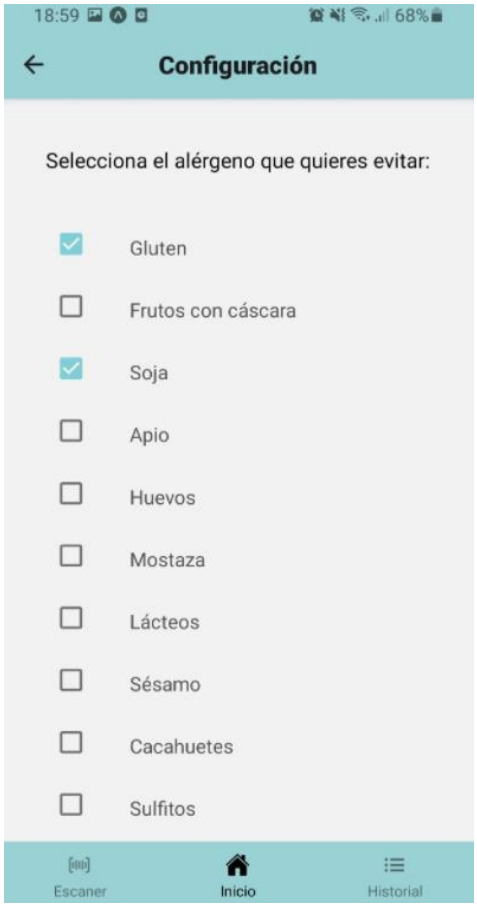

Figura 5.5: Página de configuración final

Async Storaje trabaja con strings, pero será transformado en JSON cuando haga falta, por ejemplo, a la hora de inicializarlo(**A.1**), que será el caso contrario. El fichero de configuración inicial(**A.2**) consta de un **array de objetos** con catorce entradas, cada una representando uno de los catorce alérgenos con los que se ha tenido en cuenta para el desarrollo del proyecto. Cada entrada contendrá un **nombre** de alérgeno, un **id** único, un **nombre codificado** que será como se encontrará en la API, y un **booleano** que indique si está seleccionado por el usuario o no.

5.5 Pantalla de ayuda

Este apartado tiene como propósito:

- **Informar** al usuario de la funcionalidad de la aplicación, además de instrucciones de uso.
- **Advertir** de la posibilidad de que el producto esté desactualizado o la información sea errónea debido a que se está tratando con una API pública y gratuita al tratarse de un prototipo de aplicación.
- Poner el **contexto** de propósitos y creación de la aplicación.
- **Mencionar** al autor junto a la institución, incluyendo un enlace a la página de la Universidad de La Laguna.

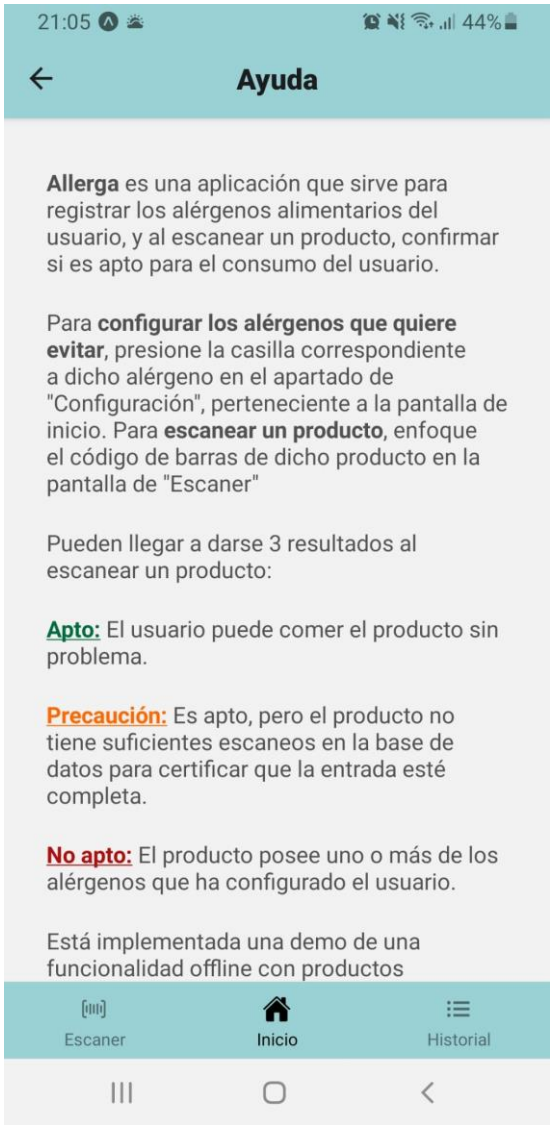

Figura 5.6: Pantalla de ayuda final

5.6 Pantalla de escaneo

Debido a la evolución de la hostelería y demás sectores para enfrentar las limitaciones ocasionadas por el impacto de la COVID-19, se ha generado un aumento de experiencia en toda la población en cuanto a la usabilidad de aplicaciones de lectura de códigos de barras y QR[40]. Este factor se ha tenido en cuenta a la hora de pensar la forma óptima, usable y accesible, a la hora de que el usuario introduzca un identificador a buscar en la base de datos.

Tras una búsqueda exhaustiva sobre la forma de implementar un escáner de códigos de barras en la aplicación, se ha decantado por utilizar el que proporciona Expo en su documentación[41], posteriormente modificado para que sea visualmente más accesible e intuitivo. La primera vez que se accede a este apartado en la aplicación, se le preguntará al usuario si quiere dar permisos de cámara. Al escanear cualquier código de barras, montará un componente "producto", el cual realizará una única llamada a la API.

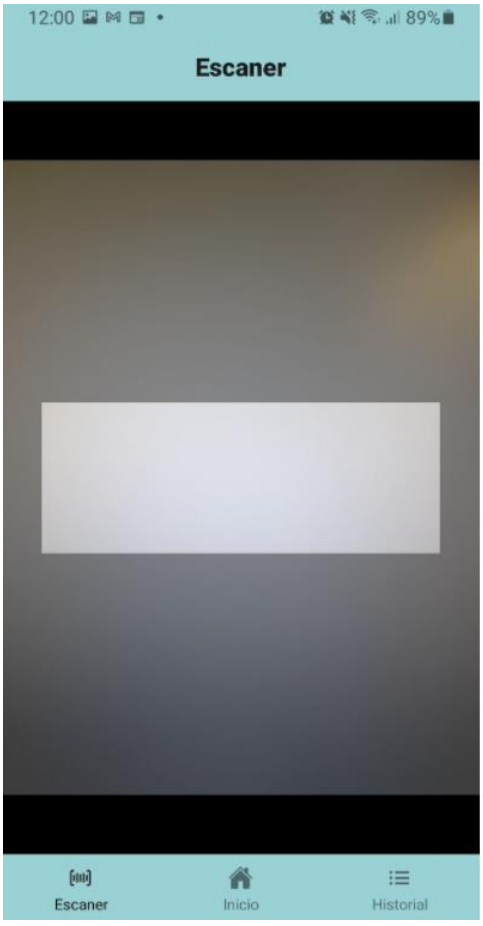

Figura 5.7: Pantalla de escáner final

[40] https://www.lavanguardia.com/comer/tendencias/20210207/6224861/codigos-qr-realidad-aumentada-evolucion-cartas-llego-covid-19.html
[41] https://docs.expo.dev/versions/latest/sdk/bar-code-scanner/

5.6.1 Conexión con la API

Como ya se ha mencionado anteriormente, la API utilizada es la proporcionada por Open Food Facts. La URL a consultar a la hora de escanear un producto ha sido la siguiente:

const url = `https://world.openfoodfacts.org/api/v0/product/${barCode}.json`;

Donde la variable *barCode* contiene el número entero referente al código de barras previamente analizado por el componente de escaneo, pasado al componente de renderizado del producto como *prop*[42].

5.6.2 Recursos API necesarios:

- **Foto**: *product.image_front_url*
- **Nombre**: *product.product_name_es, product.product_name, product.name_en, product.product_name_en*
- **Código de barras**: *product.code*
- **Popularidad**: *product.unique_scans_n*
- **Alérgenos**: *product.allergens_tags, product.traces_tags*

5.6.3 Popularidad

Ya que existen entradas incompletas en la base de datos, al tratarse de un recurso gratuito formado por cualquier usuario, se deberá tener en cuenta lo popular que es un producto; ya que esto influye a la hora de lo precisa y completa que sea la información. Existen dos atributos que se pueden utilizar para esta labor, los **escaneos totales** y los **escaneos individuales**, estos últimos no contando con que una persona escanee el mismo alimento múltiples veces. Siendo el escaneo individual el más significativo de nuestra definición de popularidad, se hará uso de él. Si el producto analizado no tiene alérgenos y tiene una popularidad baja, la aplicación mostrará un mensaje de advertencia indicando que, al no ser un alimento suficientemente popular, es muy probable que la información del producto no esté completa.

[42] https://es.reactjs.org/docs/components-and-props.html

Tras muchas pruebas para determinar que cifra de escaneos totales se puede tomar como valor mínimo, se ha acabado optado por **diez escaneos**, por lo que si el valor es menor que esta cifra, se advertirá al usuario de que la entrada podría estar incompleta.

5.7 Pantalla de producto

Al escanear un producto, posteriormente a la petición de la API, se montará el componente *ProductCard,* el cual usará un algoritmo(**A.3**) desarrollado para comprobar si se trata de un producto comestible o no.

ProductCard realiza un merge de los arrays de los **alérgenos** junto a las **trazas**, para después recorrer este conjunto comparando cada uno de los valores almacenados en el array de objetos de configuración. Cuando llega al caso en el que el nombre del conjunto de alérgenos coincide con el nombre codificado del objeto de la configuración, si está con la variable *check* activa, significa que ese alérgeno no está permitido; por lo que cambia el estado *edible* a falso y añade el nombre de este alérgeno en *allergensFound*.

Aparte de mostrar el nombre del producto, la foto y un botón para guardar el producto en el historial, dependiendo del resultado de dicho algoritmo, puede renderizarse diferentes pantallas(**5.8**):

- **Apto**: Muestra que el alimento se puede consumir sin problema, cambiando el fondo a un color verde, para que el usuario entienda de la forma más intuitiva posible que es seguro.

- **No apto**: Muestra que el alimento no se puede consumir, por las restricciones impuestas en la configuración de la aplicación. Al principio se planeó que mostrase un array de todos aquellos alérgenos detectados que no pueda consumir el usuario; pero debido a un bug desconocido, junto con una muy escasa especificación de errores de Expo, no se ha podido arreglar, mostrando únicamente el último detectado. A grandes rasgos, no causa problema para el uso del prototipo, ya que una persona que haya configurado la aplicación con varios alérgenos no podrá consumir el producto sea por uno u otro. También se ha añadido que el fondo cambie a un color rojo.

- **Apto con precaución**: Como se ha comentado anteriormente, si un producto tiene una popularidad baja, el resultado puede no ser preciso. Aparte de un fondo naranja para alertar al usuario, se ha implementado un pequeño texto explicando el por qué puede no ser cierta la información recopilada en la base de datos.

- **No encontrado**: Si el código de barras escaneado no se encuentra en la base de datos, el usuario puede estar consultando un alimento que no está registrado en esta, u otro elemento que directamente no es un alimento. En ambas situaciones, se le informa al usuario con un mensaje de error pertinente, junto a un botón de “Volver a escanear”.

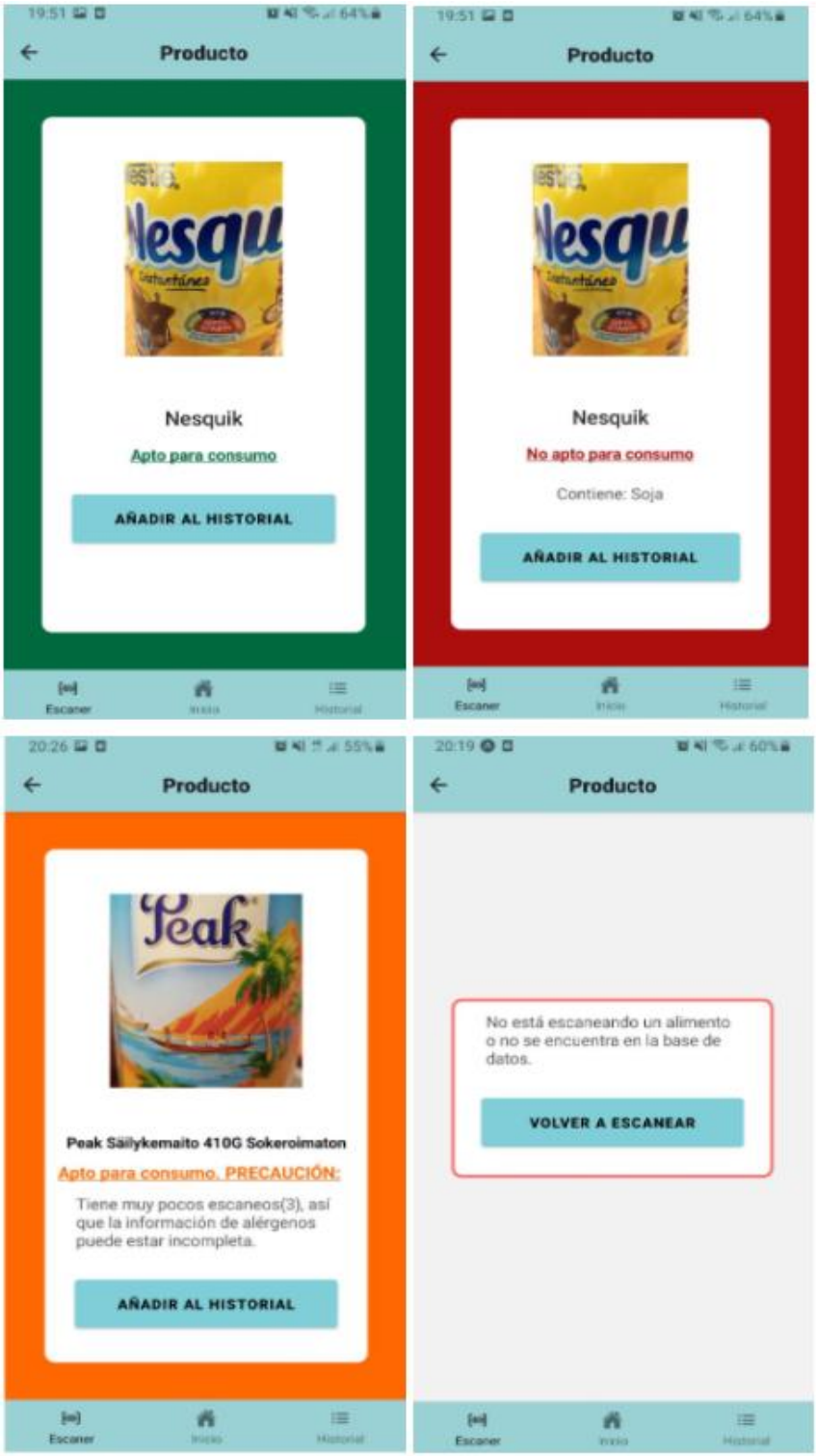

Figura 5.8: Todos los casos posibles al escanear un producto

5.8 Pantalla de historial

Se ha utilizado, de la misma forma que para la configuración, el almacenamiento de *Async Storage* para almacenar las entradas del historial. Se ha desarrollado un algoritmo de inserción de elementos en el historial(**A.3**).

En el caso de que sea un elemento válido (exista en la base de datos), primero se crea un item a introducir con las propiedades recogidas de la API y a continuación, recoge los datos de *Async Storage* de manera asíncrona, almacenándolos en una variable auxiliar cuando termine. Seguidamente, se recorre esta variable auxiliar buscando el elemento con su **clave única**: el **código de barras**. Si existe un elemento con el mismo código de barras, significa que ya se ha escaneado previamente y está almacenado en el historial. Se elimina dicho elemento del array auxiliar, con el objetivo de volverlo a añadir más adelante, ya que deberá de estar en la primera posición del historial. Además, los alérgenos pueden haber cambiado de un escaneo a otro y el resultado de si es apto o no, puede diferir del ya anteriormente almacenado.

A continuación, se limpia el *Async Storage* para luego introducir directamente el array auxiliar. Con el objetivo de no generar un historial infinito, se ha declarado una longitud máxima de **diez elementos** en el mismo, por lo que si en este paso, excede ese número, se llama a la función *pop* en el array. Por último, se añade el elemento escaneado en la primera posición del array auxiliar y se modifica la variable *setNewItem*, que es un hook *Context*[43] el cual está pendiente de si el componente de historial cambia para renderizar de nuevo.

[43] https://es.reactjs.org/docs/context.html

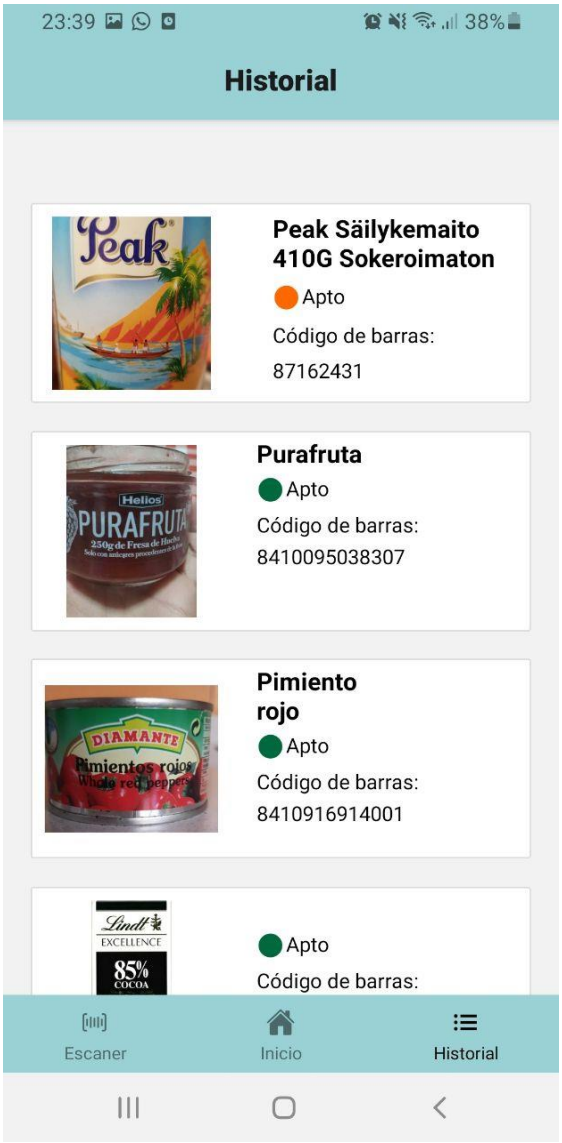

Figura 5.9: Pantalla de historial final

5.9 Funcionalidad offline

5.9.1 Optimización de la base de datos local

Open Food Facts tiene a su disposición su base de datos al completo disponible para su descarga en diferentes formatos, que inicialmente se descargó en formato CSV. Debido a que contiene una cantidad innecesaria de columnas que no se utilizarían en este proyecto (alrededor de 50), se realizó una conversión y optimización previa.

El primer acercamiento que hubo para abordar el problema de conversión de CSV a JSON, fue hacer un programa en C++ que analizara las **expresiones regulares** de cada uno de los campos requeridos y que los incluyese este JSON. Al principio se optó por esta opción de forma teórica, pero a la hora de la realización de la implementación fue muy diferente. Esta se hizo en **Python**(**A.4**), lenguaje con el que nunca se había trabajado previamente. Dicha elección fue debido a la facilidad y potencia de tratar archivos CSV tan grandes, llegando incluso a realizar la tarea requerida sin ningún problema, sin tener que trabajar con expresiones regulares previamente planteadas.

Se planteó que a la hora de guardar dichos valores, se **codificaran en valores numéricos** (0-13), para no tener que almacenar el string completo de cada uno de los alérgenos, por ejemplo:

[“en:gluten”, “en:milk”] == [0, 3]

No ahorraba la cantidad suficiente de bytes para compensar la poca claridad en el código que generaban, por lo que al final, se mantuvo esta forma anterior. Posteriormente, el archivo fue convertido a tipo SQL, para trabajar con SQLite en el proyecto. El CSV inicial pesaba la cantidad de **4 GB**, mientras que el fichero final acabó pesando **60 MB**.

Las columnas a tener en cuenta fueron la de **código de barras**, siendo esta la **clave primaria**, el **array de alérgenos** y el **array de trazas**. Se descartó incluir el nombre del producto para ahorrar más espacio, ya que solo se pretende que sirva con la funcionalidad más básica posible. También se quiso incluir la popularidad de cada uno de los productos, pero el fichero CSV no tenía ningún atributo que sirviese para la medición de esta.

```
code    url creator created_t   created_datetime    last_modified_t last_mod
0000000000017   http://world-en.openfoodfacts.org/product/0000000000017/vito
0000000000031   http://world-en.openfoodfacts.org/product/0000000000031/caca
0000000000003327986  http://world-en.openfoodfacts.org/product/0000000000332
0000000000100   http://world-en.openfoodfacts.org/product/0000000000100/mout
0000000001111111111    http://world-en.openfoodfacts.org/product/0000000000
0000000000123   http://world-en.openfoodfacts.org/product/0000000000123/sauc
0000000000178   http://world-en.openfoodfacts.org/product/0000000000178/mini
0000000000208   http://world-en.openfoodfacts.org/product/0000000000208/pist
0000000000284   http://world-en.openfoodfacts.org/product/0000000000284/pain
0000000000291   http://world-en.openfoodfacts.org/product/0000000000291/mend
000000000054    http://world-en.openfoodfacts.org/product/000000000054/limon
000000000075    http://world-en.openfoodfacts.org/product/000000000075/sache
000000000080    http://world-en.openfoodfacts.org/product/000000000080/pur-j
000000000088    http://world-en.openfoodfacts.org/product/000000000088/pate-
```

Figura 5.10: Extracto de CSV recién descargado de Open Food Facts

```
CREATE TABLE IF NOT EXISTS [ITEMS] (
[CODE] INT NULL,
[ALLERGENS] JSON NULL,
[TRACES] JSON NULL
);

INSERT INTO  VALUES
(0000000000017,[\"\"],[\"\"]),
(0000000000031,[\"\"],[\"\"]),
(0000000000003327986,[\"\"],[\"\"]),
(0000000000100,[\"en:mustard\"],[\"\"]),
(0000000001111111111,[\"\"],[\"\"]),
(0000000000123,[\"\"],[\"\"]),
(0000000000178,[\"\"],[\"\"]),
(0000000000208,[\"\"],[\"\"]),
(0000000000284,[\"\"],[\"\"]),
(0000000000291,[\"\"],[\"\"]),
(000000000054,[\"\"],[\"\"]),
(000000000075,[\"\"],[\"\"]),
(000000000080,[\"\"],[\"\"]),
(000000000088,[\"\"],[\"\"]),
(0000000000949,[\"\"],[\"\"]),
(0000000000970,[\"\"],[\"\"]),
(0000000001001,[\"\"],[\"\"]),
(0000000001007,[\"\"],[\"\"]),
(000000000112,[\"\"],[\"\"]),
(0000000001137,[\"\"],[\"\"]),
(000000000114,[\"\"],[\"\"]),
(0000000001151,[\"\"],[\"\"]),
(0000000001199,[\"en:eggs\"\,\"en:mustard\"],[\"\"]),
(0000000001281,[\"\"],[\"\"]),
```

Figura 5.11: Extracto del fichero final optimizado

5.9.2 Selección de sistema de base de datos

Primero se planeó usar sistemas como MongoDB[44], WatermelonDB[45]... pero Expo no era compatible con casi ninguno de estos. La única opción existente era SQLite, la cual tenía un hueco en su propia documentación. Al no disponer de más opciones, se optó por esta.

5.9.3 Desarrollo de la base de datos

Inicialmente se hizo uso de **NetInfo**[46], un componente dedicado para la comprobación del estado de la red. Se configuró para que, cuando la aplicación detecte que no hay señal de Internet, o esta sea demasiado leve, utilice la base de datos interna, en vez de conectarse con la API(**A.5**).

Seguidamente se empezó trabajando con SQLite pero, tras un estancamiento crítico que perduró durante semanas, se optó por realizar una versión mucho más básica. Dicho estancamiento fue generado por la imposibilidad de usar alternativas en Expo y la falta de documentación que proporcionaba este framework. Aun así, aunque no se llegó a implementar al completo con SQLite, se llegó a la conclusión de que, aunque sea una versión mínima de demostración, era una característica importante para el proyecto. El resultado final fue un simple fichero JSON, el cual contiene la misma estructura de información que el fichero generado para la base de datos, pero con un número muy reducido de elementos.

[44] https://www.mongodb.com/es
[45] https://github.com/Nozbe/WatermelonDB
[46] https://github.com/react-native-netinfo/react-native-netinfo

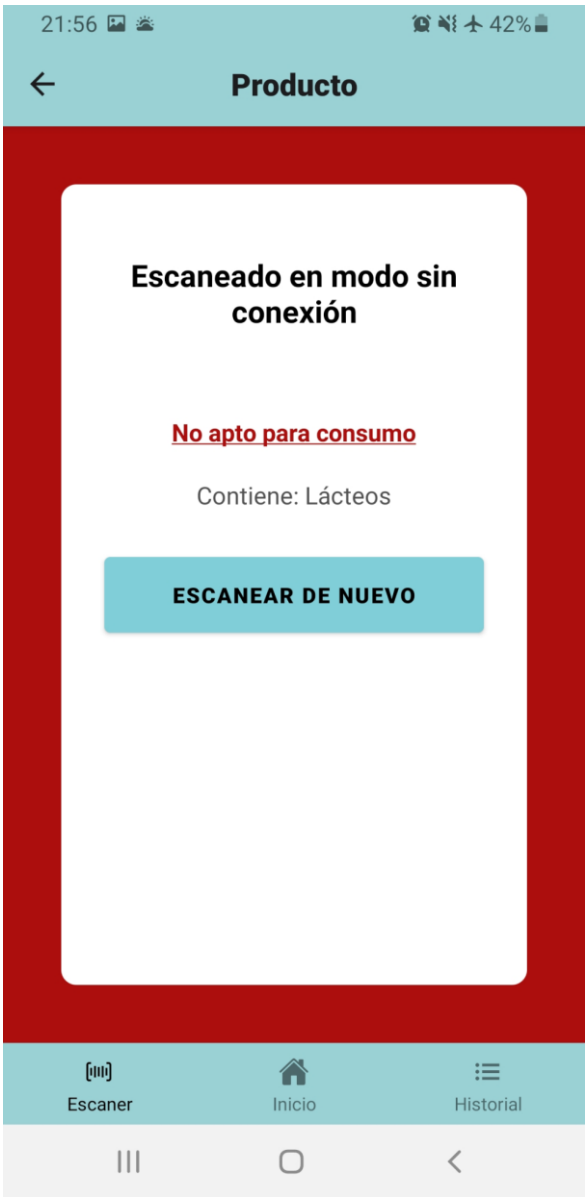

Figura 5.12: Demostración de un escaneo con modo avión activado

5.10 Versión de escritorio

Aun siendo un añadido bastante secundario en el planteamiento inicial del prototipo del proyecto, se ha implementado una versión demo del entorno web(**5.13**) de escritorio de Allerga. Debido a que el desarrollo del proyecto ha sido realizado con Expo, se ha podido implementar de forma **paralela**, aprovechando el máximo potencial de esta tecnología.

El único cambio que se ha tenido que especificar de una versión a otra, ha sido sustituir la pantalla de escáner, por un campo de texto, ya que se ha tenido en cuenta que gran parte de la población no posee cámara en el ordenador de sobremesa. Se ha ajustado esta funcionalidad para que sea simplemente un buscador de alimentos según el código de barra introducido en un campo de texto.

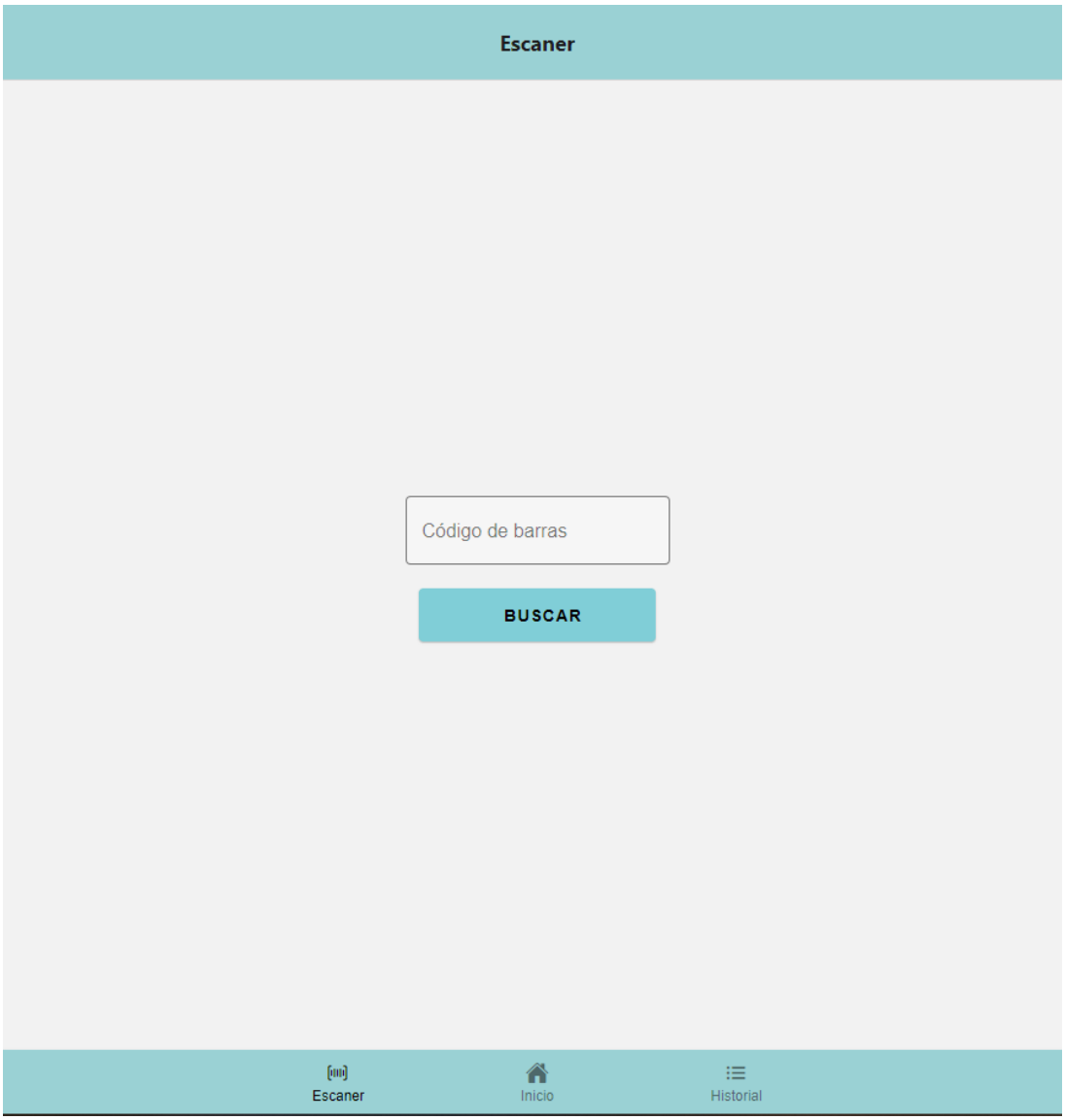

Figura 5.13: Vista previa de demostración web

5.11 Problemática encontrada

5.11.1 Problemas con Expo

Lo que más ha ralentizado el desarrollo del prototipo del proyecto ha sido Expo. Ha sido una herramienta muy útil a la hora del desarrollo del software, teniendo una de

las características más destacables el haber podido desarrollar en paralelo una versión de escritorio. No obstante se han encontrado ciertas carencias en el mismo:

- **Errores muy poco precisos:** Debido a esto, una característica, guardar un array de alérgenos detectados en vez del último, no ha sido posible ya que nunca se llegó a concretar suficientemente el error.

- **Pocas resoluciones de errores en foros:** El ejemplo más claro de ello es la información buscada sobre SQLite, en concreto, de base de datos *pre populated*.

- **Bugs críticos:** Existió un bug que consiste en que en el entorno de desarrollo de Expo, la implementación de una funcionalidad funcionaba correctamente, pero a la hora de probarlo en el APK, no. Este error perduró varios días completos de desarrollo.

- **Poca compatibilidad con otras tecnologías:** Como ya se ha comentado anteriormente, la mayor restricción de ello son las bases de datos locales.

- **Builds lentas debido a la versión gratis:** No se ha podido solucionar un error que hacía que la cámara del móvil no se iniciara nada más aceptar el uso de esta por la aplicación. Como este aspecto no se puede probar en el entorno de desarrollo y teniendo un tiempo de espera de 30 minutos entre cada generación de APK, por el uso de una versión gratuita de Expo, no era rentable para el desarrollo del proyecto.

- **Versión de escritorio en beta**: Expo tiene la implementación en escritorio en *beta* y se encontraron ciertas cosas fuera de lo común, como por ejemplo, deshabilitar el campo de texto y el botón de buscar del apartado de búsqueda sin ningún motivo aparente y sin dar ninguna retroalimentación acerca de ello. Por ello, solo se ha hecho una pequeña demostración de lo fácil que sería crear una versión de escritorio a partir de la aplicación ya generada con Expo y React Native, en vez de una versión completamente funcional.

Con estos problemas, a la hora de iniciar el proyecto oficial, se tendría en consideración el dejar de usar esta tecnología, y hacer una aplicación enteramente con React Native base.

Capítulo 6

Estudio de la viabilidad económica

Como bien se ha analizado en el capítulo 2, se han tenido como referencia dos aplicativos que cubren la misma funcionalidad con la que trabaja Allerga, sin embargo, ninguna alternativa de estas coincide exactamente con el papel que cumple Allerga y mucho menos, se han promocionado lo más mínimo con el objetivo de explotar este sector. Por ello, se ha buscado centrarse en el servicio que ofrece Allerga, como un servicio de aplicación **sencilla**, **configurable** y **accesible** en gran medida, que cumpla un **propósito específico** sin saturar a su público. Para conocer realmente la viabilidad del proyecto, debemos dar respuesta a temas como, cuánto cuesta desarrollar el proyecto, cuánto tiempo sería necesario para llevarlo a cabo, cuándo se recuperará la inversión inicial o cómo se pretende comercializar. Para ello, se ha simulado un supuesto de gestión de proyecto informático para este caso. En él, se plantea la realización completa de esta aplicación con una empresa, comprando todas las licencias pertinentes, utilizando una base de datos personalizada e implementando un sistema de monetización.

6.1 Desarrollo del proyecto

Se ha desarrollado, gracias a **Microsoft Project**, una planificación monetaria y cronológica del desarrollo del proyecto, generando además, una división entre las distintas fases de este. Este proyecto se dividirá en cuatro etapas diferentes.

Durante la primera **Fase de Análisis**, se pretende comprender cuál es el problema que queremos resolver, junto a los diferentes públicos objetivos que se quieren tener en cuenta a la hora de desarrollar la aplicación final. Para ello se constituye un conjunto de requisitos que debe cumplir el producto, indicando las funcionalidades que debe tener y las necesidades que debe satisfacer.

Seguidamente, después de la primera fase, entrará en desarrollo la **Fase de Diseño**, en la que se establecerá un diseño de arquitecturas que compondrán el producto, realizando tanto un análisis de hardware como de las tecnologías a utilizar. Además, en esta fase se llegará a tener en cuenta el diseño de interfaz de usuario final, junto al de casos de uso, llegando a cumplir en su totalidad con los diseños **UI** y **UX** [47].

[47] https://www.neoland.es/blog/que-es-el-ux-ui-design

Nombre de tarea	Duració	Comienzo	Fin	Pre	Nombres de los recursos	Costo
Allerga	**123 días?**	**mié 23/02/22**	**vie 12/08/22**			**56.455,20 €**
Fase de análisis	**24 días**	**mié 23/02/22**	**lun 28/03/22**			**4.147,20 €**
Estudio de mercado	5 días	mié 23/02/22	mar 01/03/22		Analista de sistema	1.400,00 €
Estudio de competencia	3 días	mié 02/03/22	vie 04/03/22	3	Analista de sistema;Diseñador 1[20%]	907,20 €
Análisis de la problemática	4 días	mié 09/03/22	lun 14/03/22	3;4	Analista de sistema	1.120,00 €
Arquitectura del proyecto	10 días	mar 15/03/22	lun 28/03/22	5	Jefe de proyecto[20%]	720,00 €
Fase de diseño	**23 días**	**jue 24/03/22**	**sáb 23/04/22**	2		**9.448,00 €**
Compra/creación de base de datos	7 días	jue 24/03/22	vie 01/04/22		Analista de sistema	1.960,00 €
Elección de tecnologías	8 días	mié 06/04/22	vie 15/04/22	8	Jefe de proyecto[50%];Analista de sistema[50%]	2.560,00 €
Diagrama de casos de uso	7 días	jue 24/03/22	vie 01/04/22		Diseñador 1	784,00 €
Diagramas de clase	7 días	jue 24/03/22	vie 01/04/22		Diseñador 2	784,00 €
Diseño gráfico	15 días	lun 04/04/22	vie 22/04/22	10;11	Diseñador 1;Diseñador 2	3.360,00 €
Fase de desarrollo	**67 días**	**mar 26/04/22**	**mié 27/07/22**	**7**		**39.100,00 €**
Frontend	54 días	mar 26/04/22	lun 11/07/22		Programador 1;Programador 2;Jefe de proyecto[10%]	19.580,00 €
Backend	54 días	mar 26/04/22	lun 11/07/22		Programador 3;Programador 4	17.600,00 €
Portabilidad a iOS y escritorio	12 días	mar 12/07/22	mié 27/07/22	14;15	Programador 1[50%];Programador 2[50%]	1.920,00 €
Fase de despliegue	**11 días**	**jue 28/07/22**	**jue 11/08/22**	**13**		**3.760,00 €**
Contratar servidores	7 días	jue 28/07/22	vie 05/08/22		Administrador de plataforma	1.120,00 €
Campaña de marketing	11 días	jue 28/07/22	jue 11/08/22		Marketing subcontratado 1	1.320,00 €
Especialistas en venta	11 días	jue 28/07/22	jue 11/08/22		Marketing subcontratado 2	1.320,00 €

Tabla 6.1: Tabla de las fases del proyecto

A continuación, llegará el momento de empezar la **Fase de Desarrollo**, en la que se trabajará con todo lo recopilado de las dos fases anteriores. Tanto el desarrollo del *Frontend*[48] como el del *Backend*[49] se realizarán en paralelo, dividiéndose en equipos de dos personas para cada labor. Un jefe de proyecto será el que, mediante la metodología Agile y sus reuniones diarias pertinentes, coordine ambos equipos y lleve el rumbo del proyecto.

Finalmente, una vez concluida la fase de desarrollo, llegará la Fase de Despliegue, la cual se centrará en los últimos detalles de cara al lanzamiento del producto.

[48] Parte de una aplicación que interactúa con los usuarios, es conocida como el lado del cliente.
[49] Interior de las aplicaciones que viven en el servidor, es conocida como el lado del servidor.

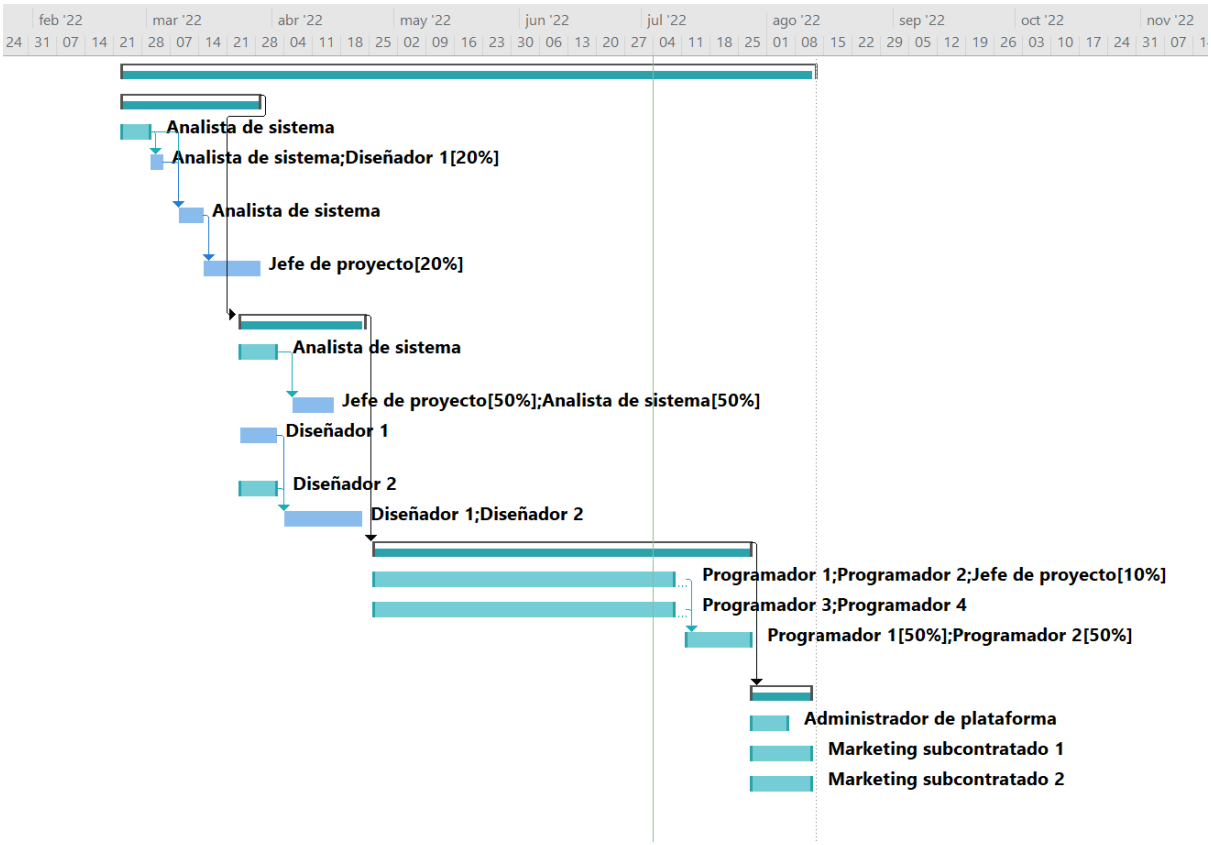

Figura 6.1: Diagrama de Gantt del proyecto

Para llevar a cabo las diferentes tareas que componen cada fase del desarrollo, se pretende contratar a un equipo conformado por:

- **Un analista de sistemas:** Trabajará en la fase de análisis y diseño, acotando el problema y el enfoque de la aplicación, definiendo además las tecnologías a utilizar tras un estudio. El salario será de 35 euros la hora.

- **Dos diseñadores gráficos:** Se encargarán del diseño de la aplicación, junto con que se cumplan los estándares de UI y UX. El salario será de 14 euros la hora por cada uno.

- **Un jefe de proyecto:** Tendrá un seguimiento en todo momento desde la primera fase del proyecto hasta el fin del desarrollo del aplicativo. El salario será de 45 euros la hora.

- **Dos programadores Frontend:** Desarrollarán la parte Frontend de la aplicación. El salario será de 14 euros la hora por cada uno.

- **Dos programadores Backend:** Desarrollarán la parte Backend y todo lo referente a base de datos de la aplicación. El salario será de 14 euros la hora por cada uno.

- **Un administrador de plataforma:** Gestionará todo lo referente a los dominios y a servidores en la nube. El salario será de 10 euros la hora.

- **Equipo de marketing subcontratado:** Se encargarán de todo lo referente a la promoción de la aplicación. El salario será de 10 euros la hora por cada uno.

La duración esperada del proyecto es de **123 días laborables**, lo equivalente a **5 meses** de desarrollo empezando el *23 de marzo* y terminando el *11 de agosto*. El presupuesto final es de **55.975,20 euros**, sin contar con las instalaciones ni el equipo.

6.2 Modelo de Monetización

Existen diferentes ideas sobre como monetizar el producto:

- **Versión gratuita**: El usuario que utilice la aplicación de forma gratuita será con las mismas funcionalidades que la fase del proyecto actual, pero sumando incluir anuncios en cada escaneo. No será publicidad que colapse al usuario ni que impida el funcionamiento con videos molestos, sino un pequeño banner de anuncio en la parte inferior de la pantalla. No solo se pretende incluir anuncios genéricos, sino también se propone contactar con ciertas marcas para promocionar un producto en particular según los alérgenos que tenga restringidos el usuario.

- **Versión premium**: Se basará en quitar anuncios y recomendar alternativas ante un producto no apto escaneado, sin publicidad de por medio pero siguiendo con los acuerdos con marcas, teniendo en cuenta el formato de evaluación alimentario de Nova[50] y promocionando un beneficio de salud para el usuario.

- **Versión premium plus**: Lo mismo que la versión premium pero añadiendo la posibilidad de hacer una lista de la compra, que previamente analice los productos. Además, se incluye una opción adicional que se basa en generar una dieta según un objetivo especificado por el usuario (ganancia muscular, pérdida de peso, mantener físico, ...) con productos que el usuario puede consumir debido a su condición y restricciones de alérgenos.

[50] https://es.openfoodfacts.org/nova

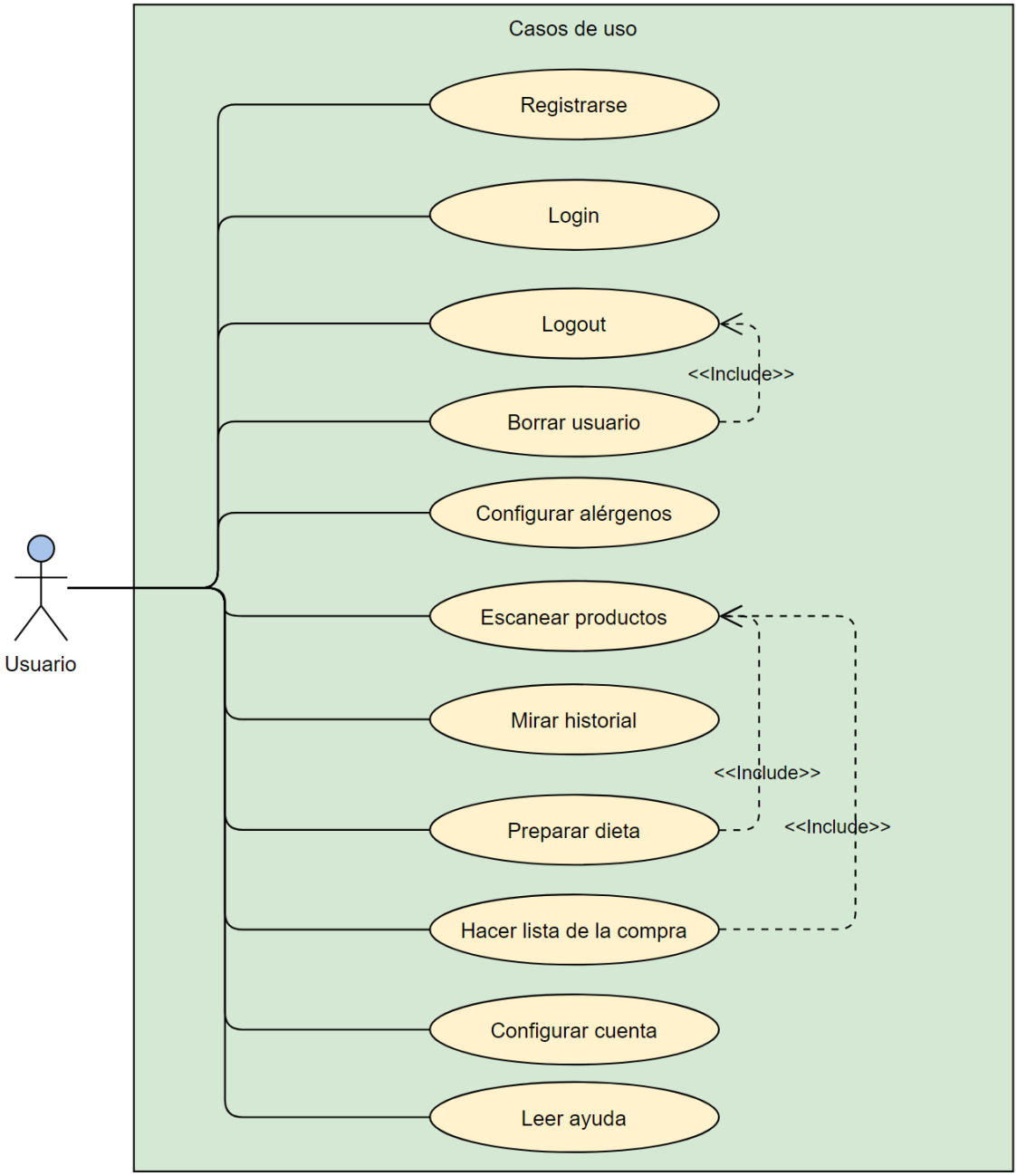

Figura 6.2: Diagrama de casos de uso

6.3 Retorno de inversión

Estamos trabajando en un mercado español prácticamente sin explotar. Más adelante, después de la salida del producto, la opción de traducirla a otras lenguas puede llegar a ser muy conveniente, debido a que la aplicación trata de solventar un problema completamente globalizado. Siendo una aplicación que podría llegar a estar en móviles de todo el mundo y casi esencial en el día a día de un gran porcentaje de la

población, se propone como objetivo llegar a que la usen cerca de **150.000** usuarios dentro de los 2 primeros años. Hay que tener en cuenta de que no solo está dirigido a un público que tiene alergias alimentarias (que ya es el 7% de la población, cifra en aumento con cada año), sino también a sus círculos cercanos de amigos y familiares.

	Usuarios
octubre	1030
noviembre	2400
diciembre	3800
enero	4500
febrero	5600
marzo	7000
abril	9400
mayo	12699
junio	14700
julio	18455
agosto	25000
septiembre	30500
octubre	37745
noviembre	43745
diciembre	50400
enero	59745
febrero	63653
marzo	70900
abril	86475
mayo	93647
junio	98600
julio	107465
agosto	110600
agosto	120600
septiembre	130600

Tabla 6.2: Estimación de usuarios registrados a lo largo de dos años

Figura 6.3: Gráfico de usuarios estimados por mes

En cuanto a la monetización, se plantea el siguiente caso:

- Se presupone que el 87% de los usuarios usen la versión gratuita: El ingreso por cada mil impresiones (RPM)[51] ronda los 3 euros y se tiene en cuenta de que una persona analice 50 productos en un mes de media.

- Se presupone que el 10% de los usuarios usen la primera versión de pago: Cada versión de pago generará 3 euros por cada usuario nuevo.

- Se presupone que el 3% de los usuarios usen la versión más alta: Cada mensualidad de cada usuario supondrá un beneficio de 12 euros al mes.

[51] https://support.google.com/adsense/answer/190515?hl=es-419

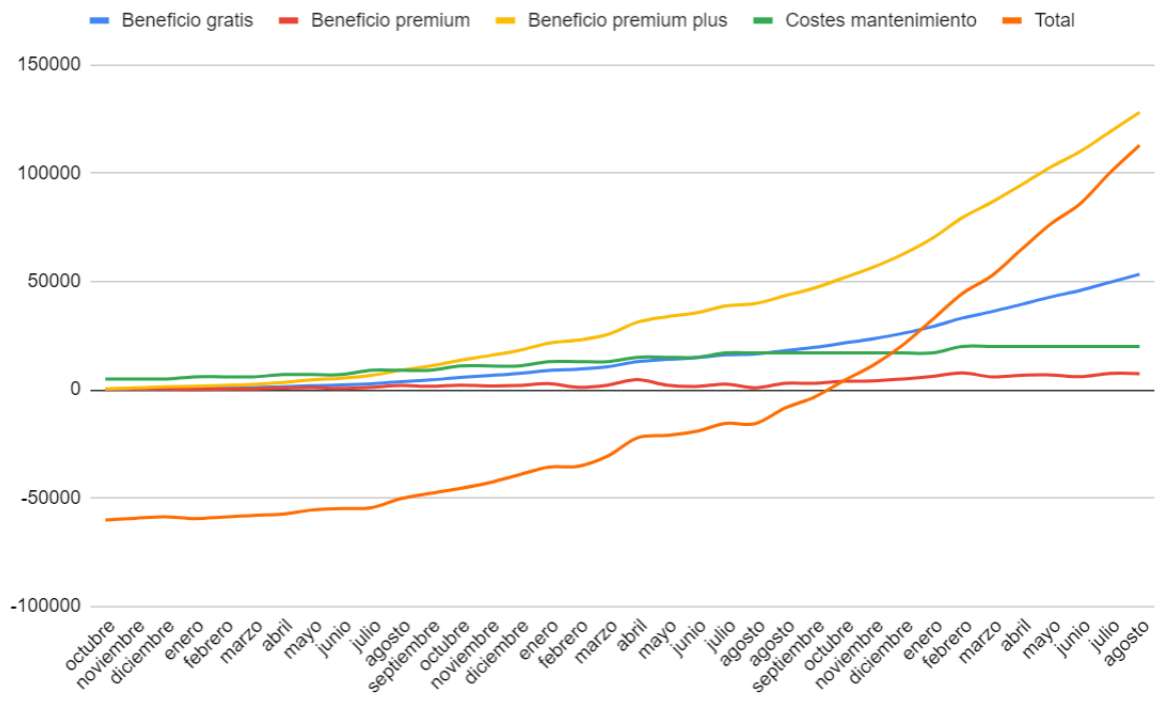

Figura 6.4: Gráfico de beneficios

Se puede observar(**6.3**) que a partir del **segundo año** en el que se ha lanzado la aplicación, ya será lo suficientemente popular como para generar beneficios para la empresa. Además, debido a la variabilidad que supone, no se ha tenido en cuenta ser contratado por alguna empresa alimentaria para promocionar sus productos en forma de anuncio, lo que subiría aún más las ganancias.

Se ha propuesto un coste en **gastos incrementales** en infraestructura de la empresa en general, para uso de licencias, alojamiento, actualizaciones y parches, mantenimiento y otros gastos generados por la aplicación. En el gráfico se observa como el mayor beneficio viene dado por la versión premium plus, aun teniendo en cuenta que solo la utilizarán el 3% de los usuarios.

	Ingreso gratis	Ingreso premium	Ingreso premium plus	Costes mantenimiento	Total
octubre	154,5	309	370,8	5000	**-60.140,90€**
noviembre	360	411	864	5000	**-59.340,20€**
diciembre	570	420	1368	5000	**-58.617,20€**
enero	675	210	1620	6000	**-59.470,20€**
febrero	840	330	2016	6000	**-58.789,20€**
marzo	1050	420	2520	6000	**-57.985,20€**
abril	1410	720	3384	7000	**-57.461,20€**
mayo	1904,85	989,7	4571,64	7000	**-55.509,01€**
junio	2205	600,3	5292	7000	**-54.877,90€**
julio	2768,25	1126,5	6643,8	9000	**-54.436,65€**
agosto	3750	1963,5	9000	9000	**-50.261,70€**
septiembre	4575	1650	10980	9000	**-47.770,20€**
octubre	5661,75	2173,5	13588,2	11000	**-45.551,75€**
noviembre	6561,75	1800	15748,2	11000	**-42.865,25€**
diciembre	7560	1996,5	18144	11000	**-39.274,70€**
enero	8961,75	2803,5	21508,2	13000	**-35.701,75€**
febrero	9547,95	1172,4	22915,08	13000	**-35.339,77€**
marzo	10635	2174,1	25524	13000	**-30.642,10€**
abril	12971,25	4672,5	31131	15000	**-22.200,45€**
mayo	14047,05	2151,6	33712,92	15000	**-21.063,63€**
junio	14790	1485,9	35496	15000	**-19.203,30€**
julio	16119,75	2659,5	38687,4	17000	**-15.508,55€**
agosto	16590	940,5	39816	17000	**-15.628,70€**
agosto	18090	3000	43416	17000	**-8.469,20€**
septiembre	19590	3000	47016	17000	**-3.369,20€**
octubre	21563,7	3947,4	51752,88	17000	**4.288,78€**
noviembre	23618,4	4109,4	56684,16	17000	**11.436,76€**
diciembre	26108,7	4980,6	62660,88	17000	**20.774,98€**
enero	29201,25	6185,1	70083	17000	**32.494,15€**
febrero	33111,9	7821,3	79468,56	20000	**44.426,56€**
marzo	36086,7	5949,6	86608,08	20000	**52.669,18€**
abril	39427,05	6680,7	94624,92	20000	**64.757,47€**
mayo	42860,4	6866,7	102864,96	20000	**76.616,86€**
junio	45875,55	6030,3	110101,32	20000	**86.031,97€**
julio	49627,05	7503	119104,92	20000	**100.259,77€**
agosto	53361,75	7469,4	128068,2	20000	**112.924,15€**

Tabla 6.3: Tabla de beneficios

Capítulo 7

Conclusiones, resultados y líneas futuras

El proyecto, en aspectos generales, ha cumplido con casi todas las actividades propuestas a la perfección, a excepción de las ya mencionadas anteriormente. Se han obtenido las competencias pertinentes al desarrollo de un proyecto a gran escala sin contar con un equipo de trabajo, solo bajo la supervisión y recomendación del tutor.

El resultado ha sido una aplicación optimizada y veloz, cuya funcionalidad es informar al usuario, tras una previa configuración, si al escanear un producto, es apto para su compra. Se han tenido en cuenta los catorce alérgenos más populares, junto a una interfaz simple, accesible, clara y bonita.

En cuanto a las tecnologías utilizadas, hay altibajos a la hora de echar la vista atrás. Expo ha sido un descubrimiento bastante interesante, pero por los problemas que conlleva, si empezase otro proyecto similar basado en React Native, no creo que lo volviese a utilizar. Si al menos hubiese desarrollado también una versión para iOS, a lo mejor le hubiese sacado más partido a sus funcionalidades, siendo esta la única razón por lo que me lo plantearía, pero no ha sido el caso para este proyecto.

El desarrollo de esta aplicación se planea dejar tal y como está actualmente. El objetivo principal de este proyecto era poner a prueba todos mis conocimientos aprendidos a lo largo de mi paso por la universidad, emprender un reto de aprender y desarrollar mi primera aplicación móvil y hacer que uno de mis clientes más importantes, mi madre, por fin dejase de llevar gafas al supermercado. Todo esto ha sido cumplido.

Capítulo 8

Summary and Conclusions

The project, in general aspects, has fulfilled almost all the proposed activities by perfection, except for those already mentioned above. Competencies relevant to the development of a large-scale project have been obtained without having a work team, only under the supervision and recommendation of the tutor.

The result has been an optimized and fast application, whose functionality is to inform the user, after a previous configuration, if when scanning a product, it can be consumed by it. The 14 most popular allergens have been considered, along with a simple, accessible, clear and beautiful interface.

In terms of the technologies used, there are ups and downs when looking back. Expo has been quite an interesting discovery, but due to the problems involved, if I started another similar project based on React Native, I don't think I would use it again. If at least I had also developed a version for iOS, maybe I would have gotten more out of its functionalities, this being the only reason why I would consider it, but it has not been the case for this project.

The development of this application is planned to be left as it currently is. The main objective of this project was to test all my knowledge learned throughout my time at university, to undertake a challenge of learning and developing my first mobile application, and to make one of my most important clients, my mother, finally stop wearing glasses in the supermarket. All of this has been accomplished.

Capítulo 9

Presupuesto: Trabajo Fin de Grado

A parte del cálculo de la inversión necesaria para llevar a cabo el desarrollo de Allerga, se ha realizado un presupuesto sobre el trabajo realizado en el Trabajo de Fin de Grado.

Tareas	Horas	Precio
Desarrollo de un plan de negocio	20 horas	15€/hora
Estudio previo del tema y búsqueda de bases de datos	20 horas	15€/hora
Estudio previo de tecnologías	20 horas	15€/hora
Desarrollo del diseño UX y UI	20 horas	15€/hora
Desarrollo del prototipo	200 horas	15€/hora
Documentación	60 horas	15€/hora
Total	**340 horas**	**5.100€**

Tabla 9.1: Presupuesto del proyecto

Este proyecto se ha realizado en 340 horas de trabajo, las cuales suponen un coste de 5.100€.

Apéndice

A.1. DataStream.js

```
import AsyncStorage from '@react-native-async-storage/async-storage';

// Para obtener los datos de Async Storage en formato JSON
const getJsonConfig = async (key) => {
  try {
    const jsonValue = await AsyncStorage.getItem(key);
    return jsonValue != null ? JSON.parse(jsonValue) : null;
  } catch (e) {
    alert(e);
  }
};

// Para escribir los datos de Async Storage en string
const setJsonConfig = async (value, key) => {
  try {
    const jsonValue = JSON.stringify(value);
    await AsyncStorage.setItem(key, jsonValue);
  } catch (e) {
    alert(e);
  }
};

// Para borrar un item de Async Storage
const removeJsonConfig = async (key) => {
  try {
    await AsyncStorage.removeItem(key);
  } catch (e) {
    alert(e);
  }
};

export {
  removeJsonConfig, setJsonConfig, getJsonConfig,
};
```

A.2. config.json


```
[
    {
      "name": "Gluten",
      "id": "0",
      "encoded": "en:gluten",
      "check":false
    },
    {
      "name": "Frutos con cáscara",
      "id": "1",
      "encoded": "en:nuts",
      "check":false
    },
    {
      "name": "Soja",
      "id": "2",
      "encoded": "en:soybeans",
      "check":false
    },
    {
      "name": "Apio",
      "id": "3",
      "encoded": "en:celery",
      "check":false
    },
    {
      "name": "Huevos",
      "id": "4",
      "encoded": "en:eggs",
      "check":false },
    {

      "name": "Mostaza",
      "id": "5",
      "encoded": "en:mustard",
      "check":false },
    {
      "name": "Lácteos",
      "id": "6",
      "encoded": "en:milk",
      "check":false
    },
    {
      "name": "Sésamo",
      "id": "7",
      "encoded": "en:sesame",
```

```
      "check":false
    },
    {
      "name": "Cacahuetes",
      "id": "8",
      "encoded": "en:peanuts",
      "check":false
    },
    {
      "name": "Sulfitos",
      "id": "9",
      "encoded": "en:sulphites",
      "check":false
    },
    {
      "name": "Crustáceos",
      "id": "10",
      "encoded": "en:crustaceans",
      "check":false
    },
    {
      "name": "Pescado",
      "id": "11",
      "encoded": "en:fish",
      "check":false
    },
    {
      "name": "Altramuces",
      "id": "12",
      "encoded": "en:lupin",
      "check":false
    },
    {
      "name": "Moluscos",
      "id": "13",
      "encoded": "en:molluscs",
      "check":false
    }
]
```

A.3. ProductCard.jsx

```
useEffect(() => {
    if (data == null || data === undefined) return;
    if (data.status === 0) {
      setIsValid(false);
      return;
    }
    // Fusiona ambos arrays
    const mergeAllergens = data.product.allergens_tags.concat(
      data.product.traces_tags,
    );
    // Recorre el conjunto previo y el array de configuración
    for (const tag of mergeAllergens) {
      for (const elemTag of config) {
        // Al encontrar el mismo nombre en configuración
        if (tag === elemTag.encoded && elemTag.check) {
          setEdible(false);
          setAllergensFound(elemTag.name);
        }
      }
    }
  });
```

A.4. conversor.py

```
import csv
import json
from tqdm import tqdm

def csv_to_json(csvFilePath, jsonFilePath):
  jsonArray = []

  with open(csvFilePath,"r") as csvf:
    data = csvf.readlines()

  header = data[0].strip().split("\t")

  for row in tqdm(data[1:]):
    items = row.strip().split("\t")
    jsonArray.append({
      "c": items[header.index("code")],
      "a": items[header.index("allergens")].split(','),
```

```
      "t": items[header.index("traces_tags")].split(',')
    })

  with open(jsonFilePath, "w") as jsonf:
    json.dump(jsonArray, jsonf)

csvFilePath = r"data.csv"
jsonFilePath = r"data.json"

csv_to_json(csvFilePath, jsonFilePath)
```

A.5. ProductScreen.jsx

```
  NetInfo.fetch().then((state) => {
    !state.isConnected
    || !state.isInternetReachable
    || state.details.cellularGeneration === '2g'
    || state.type === null
      ? setOffline(true)
      : setOffline(false);
  });

  function renderCard() {
    if (!offline) {
      return (
     <ProductCard data={response} config={config} navigation={navigation} />
      );
    }
    // Se recorre el json de showcase hasta encontrar el barcode en local
    for (let i = 0; i < db.length; i++) {
      if (db[i].code === barCode) {
        return <OfflineCard data={db[i]} config={config}
navigation={navigation} />;
      }
    }
    return <OfflineCard data={offlineItemNotFound} config={config}
navigation={navigation} />;
```

Bibliografía

[1] Alexa Diéguez. El Confidencial. “Dos millones de españoles, con alergia a los alimentos: qué ocurre”. url: https://www.alimente.elconfidencial.com/bienestar/2018-07-23/alergia-a-alimentos-el-riesgo-aumenta-en-verano_1586738/

[2] Ralph Jones. BBC Future. “Por qué están aumentando las alergias alimentarias”. url: https://www.bbc.com/mundo/vert-fut-54722040

[3] Yúbal Fernandéz. Xataka. “API: qué es y para qué sirve”. url: https://www.xataka.com/basics/api-que-sirve

[4] Teamleader. “¿Qué es y para qué sirve un diagrama de Gantt?”. url: https://www.teamleader.es/blog/diagrama-de-gantt

[5]: Ophélia. Yuka. “¿Cómo se ha constituido la base de datos?”. url: https://help.yuka.io/l/es/article/5a4z64amnk-constitucion-base-de-datos

[6] Directorate-General for Health and Food Safety (European Commission). RASFF. “RASFF annual report 2020”. url: https://op.europa.eu/en/publication-detail/-/publication/2acefdec-0486-11ec-b5d3-01aa75ed71a1/language-en/format-PDF/source-260930792

[7] Parlamento Europeo. “REGLAMENTO (UE) No 1169/2011 DEL PARLAMENTO EUROPEO Y DEL CONSEJO”. url: https://www.boe.es/doue/2011/304/L00018-00063.pdf

[8] Santalucía Impulsa.“Metodología Agile: ¿Qué es y para qué sirve?”. url: https://www.santaluciaimpulsa.es/metodologia-agile-que-es-para-que-sirve/

[9] Ionos. “Los web components en detalle”. url: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/web-components/

[10] MDN contributors. JavaScript. url: https://developer.mozilla.org/es/docs/Web/JavaScript

[11] Natalie Mclaren. Medium. “React Navigation 5: Stack, Tab, and Drawer All in One”. url: https://betterprogramming.pub/react-navigation-5-stack-tab-drawer-all-in-one-ead723188056

[12] Palco23. “Under Armour estudia la venta de la ‘app’ MyFitnessPal por 475 millones de dólares”. url: https://www.palco23.com/equipamiento/under-armour-estudia-la-venta-de-la-app-myfitnesspal-por-475-millones-de-dolares.html