UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

StayFresh

Presentado por Juan José Santos Cambra
en Universidad de Burgos — 20 de mayo de 2024
Tutor: Jesús Manuel Maudes Raedo y José Antonio Barbero Aparicio

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

El Dr. Jesús Manuel Maudes Raedo y el Dr. José Antonio Barbero Aparicio. Profesores del Departamento de Ingeniería Informática, Área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Juan José Santos Cambra, con DNI 71296402J, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 20 de mayo de 2024

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
Dr. Jesús Manuel Maudes Raedo	Dr. José Antonio Barbero Aparicio

Resumen

La gestión de los alimentos en el hogar es de vital importancia para reducir el desperdicio y así promover hábitos de consumo más sostenibles. Realizar un seguimiento manual de los productos almacenados y consumidos puede convertirse en una tarea tediosa además de propensa a errores, lo que finalmente resulta en el abandono y por ello, el desperdicio de dichos alimentos.

StayFresh ha sido desarrollada como una aplicación de código abierto para lidiar con este fenómeno. Esta aplicación integra las herramientas necesarias para registrar, monitorizar y organizar los alimentos almacenados en el hogar. Podemos tener acceso a través de su página de Github o de la Play Store si se está invitado.

Descriptores

Gestión de alimentos en el hogar, reducción del desperdicio de alimentos, automatización del seguimiento de alimentos, aplicación móvil para gestión alimentaria, monitoreo de fechas de caducidad, sostenibilidad y consumo responsable, organización de productos alimentarios.

Abstract

Food management at home is of vital importance to reduce waste and thus promote more sustainable consumption habits. Keeping track manually of stored and consumed products can become a tedious chore and be as well prone to errors, which ultimately results in the abandonment and therefore waste of said foods.

StayFresh has been developed as an open source application to deal with this phenomenon. This application integrates the necessary tools to record, monitor and organize food stored at home. It can be accessed through its Github page or from the Play Store if you have been invited.

Keywords

Home food management, food waste reduction, food tracking automation, mobile application for food management, expiration date monitoring, sustainability and responsible consumption, food product organization.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Uno de los desafíos más importantes a los que se enfrenta la humanidad es la gestión sostenible de los recursos naturales. La FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura) indica que alrededor de un tercio de la producción de los alimentos destinados al consumo humano se pierde o desperdicia en todo el mundo [8]. En 2023 se tiraron a la basura en toda España 2,8 millones de toneladas de comida y cada hogar desperdició una media de 61 kilos por persona [1].

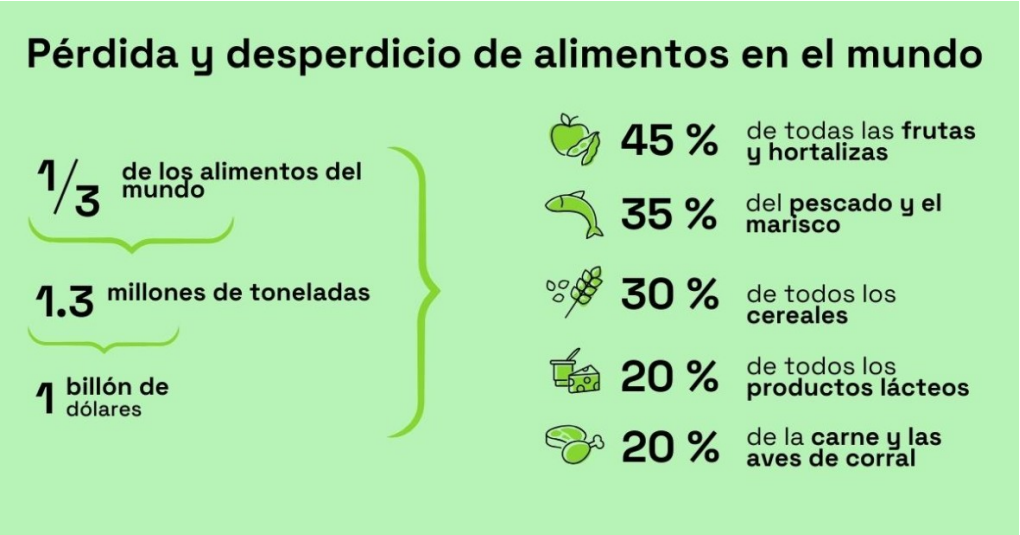

Figura 1.1: Desperdicio de alimentos

Además del impacto medioambiental que supone el desperdicio de alimentos también tenemos que añadir el impacto económico sobre una sociedad donde los alimentos sufrieron un incremento inflacionista del 15,7 % en 2022, del 7,3 % en 2023 y donde está siendo de un 3,3 % en 2024 [6].

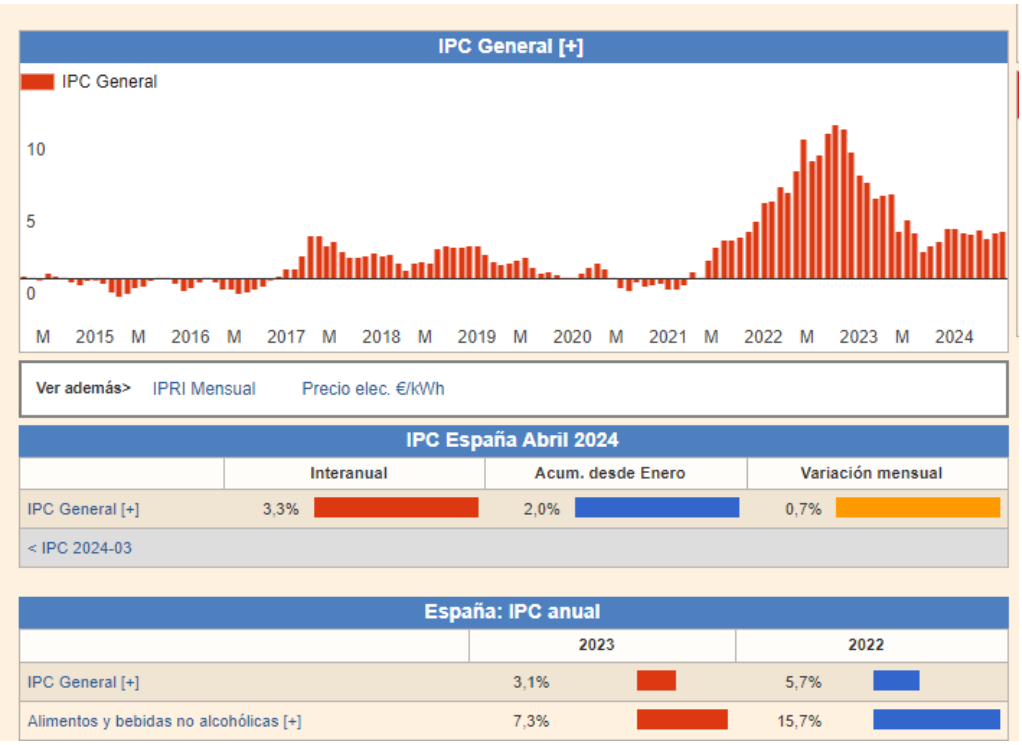

Figura 1.2: Inflación alimentos 2022-2024

En este contexto, es crucial el desarrollo de tecnologías que permitan un uso más eficiente de los alimentos. StayFresh nace con la idea de permitir a los usuarios tener un control rutinario de manera sencilla y amigable sobre los distintos productos almacenados contribuyendo así de manera directa a reducir su impacto sobre el medio ambiente. La aplicación permite a los usuarios definir una representación virtual de su hogar donde darán de alta los distintos espacios de almacenaje presentes en su vida cotidiana para llevar un orden en sus productos.
StayFresh incluye lectura de códigos de barras, gestión de fechas de caducidad, recomendación de productos basados en el histórico del usuario, gestión multiusuario por cada hogar, etc.

Figura 1.3: Logo de la aplicación

2. Objetivos del proyecto

A continuación se introducen los objetivos tanto generales como técnicos que han motivado el desarrollo de la aplicación.

2.1. Objetivos generales

1. Desarrollar una aplicación para smartphones en un framework *cross-platform* que permita la gestión del stock y caducidades de los productos perecederos del hogar.

2. Almacenar los datos de la aplicación en una plataforma cloud para que puedan ser compartidos entre distintos usuarios que conformen un mismo hogar.

3. Facilitar la introducción de datos de productos a través de lectura de códigos de barras de los productos a través de la cámara.

4. Realizar recomendaciones a la hora de introducir productos basadas en stocks pasados.

5. Permitir a los usuarios manejar varios hogares de manera simultánea y poder compartirlos de manera sencilla mediante generaciones y lecturas de códigos QR.

2.2. Objetivos técnicos

1. Desarrollar la aplicación en *React Native* usando *Javascript* como lenguaje de desarrollo.

2. Manejar el ecosistema *Expo* y *Eas* para el desarrollo y compilación.
3. Utilizar un modelo *Backend-as-a-Service* usando el Cloud de *Firebase* como proveedor.
4. Gestionar el registro y login de la aplicación con envío de emails a través de los servicios de *Firebase*.
5. Hacer uso de la API para gestión de la cámara del Smartphone y lectura de códigos de barras y QR.
6. Almacenar los datos de la aplicación en una base de datos de documentos NoSQL.
7. Gestionar la localización del idioma de la aplicación de manera automática.
8. Crear componentes no estándar, reutilizables y propios de la aplicación.
9. Gestionar el control de versiones de la aplicación a través del repositorio *GitHub*.

2.3. Objetivos personales

1. Aportar a la sostenibilidad de los hogares y el desperdicio de alimentos por falta de gestión individual.
2. Utilizar herramientas para la generación de mockups como *Figma*.
3. Aprender sobre las herramientas y flujo del desarrollo de aplicaciones móviles.
4. Adquirir conocimientos en un lenguaje que sea de utilidad tanto para aplicaciones móviles como para Web.
5. Manejar servicios tipo *BaaS* muy comunes en el desarrollo Web y Mobile.
6. Conocer la arquitectura de bases de datos de documentos no relacionales, NoSQL.

3. Conceptos teóricos

En el proyecto podemos encontrar dos conceptos teóricos principales: *Firebase* y *React Native*. Los dos requieren un mínimo desarrollo para poder comprender la estructura y funcionamiento general del aplicativo.

3.1. Firebase

Firebase es una plataforma web lanzada en 2011 y adquirida por Google en 2014 que ofrece un conjunto de herramientas basadas en un entorno Cloud que facilitan el desarrollo y escalabilidad de las aplicaciones. Ofrece distintos productos como: Hosting, almacenaje en la nube, servicios de autenticación y automatización de procesos, bases de datos en la nube, etc.

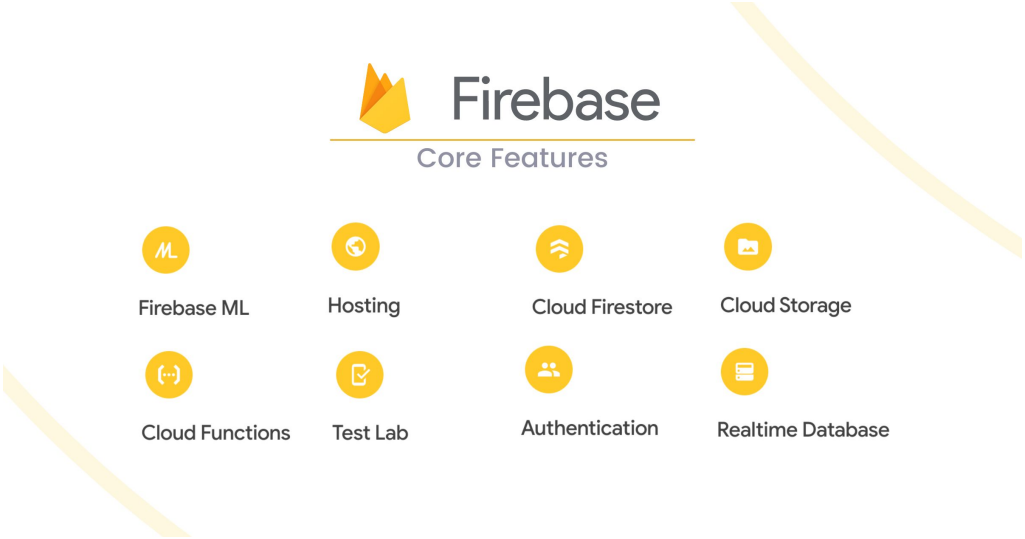

Figura 3.1: Qué es Firebase

Firestore

Firestore es uno de los productos que ofrece *Firebase* y consiste en una base de datos propietaria de Google que permite almacenar, sincronizar y consultar documentos NoSQL con una sintaxis y un SDK propio. *Firestore* nos proporciona una arquitectura con la cuál no es necesario el disponer de un backend tradicional si no que pueden ser los clientes los que realicen la comunicación directamente contra el almacenamiento de datos.

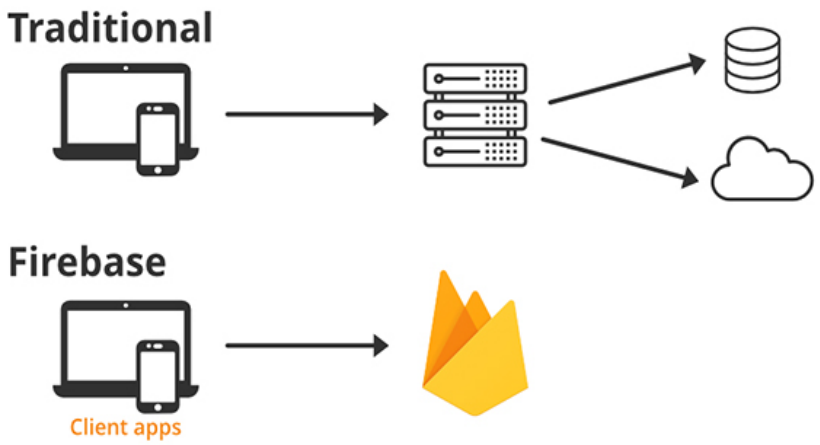

Figura 3.2: Qué es Firebase

Todas las comunicaciones con *Firestore* son seguras ya que implementan HTTPS y están basadas en HTTP/2. [9]. Este servicio ha sido elegido como base de datos para la aplicación.

Firebase Auth

Firebase Auth es otro de los servicios que proporciona *Firebase*. Se encarga de gestionar la autenticación de usuarios de una manera sencilla mediante distintos métodos: Email y contraseña, Google SignIn, Facebook, GitHub... exponiendo un SDK que facilita, unifica y transparenta la integración de todos ellos.

3.2. React Native

React Native es un framework de programación web de código abierto creado por Meta y usado para el desarrollo de aplicaciones en las plataformas móviles Android e iOS principalmente. La idea de este framework surgió tras identificar Facebook en 2012 que su apuesta por HTML5 para sus aplicaciones móviles había sido un error ya que carecía del rendimiento adecuado, en 2015, tres años más tarde, Facebook lanzó la primera versión del framework.
Los principios de funcionamiento de este framework son idénticos a los de React salvaguardando la principal diferencia de que React Native no manipula el DOM (Document Object Model). *React Native* implementa un servicio denominado *Bridge* que interpreta el código escrito por el desarrollador y lo comunica a las APIs de la plataforma nativa. Esto permite que una aplicación de *React Native* pueda ser desplegada tanto a la plataforma Android como a la plataforma iOS compartiendo una sola base de código común.

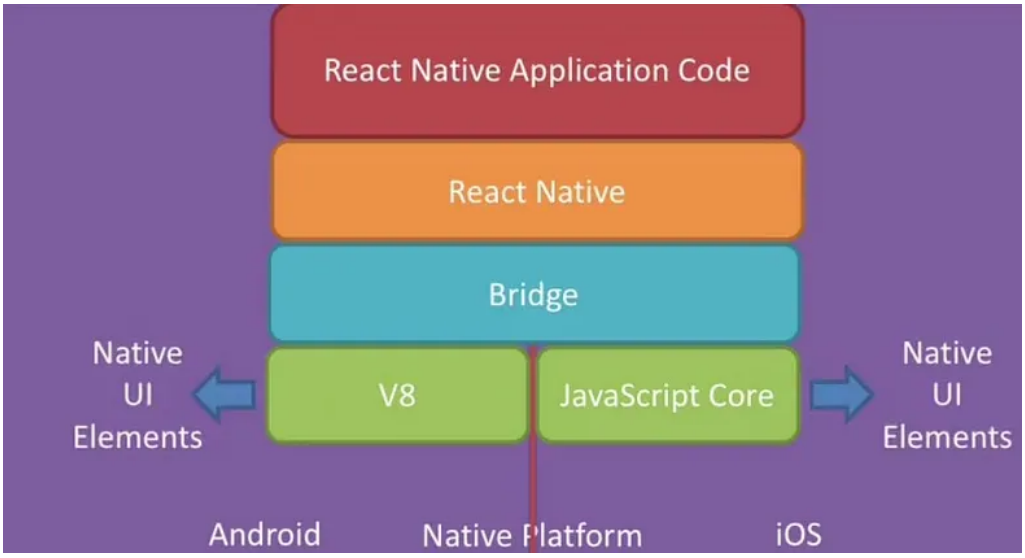

Figura 3.3: React Native Bridge

4. Técnicas y herramientas

4.1. Metodología

Scrum

Es un marco de gestión de proyectos de metodología ágil que ayuda a estructurar y gestionar el trabajo mediante un conjunto de valores, principios y prácticas. En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final [3].

GitFlow

Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales. Fue Vincent Driessen en nvie quien lo publicó por primera vez y quien lo popularizó. En lugar de una única rama main, este flujo de trabajo utiliza dos ramas para registrar el historial del proyecto. La rama main o principal almacena el historial de publicación oficial y la rama develop o de desarrollo sirve como rama de integración para las funciones [2].

4.2. Herramientas

Visual Studio Code

Visual Studio Code es un editor open source lanzado y mantenido por Microsoft desde el año 2015. Sus principales ventajas residen en su ligereza, su alto grado de personalización y el amplio rango de lenguajes de programación que soporta a través de su marketplace de extensiones. Está basado en Electron/Chromium.

NPM

También conocido como Node Package Manager cumple con dos propósitos principales, por un lado es el mayor registro de librerías software en el mundo incluyendo más de 800.000 entradas y por otro es un gestor de paquetes e instalador. Más específicamente en la aplicación actual ha sido utilizado para descargar y gestionar todas las dependencias externas necesarias. Se valoraron otras opciones adicionales para esta gestión como: Yarn, pnpm y bun, todas ellas son más rápidas en la descarga de paquetes que NPM y tienen un menor uso del espacio en disco al no descargar dependencias duplicadas pero se descartó su uso por incompatibilidades con ciertos paquetes como Expo.

Android Studio, AVD Manager, SDK Manager

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. No ha sido necesario el uso de Android Studio como IDE, sino como medio para utilizar las herramientas que este provee: AVD Manager y SDK Manager. La primera de ellas nos permite instalar dispositivos virtuales de Android que posteriormente pueden ser lanzados desde un terminal o desde Visual Studio Code usando extensiones como Android iOS Emulator sin tener que abrir Android Studio de nuevo. La segunda de las herramientas mencionadas nos permite gestionar las versiones el SDK de Android que tenemos instaladas en nuestro equipo y que son necesarias para correr la aplicación.

Expo CLI

Expo CLI es una herramienta disponible a través de la línea de comandos y que nos permite realizar acciones como: iniciar la aplicación en el emulador, enviar la aplicación a un dispositivo real a través de un servidor web, identificar qué paquetes están en estado deprecated dentro de nuestras dependencias o construir la aplicación enviando nuestros binarios a los servidores de EAS.

Git

Git es un software de control de versiones distribuido diseñado por Linus Torvalds y lanzado en el año 2005. Su ventajas principales son su naturaleza distribuida y su velocidad.

GitHub

GitHub es una plataforma para desarrolladores que permite crear, almacenar, gestionar y compartir código, utiliza Git como software de control de versiones distribuido. Fue lanzado en el año 2008 y adquirido por Microsoft en el año 2018. Además de las características relativas a su funcionamiento como repositorio de código también aporta otras funcionalidades adicionales como gestión de tickets o issues y automatización de flujos de trabajo y despliegues. En lo que atañe al proyecto actual, GitHub ha sido utilizado como repositorio de código y gestor de tareas e incidencias.

Code Climate

Code Climate es una herramienta orientada a analizar y mejorar el código. Provee de un análisis estático de la calidad de este donde se revisan distintas métricas como: mantenibilidad, complejidad, repetitividad... En el proyecto actual ha sido utilizado para revisar la calidad del código y obtener la métrica de mantenibilidad para la cuál se ha obtenido una A. Code Climate también provee de un badge que se puede usar en el readme de Github para actualizar visualmente la calidad de la aplicación cada vez que se realiza un commit. Se ha elegido esta herramienta frente a otras como SonarCloud o Sentry debido a su fácil integración con *React Native*.

Figma

Figma es una herramienta colaborativa basada principalmente en web para diseñar y prototipar interfaces. Se ha usado esta herramienta para generar los mockups de la aplicación. Se valoraron otras opciones como Adobe XD pero *Figma* ha sido preferida debido a su sencillez, elementos de la comunidad y a su ejecución sin instalación.

Excalidraw

Excalidraw es una herramienta open source web colaborativa para realizar diagramas. Se ha usado en el proyecto para generar distintos gráficos relativos al diseño de la aplicación. Incluye integración con la herramienta Mermaid la cual permite crear diagramas a partir de un pseudo lenguaje de programación consiguiendo así unificar el estilo de estos.

Overleaf

Overleaf es un editor de Latex online que incluye colaboración y control de versiones. Esta herramienta ha sido usada para generar la documentación del proyecto.

4.3. Bibliotecas

En esta sección se presentan algunas de las librerías más importantes de la aplicación.

React Native Paper

React Native Paper es una de las bibliotecas de componentes más usadas de React Native con unas 220.091 descargas semanales. Está basada en Material Design e incluye decenas de componentes como: Badges, Banners, Dividiers, TextInput... Se han valorado otras opciones como React Native Elements o Native Base. Esta biblioteca ha sido elegida debido a su sencilla instalación y a la cantidad de componentes personalizables de los que dispone.

Jest

Jest es un framework de testeo de código abierto desarrollado y mantenido por Facebook que se centra en la sencillez. Incluye características como Snapshot Testing para tomar "fotografías" del estado de los componentes, aislamiento para ejecutar pruebas de manera aislada y paralela, mocking integrado para no depender de recursos externos y cobertura de código. Se ha elegido esta biblioteca por ser la más extendida en React.

use-debounce

Cuando un usuario introduce datos en campos de manera manual y se ejecutan consultas dinámicas en segundo plano que responden a este input se puede producir un problema de rendimiento. Esto lo podemos ejemplificar cuando un usuario intenta filtrar un dato en un campo de búsqueda, si la aplicación intenta recalcular el filtro a la velocidad de tecleo del usuario, ante un gran número de datos, el resultado puede ser demasiado pesado de procesar y dar la sensación de retardo. Para evitar esta problemática se utilizan las funciones debounce. Este tipo de utilidades permiten ejecutar una tarea de manera retardada al detectar el final del input del usuario, la librería use-debounce expone un hook de React que permite gestionarlo de manera sencilla. Este paquete ha sido utilizado en todos los campos de búsqueda de la aplicación para así evitar cálculos innecesarios.

5. Aspectos relevantes del desarrollo del proyecto

En este apartado se revisarán los aspectos más interesantes del proyecto explicando las distintas tomas de decisiones que han llevado al producto final.

5.1. Origen del proyecto

La idea de este proyecto surge con el objetivo de cubrir una necesidad real. Hace unos años me mudé a un pueblo y con ello cambió mi filosofía a la hora de hacer la compra, pasé de hacer compras pequeñas casi cada día en la ciudad a realizar prácticamente una sola compra más grande y semanal para evitar desplazamientos innecesarios. Reducir la frecuencia y aumentar la cantidad conlleva también ser más precavido con la caducidad de los alimentos que se almacenan. Empecé a percatarme de que en repetidas ocasiones me encontraba productos caducados o al borde de la caducidad, esto me hizo cuestionarme cuántos alimentos se pueden llegar a desperdiciar por circunstancias similares. Como ya se ha visto en la introducción, se desperdicia casi un tercio del total de alimentos. Es por ello que surgió esta idea con el fin de poder utilizar la aplicación en mi hogar y cuantos lo crean necesario para tener un control sencillo de los stocks y las caducidades.

5.2. Elección de tecnologías

Lenguaje y framework

A día de hoy la oferta de tecnologías para el desarrollo de aplicaciones móviles es muy amplia. Se pueden dividir principalmente en dos grupos, aquellas que son nativas a una plataforma en concreto: Java con Android, Swift con iOS... y las que son multiplataforma: Flutter, React Native, Ionic, Xamarin... Las primeras fueron descartadas ya que no quería casar la aplicación con una plataforma en concreto, sino tener la posibilidad de, en un futuro, poder ampliar con los ajustes mínimos.

Figura 5.1: Top 10 lenguajes más populares para desarrollo móvil

Tras revisar las distintas opciones multiplataforma, las más usadas eran Flutter y React Native, originalmente la idea de usar Flutter me pareció muy atractiva debido a su paradigma basado en widgets y el lenguaje de programación Dart, el cuál Google presentó con el objetivo de que acabara sustituyendo a Javascript en el mundo de la web. Tras realizar mucha labor de investigación al respecto acabé decidiéndome por React Native por dos motivos principales. El primero es que ya había tenido contacto previo con Javascript y al ser el lenguaje web más extendido iba a tener más facilidad a la hora de encontrar soluciones a posibles problemas que se me fueran presentando.

El segundo es que tras leer mucho acerca de cómo Google saca del mercado repentinamente algunos de sus productos, temí que en cierto momento tuvieran la tentación de hacer lo mismo con Flutter, aunque solo son suposiciones. En el siguiente link se pueden ver todos los productos descontinuados por Google KilledByGoogle. Recientemente Google ha recortado parte del equipo del core de Flutter [4]. Este último motivo es el de menor peso pero fue el que decantó la balanza definitivamente.

Una vez elegido React Native quedaba la última decisión, usar Javascript o usar Typescript para el desarrollo ya que soporta ambos. Teniendo en cuenta mi conocimiento previo de Javascript aunque limitado y que muchos de los ejemplos y documentación en la red no se encuentran en Typescript todavía, opté por Javascript. Si tuviera que elegir uno de los dos lenguajes para desarrollar una aplicación empresarial que fuera a ser programada por un equipo, hubiera optado por Typescript sin duda alguna debido a su tipado estático ya que evitaría muchos conflictos entre diferentes estilos de programación. Pero no era el caso.

BaaS

La aplicación se planteó con un backend BaaS (Backend as a service). BaaS es un modelo de servicio en la nube en el que los desarrolladores subcontratan todos los aspectos en segundo plano de una aplicación web o móvil para que solo tengan que escribir y mantener el frontend [5]. Se eligió este formato debido a que la mayor parte de este tipo de proveedores incluyen en sus paquetes gratuitos las características que son necesarias para la arquitectura de este software lo que recorta el tiempo invertido y la complejidad de tener que desarrollar un backend a medida y posteriormente hostearlo, securizarlo con HTTPs, mantenerlo... Las necesidades principales aportadas por este servicio son:

- Gestión de base de datos.
- Autenticación de usuarios.
- Hosting.

Se valoraron dos opciones inicialmente, Supabase y Firebase. La primera de ellas es un proyecto de código abierto lanzado en 2023 que incluye una base de datos relacional PostgreSQL y distintos tipos de autenticación e integración con terceras partes. Su principal ventaja es que se puede hostear de manera privada sin depender de servicios externos aunque también se puede usar el cloud que proporciona Supabase de manera gratuita hasta cierto límite, la desventaja de este es que es un producto muy reciente en el mercado. La segunda de las opciones, Firebase, es un producto muy maduro y que lleva muchos años siendo usado por grandes compañías como Twitch, Glovo o Square entre otros. Esto hace que sea una herramienta que está muy documentada y para la cual se dispone de mucha información ya en la red, además, sumado el hecho de que dispone de una base de datos no relacional la hacía una solución perfecta para el aplicativo buscado.

5.3. Formación

El apartado formativo consistió en dos procesos diferentes de manera intensiva, está registrado en los tickets #1 y #5. Fue necesario que me formara en los dos aspectos clave del proyecto, React Native y Firebase.

React Native

React Native cuenta con una oferta online formativa muy extensa, para empezar, la propia página del framework ofrece un tutorial muy trabajado con el que además se puede interactuar y probar online. Además de esto, se utilizaron contenidos desarrollados por la propia comunidad como pueden ser cursos de Youtube.

Firebase

Se utilizó la documentación oficial de Firebase y distintos tutoriales de Youtube como aplicación tipo Todo y autenticación.

Javascript, html y css

También fue necesario refrescar conceptos básicos relacionados con estas tecnologías, para ello se utilizaron distintos recursos interactivos como Flexbox Adventure y documentación de la página de Mozilla Developers.

Snack

Para reforzar todos los conocimientos anteriormente mencionados se utilizó Snack, el entorno de testeo online que ofrece Expo para entrenar antes de tener preparado todo el entorno Snack.

5.4. Mockups

Desde el inicio se tuvo una idea clara acerca de cómo debía ser la interfaz de la aplicación de manera general aunque esta se fue depurando para mejorar la usabilidad con el tiempo. En las fases iniciales del proyecto se desarrollaron varias interfaces utilizando la herramienta Figma las cuales posteriormente se incluyeron como requisitos del proyecto. Ciertas pantallas como la principal finalmente no variaron demasiado del mockup a la aplicación final aunque sí se realizaron pequeños retoques estéticos o mejoras.

En la siguiente imagen podemos ver el mockup original de la pantalla home:

Figura 5.2: Pantalla home mockup

La implementación final de la pantalla home dentro de la aplicación:

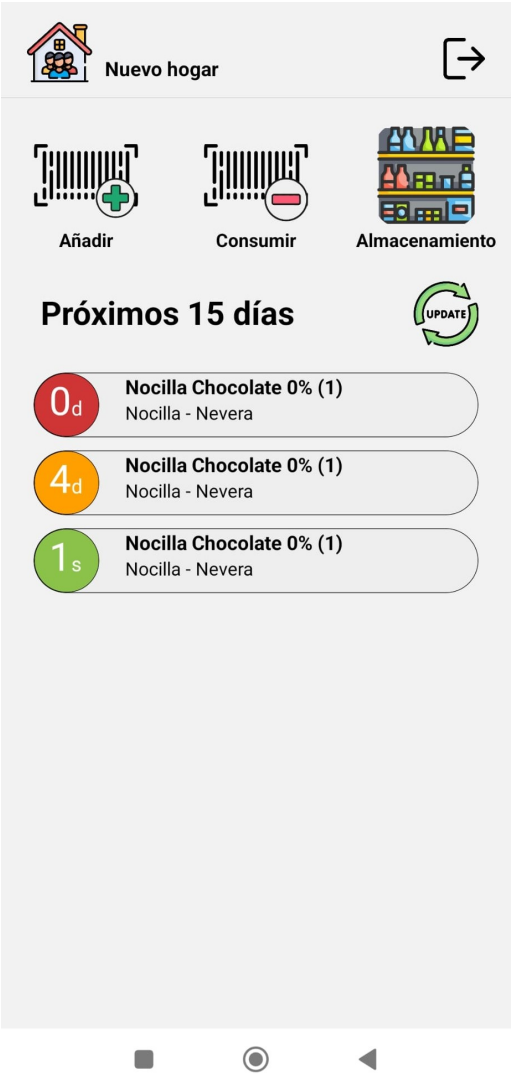

Figura 5.3: Pantalla home final

El resto de mockups los podemos encontrar en la sección cinco del apéndice B de los anexos.

5.5. Arquitectura del sistema

En el apartado cuatro del apéndice C de los anexos se presentan los distintos patrones de diseño de la aplicación, uno de los utilizados es Container/Presentational Components a través del cual se separa la lógica de la vista. Esto hace que tengamos componentes que están solo enfocados en la visualización y servicios que se encargan de gestionar la lógica y recuperar datos de Firebase. En el siguiente diagrama podemos ver cómo está estructurada la aplicación:

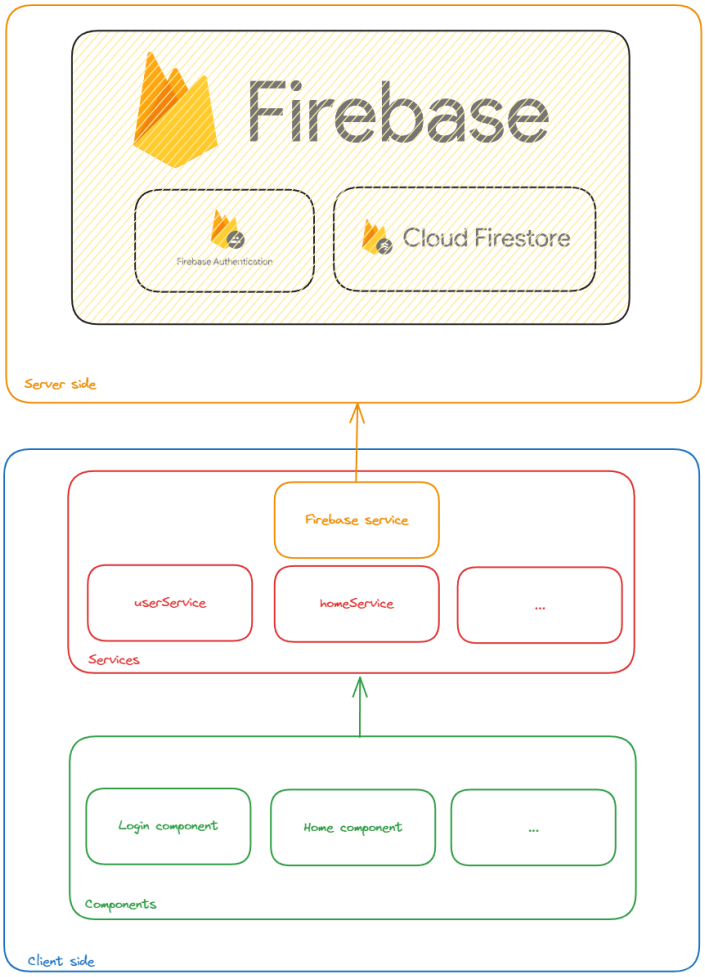

Figura 5.4: Arquitectura general

A la hora de diseñar la arquitectura se han tenido en cuenta todos los requisitos pero algunos han sido más determinantes que otros como:

- RF-1.1 Login con email: La aplicación debe permitir a los usuarios loguearse con email y contraseña
- RF-2.3 Escanear productos: La aplicación debe permitir escanear el código de barras para reconocer un producto ya en stock o darlo de alta usando la cámara del dispositivo.
- RF-3 Gestión de hogares: La aplicación debe poder gestionar los hogares del usuario.
- RNF-3 Escalabilidad: La aplicación debe estar preparada para poder escalar tanto en número de usuarios como en número de productos.
- RNF-4 Rendimiento: La aplicación debe tener tiempos de carga mínimos y transiciones óptimas entre pantallas.
- RNF-6 Persistencia de datos: La aplicación debe poder conservar los datos del usuario de manera independiente al dispositivo que este esté usando.

Todos los requisitos anteriormente mencionados han moldeado la aplicación de tal manera que tenga el diseño actual donde se utiliza la autenticación de Firebase así como Firestore. Inicialmente se plantearon otras arquitecturas basadas en un almacenamiento local pero quedaron descartadas tras comprobar que no eran compatibles con la solución buscada.

5.6. Gestión del proyecto

Como se ha explicado en el apartado de metodología, el marco de trabajo utilizado ha sido Scrum. El proyecto se ha finalizado en cuatro sprints de dos semanas cada uno:

- Sprint 0: Este sprint marca el inicio del proyecto. Se realiza la primera reunión con el tutor Jesús Manuel Maudes Raedo el día 11 de marzo para presentar la idea del proyecto, marcar la forma de trabajar, tecnologías a utilizar, hitos y entregables que cumplimentar.
- Sprint 1: En este sprint se inician las tareas de programación y ya se empiezan a ver los primeros commits en Github.
- Sprint 2: En ese sprint se continúa con el código de la aplicación centrándose en tener un producto funcional y completar los objetivos marcados. En este sprint se incorpora al proyecto el cotutor Jose Antonio Barbero Aparicio.
- Sprint 3: Este Sprint marca la recta final de la programación y la release 2.0.0 en un estado completamente funcional, con los objetivos marcados cumplidos y con nuevos añadidos fuera del alcance original.

Para gestionar todas las tareas a realizar en cada sprint se utilizaron las github issues junto con las milestone para agruparlas.

5.7. Desarrollo de la aplicación

El desarrollo de la aplicación se comenzó en el sprint 1 tras finalizar el proceso de formación, requisitos y tener el entorno de programación preparado. Durante el sprint 1 se realizaron las siguientes tareas:

- Se creó el esqueleto básico de la aplicación.
- Se descargaron todas las dependencias mínimas necesarias para el proyecto.
- Se añadió el linter de código Eslint con el ruleset de standard: En este punto se encontraron bastantes problemas con la compatibilidad de Expo y el linter inicialmente pero tras investigar se descubrió que Expo disponía de una forma concreta para instalar la librería [7].
- Se añadió la conexión a Firebase: Añadir este punto supuso cargar todas las Api keys que proporciona la herramienta y crear los objetos locales de autenticación y Firestore. A pesar de parecer una tarea crítica resultó muy sencilla debido a la cantidad de documentación y a lo maduro que es el producto.
- Se creó la pantalla de login: Durante la programación de esta pantalla se utilizaron los servicios de autenticación y Firebase y se probó por primera vez el envío de emails de registro lo cual resultó satisfactorio en las primeras pruebas. Se encontró un desafío al realizar la redirección automática al home screen si el usuario ya estaba autenticado en la app, esto fue debido al hecho de que la librería de autenticación funciona de forma asíncrona con un evento lo que generaba que siempre se mostrara la pantalla de login y unas décimas de segundo después se pasara el home screen. Para solucionar este problema se modificó el App.js para no cargar ninguna ventana hasta tener datos por parte de la librería de autenticación.
- Se creó la pantalla de home: Se añadió esta pantalla con los botones para redireccionar a otras pero al no estar todavía implementadas hubo que modificarlo posteriormente.

En el sprint 2 se continuó con el grueso del desarrollo implementando la mayor parte de pantallas y lógica restante y se generó la primera release de la aplicación:

- Se implementó la pantalla AddProductScreen: En esta pantalla se utilizó por primera vez la librería use-debounce con su hook para el apartado de búsquedas.
- Se implementó la pantalla CameraScreen: Se empezó a implementar utilizando la librería expo-barcode-scanner pero el expo doctor detectó la librería como deprecated y se pasó a usar expo-camera. Se programó la gestión de petición de permisos de uso de cámara al usuario.
- Implementar HomeManagementScreen: Se utilizó la ya creada CameraScreen para la lectura de códigos QR y las librería de generación de svg.
- Implementar ConsumeProductScreen: Esta pantalla resultó sencilla de implementar al ser la mayor parte común con la de AddProductScreen.
- Implementar StorageScreen: Resultó ser la programación más complicada de la aplicación debido a la gestión de los nodos anidados con sus tres niveles de profundidad pero al estar bien estructurado el diseño de base de datos de la aplicación fue sencillo gestionar el funcionamiento de los "descendientes".
- Release 1.0.0 de la aplicación: Se usó por primera vez EAS (Expo Application Services) para generar el APK con un build. Inicialmente la aplicación parecía funcionar igual que en modo depuración pero rápidamente llegó la mala noticia al no funcionar correctamente la pantalla de la cámara, al cargar se quedaba en un estado negro sin ningún tipo de feedback ni error. Tras muchas pruebas e investigación se descubrió que existía un bug en la librería de Expo y se solucionó incluyendo la importación del paquete expo-barcode-scanner aunque estuviera deprecated y no tuviera uso [10].

El sprint 3 fue en el que se generaron las releases 2.0.0 y 2.0.1 y se dio por concluido el desarrollo:

- Se añadió la rama y estructura de develop: Se consideró necesaria esta estructura para el desarrollo de los nuevos features y releases.
- Añadir test a la aplicación: Se añadieron test con Jest.
- Añadir pantalla onboarding a la aplicación: Tras recibir feedback del entorno cercano se consideró necesario añadir esta pantalla para mejorar la introducción a la aplicación.
- Mejoras a ventana de storage: Tras usar la aplicación de modo rutinario se encontraron mejoras que se aplicaron en este apartado, incluyendo por ejemplo la adición de productos directamente a un storage.
- Generación de la Release 2.0.0 y 2.0.1: Se encontraron distintos bugs en la aplicación que se fueron corrigiendo y generando releases con ello.

A continuación se muestran los KPIs principales del proyecto y repositorio:

Item	**Valor**
Commits	109
Issues	62
Tags	3
Ramas activas	2
Ramas mergeadas	6
Pull requests	5
Releases	3
Mantenibilidad del código	A

Tabla 5.1: Resumen del repositorio

5.8. Validación

Para asegurar la calidad y robustez de la aplicación se combinaron procesos con herramientas automáticas y validación humana. Al inicio del proyecto se añadió la dependencia de desarrollo Eslint junto con Prettier como linter para supervisar el seguimiento de reglas estándar de formato de código y así prevenir posibles errores derivados de la falta de consistencia.
En cuanto se tuvo la primera release, 1.0.0 se instaló la aplicación tanto en mi dispositivo Android como en el del entorno cercano para obtener datos de los posibles bugs que fueran surgiendo con el uso a modo de quality assurance recibiendo feedback directo de usuarios reales de la aplicación.
Una vez la aplicación estuvo más madura se introdujo un analizador de código estático, Code Climate para evaluar la calidad del código y corregir los code smells más preocupantes. También se añadieron test con Jest para asegurar que las futuras features y releases se validan correctamente antes de mergear a main.
Actualmente el repositorio cuenta con un buen estado de salud obteniendo una rating de mantenibilidad A:

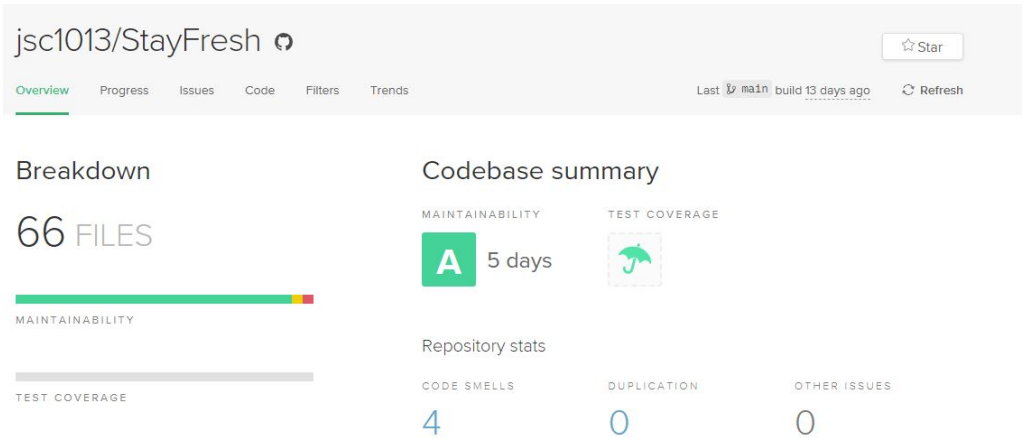

Figura 5.5: Valoración del repositorio

5.9. Usabilidad

En el apartado de usabilidad se tuvieron en cuenta ciertas leyes de diseño de UX para mejorar la experiencia [12].

1. **Los usuarios suelen percibir un diseño estéticamente agradable como un diseño más fácil de usar**: Esto se aplicó de manera general siguiendo esquemas de colores compatibles entre sí basados en el material palette, iconos no genéricos para identificar inequívocamente los distintos menús o animaciones para realizar ciertas acciones como el menú rápido para eliminar o actualizar productos.

2. **La productividad se dispara cuando un ordenador y sus usuarios interactúan a un ritmo (<400 ms) que garantiza que ninguno tenga que esperar al otro**: En este caso para evitar tiempos de espera innecesarios se introdujo el stack navigator que almacena componentes ya cargados para no tener que volver a generarlos, se optimizaron muchas de las interacciones con Firestore para que corrieran en segundo plano evitando efectos de retardo y se introdujo el debounce ya explicado anteriormente con un tiempo de 300 ms.

3. **Ley de Fitts. El tiempo de adquisición de un objetivo es función de la distancia al mismo y de su tamaño**: Esto lo podemos ver ejemplificado en los botones de navegación de la aplicación los cuales están a suficiente distancia entre sí y su tamaño es adecuado para el manejo desde un smartphone.

4. **Ley de proximidad. Los objetos cercanos o próximos tienden a agruparse**: En la pantalla de storage los alimentos estrán primeramente agrupados en contenedores al ser estos próximos en esa característica y a su vez están agrupados por alimento aunque tengan distinta fecha de caducidad.

5. **Paradoja del usuario activo. Los usuarios nunca leen los manuales, sino que empiezan a utilizar el programa inmediatamente**: Dentro de la aplicación se añadió la pantalla de onboarding que te da la bienvenida y muestra paso a paso el funcionamiento ya dentro de la aplicación sin necesidad de leer manuales. También se añadieron controles teniendo en cuenta a usuarios que no hayan prestado atención al funcionamiento como warnings al intentar entrar a pantallas sin realizar ciertos pasos previos, estos controles indican al usuario qué debe hacer primero para poder continuar.

Además de lo anteriormente mencionado se contemplaron otras dos opciones principales de usabilidad que están reflejadas en las issues #16 y #18.
La primera de ellas se centraba en la introducción de los datos de la fecha de caducidad de un producto, se propuso el uso de un OCR para reconocer la fecha, la idea quedó descartada por dos motivos principales:

1. Falta de estandarización: Tras comprobar distintos productos de distintas marcas se pudo apreciar que a día de hoy los envases no están ni mucho menos estandarizados en lo referente al etiquetado de la fecha de caducidad. Cada envase muestra la fecha en un formato diferente variando tamaños, colores y localización lo que complicaría mucho al usuario tener que fotografiar por cada producto introducido.

2. Supervisión: Si se implementara este OCR, el usuario tendría que además de fotografiar, esperar al resultado del procesado y posteriormente verificar que lo que la aplicación ha generado es correcto, pudiendo causar al usuario la necesidad de volver a revisar el envase en caso de no recordar la fecha.

La segunda de las opciones contemplaba utilizar speech to text para la introducción de productos, la idea finalmente se descartó por un factor de usabilidad al tener el usuario que verificar que la aplicación había reconocido correctamente los datos a través de la voz siendo más rápido el proceso de búsqueda.

5.10. Publicación en Google Play

Para poder compartir la aplicación y sus actualizaciones de manera sencilla se ha subido la aplicación a la Play Store, para ello ha sido necesario adquirir la licencia de google developer. En noviembre del 2023 Google modificó los requerimientos para publicar aplicaciones siguiente la siguiente clasificación [11]:

- Testeo interno: Antes de terminar de configurar la aplicación se pueden distribuir rápidamente versiones preliminares a un pequeño grupo de testers de confianza.
- Testeo cerrado: Con las pruebas cerradas se puede compartir la aplicación con un amplio grupo de usuarios. Esto permite solucionar problemas y asegurar que la aplicación cumple la política de Google Play antes de lanzarla. Se debe realizar una prueba cerrada antes de solicitar la publicación de la aplicación en producción.
- Testeo público: Permite sacar a la superficie la versión de prueba de la aplicación en Google Play. Si se ejecuta una prueba abierta, cualquiera puede unirse al programa de pruebas y enviar comentarios privados.
- Producción: Se pone la aplicación a disposición de miles de millones de usuarios en Google Play. Antes de poder solicitar la publicación de la aplicación en producción se debe realizar una prueba cerrada que cumpla los criterios de Google Play. Cuando se realiza la solicitud también se debe responder a algunas preguntas sobre la prueba cerrada. Para solicitar acceso a producción, al menos 20 testers deben estar inscritos en la prueba cerrada y deben haber participado de forma ininterrumpida durante los últimos 14 días.

Para la aplicación StayFresh no ha sido posible conseguir el criterio mínimo tan exigente que ha impuesto Google para poder poner la aplicación en producción al no haber podido obtener 20 testers que usen la aplicación de manera ininterrumpida durante 14 días. Por lo tanto, la aplicación ha sido dispuesta en un canal de testeo cerrado:

Figura 5.6: Canal de testeo cerrado

No ha sido posible poner la aplicación en un canal de testeo abierto ya que el criterio de Google es el mismo que para la publicación:

Figura 5.7: Requisitos prueba abierta

Se ha publicado en el canal de pruebas cerrado la versión 3 de la aplicación tras haber pasado los test y análisis requerido por Google. StayFresh tiene un total de 17805 dispositivos compatibles.

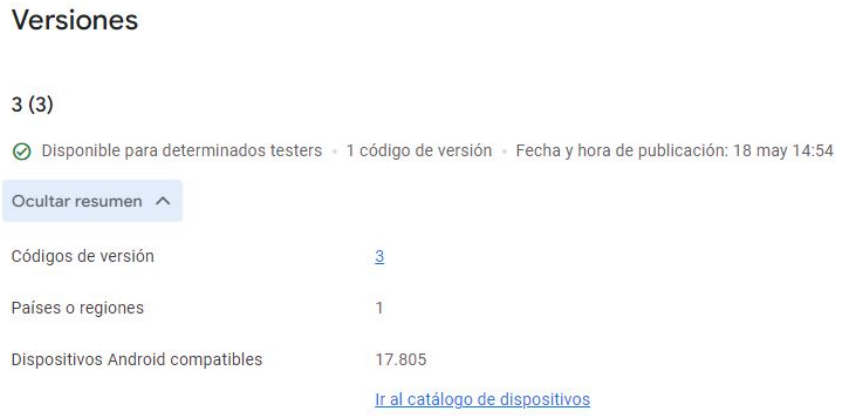

Figura 5.8: Version 3 de la aplicación publicada

6. Trabajos relacionados

En este apartado se presentarán otras aplicaciones especializadas en el mismo campo que StayFresh y se compararán sus características principales.

- Expiring Product Notifications: La aplicación se centra en la notificación de los productos que están cerca de la caducidad registrando los datos de modo local.
  - **Ventajas**: Incluye notificaciones y categorías de productos.
  - **Desventajas**: No incluye persistencia al no estar sincronizada en Cloud, no tiene gestión de almacenamientos, no incluye el concepto hogar y multiusuario, no permite escanear códigos de barras, no está en castellano y no es de código abierto.
- KitchenPal: Esta aplicación está más centrada en la lista de la compra y recetas.
  - **Ventajas**: Incluye base de datos con productos ya existentes, incluye lista de la compra, recetas y área con estadísticas generales.
  - **Desventajas**: Tiene versión de pago, no está en castellano y no es de código abierto.
- Your food: El objetivo de la aplicación es muy similar al de StayFresh.
  - **Ventajas**: Incluye lista de la compra
  - **Desventajas**: No incluye persistencia al no estar sincronizada en Cloud, no incluye el concepto hogar y multiusuario, no permite escanear códigos de barras, no está en castellano, la interfaz está demasiado recargada y no es de código abierto.

Del análisis anteriormente realizado se pueden extraer varias conclusiones sobre las carencias actuales y lo que aporta StayFresh en este campo. Algunas de las aplicaciones existentes, como Expiring Product Notifications o Your Food no incluyen sincronización con el Cloud, lo que significa que los datos no tienen persistencia en caso de pérdida. Tampoco incorporan el concepto de hogar ni multiusuario, lo que limita su utilidad en entornos familiares o compartidos. Otras aplicaciones como Kitchen Pal sí que incluyen este tipo de características pero solo bajo su versión de pago. Además de esto, hay que añadir que de estas aplicaciones presentadas, ninguna es de código abierto y tampoco están disponibles en español.
StayFresh se presenta como una solución de código abierto más completa y gratuita. Al contrario que las aplicaciones mencionadas, StayFresh ofrece sincronización con el Cloud, gestión de almacenamientos, el concepto de hogar y multiusuario, además de la capacidad de escanear códigos de barras para ingresar productos. A todo lo mencionado habría que añadir que StayFresh está disponible tanto en castellano como en inglés y es fácilmente ampliable a otros idiomas.

7. Conclusiones y Líneas de trabajo futuras

El desarrollo de este proyecto ha permitido la creación de una aplicación robusta y funcional, cumpliendo con los objetivos inicialmente establecidos. En lo personal me ha permitido explorar y desarrollar competencias aplicables tanto a entornos de programación web dada la naturaleza del framework React Native al estar basado en React, como móviles debido a la finalidad de la aplicación. He podido comprender los flujos de desarrollo de este tipo de aplicaciones y el funcionamiento de los servicios BaaS junto con el diseño de bases de datos no relacionales y sistemas de autenticación y persistencia de sesiones de usuario.
La programación de esta aplicación ha resultado en incluir StayFresh en la pantalla principal tanto de mi smartphone como en el del resto de personas con las que convivo mejorando significativamente la organización, comunicación y desperdicio de alimentos contribuyendo así a un futuro más sostenible reduciendo nuestro impacto.

Se han considerado una serie de mejoras futuribles para la aplicación con las cuáles ganaría mucha calidad.

- Generar base de datos global: Actualmente la aplicación utiliza datos locales a los usuarios relativos a los hogares que ocupan ya que se carece de una base de datos global de alimentos con sus códigos de barras. Esta mejora consistiría en procesar todos los datos que generan los usuarios en *Firestore* buscando códigos de barras repetidos para así crear de manera automatizada nuevos registros de productos que recomendar. Se podría generar una API con los productos como hacen otras empresas hoy en día, por ejemplo: FatSecret.
- Compatibilidad con Alexa: Se ha visto que podría ser interesante integrar la aplicación con plataformas como Amazon Alexa para utilizar ciertas herramientas como Lista de la compra de tal manera que tras consumir un alimento se pueda añadir su reemplazo.
- Notificaciones push: Para conseguir que la aplicación se convierta en un hábito sería buena idea incorporar notificaciones tipo push en dos casos:
  - Cuando un producto caduque pronto: De esta manera se conseguiría que hubiera una menor necesidad de abrir la aplicación cada cierto tiempo para revisar las próximas caducidades haciendo que la aplicación fuera menos pesada para el usuario.
  - Cuando otro usuario añada productos al hogar común: Esta funcionalidad sería interesante para sincronizar la actividad entre distintos usuarios de un mismo hogar y que todos estén actualizados con las últimas novedades.
- Informes y objetivos: Sería interesante incluir objetivos o rachas que los usuarios tengan que cumplir a modo de "minijuegos" de tal manera que sea más fácil habituarse al uso de la aplicación. Además, también se incluirían informes para que los usuarios tengan feedback de su contribución por el uso de la aplicación.

Bibliografía

[1] ABC. España tiró millones de toneladas de comida a la basura. https://www.abc.es/antropia/espana-tiro-basura-millones-toneladas-comida-basura-20240327133046-nt.html?ref=https%3A%2F%2Fwww.abc.es%2Fantropia%2Fespana-tiro-basura-millones-toneladas-comida-basura-20240327133046-nt.html, 2024. [Internet; accedido 02-mayo-2024].

[2] Atlassian. Flujo de trabajo gitflow. https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=%C2%BFQu%C3%A9%20es%20Gitflow%3F,vez%20y%20quien%20lo%20populariz%C3%B3., 2024. [Internet; accedido 09-mayo-2024].

[3] Atlassian. ¿qué es scrum? https://www.atlassian.com/es/agile/scrum, 2024. [Internet; accedido 09-mayo-2024].

[4] BusinessInsider. Google recorta equipos open source. https://www.businessinsider.es/google-recorta-equipos-desarrolladores-codigo-abierto-1383134, 2024. [Internet; accedido 10-mayo-2024].

[5] Cloudfare. Baas. https://www.cloudflare.com/es-es/learning/serverless/glossary/backend-as-a-service-baas/, 2024. [Internet; accedido 10-mayo-2024].

[6] Datosmacro. Ipc españa. https://datosmacro.expansion.com/ipc-paises/espana, 2024. [Internet; accedido 02-mayo-2024].

[7] Expo. Using eslint. https://docs.expo.dev/guides/using-eslint/, 2024. [Internet; accedido 12-mayo-2024].

[8] FAO. Desperdicio mundial de alimentos. https://openknowledge.fao.org/items/afbe7e5a-ce51-4931-b574-a584f9e398cd, 2012. [Internet; accedido 02-mayo-2024].

[9] Firebase. Http/2 comes to firebase hosting. https://firebase.blog/posts/2016/09/http2-comes-to-firebase-hosting, 2016. [Internet; accedido 04-mayo-2024].

[10] Github. Expo camera error. https://github.com/expo/expo/issues/21929, 2024. [Internet; accedido 12-mayo-2024].

[11] Google. App testing requirements for new personal developer accounts. https://support.google.com/googleplay/android-developer/answer/14151465#overview, 2024. [Internet; accedido 19-mayo-2024].

[12] LawsOfUX. Lawas of ux. https://lawsofux.com/, 2024. [Internet; accedido 12-mayo-2024].