



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**StayFresh
Documentación Técnica**



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

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	vi
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	11
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	17
B.3. Catálogo de requisitos	18
B.4. Especificación de requisitos	22
B.5. Mockups de interfaces	50
Apéndice C Especificación de diseño	57
C.1. Introducción	57
C.2. Diseño de datos	57
C.3. Diseño procedimental	63
C.4. Diseño arquitectónico	73
Apéndice D Documentación técnica de programación	80
D.1. Introducción	80
D.2. Estructura de directorios	80

<i>Índice general</i>	II
D.3. Manual del programador	81
D.4. Compilación, instalación y ejecución del proyecto	92
D.5. Buenas prácticas	93
D.6. Pruebas del sistema	94
Apéndice E Documentación de usuario	101
E.1. Introducción	101
E.2. Requisitos de usuarios	101
E.3. Instalación	102
E.4. Manual del usuario	103
Apéndice F Anexo de sostenibilización curricular	132
F.1. Introducción	132
F.2. Objetivos relacionados	132
F.3. Adherencia	134
F.4. Futuras líneas de trabajo	134
Bibliografía	135

Índice de figuras

A.1. Sprint 0 Roadmap	3
A.2. Sprint 0 Gráfica	4
A.3. Sprint 1 Roadmap	5
A.4. Sprint 1 Gráfica	6
A.5. Sprint 2 Roadmap	7
A.6. Sprint 2	8
A.7. Sprint 3 Roadmap	9
A.8. Sprint 3	10
A.9. Resumen milestones	11
B.1. Diagrama de casos de uso.	22
B.2. Login de la APP.	50
B.3. Página principal.	51
B.4. Página añadir producto.	52
B.5. Página consumir producto.	53
B.6. Página gestión de hogares.	54
B.7. Página gestión de ubicaciones.	55
B.8. Opciones en gestión de ubicaciones.	56
C.1. Diagrama de Firebase	60
C.2. Navegacion	64
C.3. Diagrama componente login	65
C.4. Diagrama componente home	66
C.5. Diagrama useTranslate	67
C.6. Diagrama componente addProduct	68
C.7. Diagrama componente consumeProduct	69
C.8. Diagrama componente camera	70
C.9. Diagrama componente homeManagement	71

C.10.Diagrama componente storage	72
C.11.Arquitectura general	74
C.12.Component composition	75
C.13.Diagrama componente storage	76
C.14.Atomic design	77
D.1. Creación de proyecto en Firebase	83
D.2. Habilitar autenticación Firebase	84
D.3. Indices de Firebase	84
D.4. Clonación de repositorio	86
D.5. Estructura tras inicialización de dependencias	87
D.6. Estrategia de ramas de desarrollo	88
D.7. npm start	90
D.8. Arrancar emulador Android	91
D.9. Paquete jest	94
D.10.Estructura de test	95
D.11.Ejemplo de test	96
D.12.Test script	97
D.13.Resultado de test	97
D.14.Resultado de cobertura	98
D.15.Github actions	99
D.16.Comprobaciones pasadas	100
E.1. Descarga de APK	102
E.2. Registro 1	103
E.3. Registro 2	104
E.4. Registro 3	105
E.5. Tutorial 1	106
E.6. Tutorial 2	107
E.7. Menú principal 1	109
E.8. Menú principal 2	110
E.9. Añadir un hogar 1	111
E.10.Añadir un hogar 2	112
E.11.Añadir un hogar 3	113
E.12.Añadir un hogar 4	114
E.13.Añadir un hogar 5	115
E.14.Añadir almacenamiento 1	116
E.15.Añadir almacenamiento 2	117
E.16.Añadir almacenamiento 3	118
E.17.Añadir productos 1	119
E.18.Añadir productos 2	120

E.19.Añadir productos 3	121
E.20.Añadir productos 5	121
E.21.Añadir productos 6	122
E.22.Resumen de productos 1	123
E.23.Resumen de productos 2	124
E.24.Resumen de productos 3	125
E.25.Gestión de almacenamiento 1	126
E.26.Gestión de almacenamiento 2	127
E.27.Gestión de almacenamiento 3	128
E.28.Consumir productos 1	129
E.29.Consumir productos 2	130
E.30.Categorización de productos 1	131

Índice de tablas

A.1. Resumen del repositorio	11
A.2. Costes totales salariales	12
A.3. Costes totales material y software	12
A.4. Costes totales del proyecto 5384€	13
A.5. Resumen posible precio de licencias	13
A.6. Resumen de licencias software	14
A.7. Resumen de licencias paquetes npm	14
B.1. CU-1 Autenticación y registro.	23
B.2. CU-2 Login con email.	24
B.3. CU-3 Creación de cuenta.	25
B.4. CU-4 Envío de email.	26
B.5. CU-5 Validación de email.	26
B.6. CU-6 Gestión de productos.	27
B.7. CU-7 Añadir productos.	27
B.8. CU-8 Consumir productos.	28
B.9. CU-9 Escanear productos.	29
B.10.CU-10 Recomendar productos.	30
B.11.CU-11 Editar fechas.	30
B.12.CU-12 Editar cantidades.	31
B.13.CU-13 Editar datos.	32
B.14.CU-14 Cálculo y clasificación de caducidad.	33
B.15.CU-15 Gestión de hogares.	33
B.16.CU-16 Mis hogares.	34
B.17.CU-17 Visualizar hogares.	34
B.18.CU-18 Cambiar nombre al hogar.	35
B.19.CU-19 Establecer hogar por defecto.	36
B.20.CU-20 Salir de un hogar.	37

B.21.CU-21 Ver Id del hogar.	38
B.22.CU-22 Copiar Id del hogar.	39
B.23.CU-23 Ver QR del hogar.	40
B.24.CU-24 Añadir hogares.	40
B.25.CU-25 Agregar hogar existente.	41
B.26.CU-26 Escanear hogar existente.	41
B.27.CU-27 Crear nuevo hogar.	42
B.28.CU-28 Gestión de ubicaciones.	42
B.29.CU-29 Visualizar ubicaciones.	43
B.30.CU-30 Añadir ubicaciones.	43
B.31.CU-31 Eliminar ubicaciones.	44
B.32.CU-32 Editar ubicación.	45
B.33.CU-33 Ordenar ubicaciones.	46
B.34.CU-34 Buscar en ubicaciones.	47
B.35.CU-35 Mover productos entre ubicaciones.	48
B.36.CU-36 Resumen de productos.	49

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apéndice se proporcionará el detalle de la planificación cronológica del desarrollo del proyecto así como un estudio de su viabilidad económica y legal.

En el primer apartado se mostrarán los plazos en los que el proyecto ha sido desarrollado así como la distribución de las tareas.

El segundo apartado está dividido en dos estudios diferentes:

- Viabilidad económica: Se realizará una estimación de los costes y beneficios que podría haber tenido el proyecto en caso de haber sido real.
- Viabilidad legal: Se analizarán las leyes aplicables al software que puedan influenciar en la aplicación.

A.2. Planificación temporal

El desarrollo del proyecto ha sido realizado a través de la metodología ágil Scrum con sprints de duración de dos semanas. No se han tenido en cuenta para la planificación reuniones de equipo por carecer de aplicación real. El seguimiento de la metodología ágil no ha sido totalmente homogéneo a lo largo del tiempo debido a que no se ha podido contar con unas horas de trabajo diarias dedicadas como se tendría en un entorno profesional.

A continuación se detallan los sprint realizados durante el proyecto junto con la compilación de las tareas realizadas y los objetivos.

Sprint 0: 12 de marzo a 26 de marzo

Es el sprint que marca el inicio del proyecto. Se realizó una 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.

Durante este primer sprint los objetivos principales fueron:

- Preparar el repositorio de Github.
- Aprender la tecnología React Native. #1.
- Generar los mockups de la aplicación. #2.
- Definir los objetivos de la aplicación. #3.
- Generar el proyecto en Overleaf. #4.
- Estudiar el funcionamiento y pruebas con *Firebase*. #5.
- Preparar el entorno de programación y simulación. #11.
- Generar los requisitos funcionales. #14.

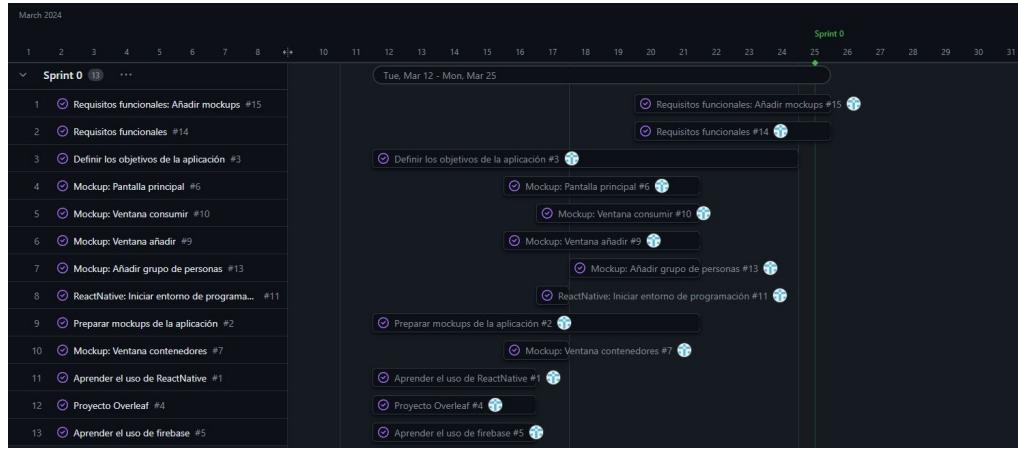


Figura A.1: Sprint 0 Roadmap

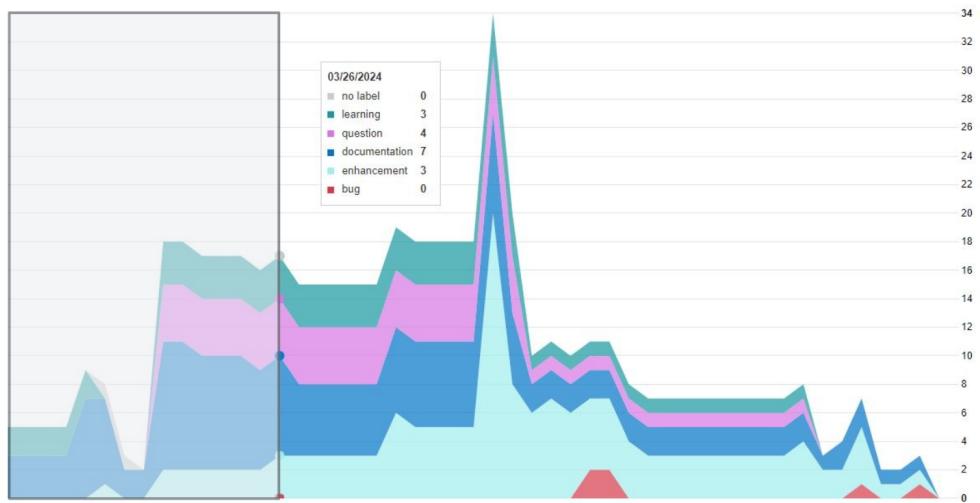


Figura A.2: Sprint 0 Gráfica

Sprint 1: 26 de marzo a 9 de abril

En este sprint se inician las tareas de programación y ya se empiezan a ver los primeros commits en Github, los objetivos principales del sprint fueron:

- Realizar esqueleto básico de la aplicación.
- Iniciar programación de la aplicación. [#21](#).
- Descarga de paquetes de dependencias necesarios. [#23](#).
- Crear pantalla login. [#24](#).
- Añadir conexión a firebase. [#25](#).
- Añadir linter de código. [#26](#).
- Crear pantalla home. [#37](#).

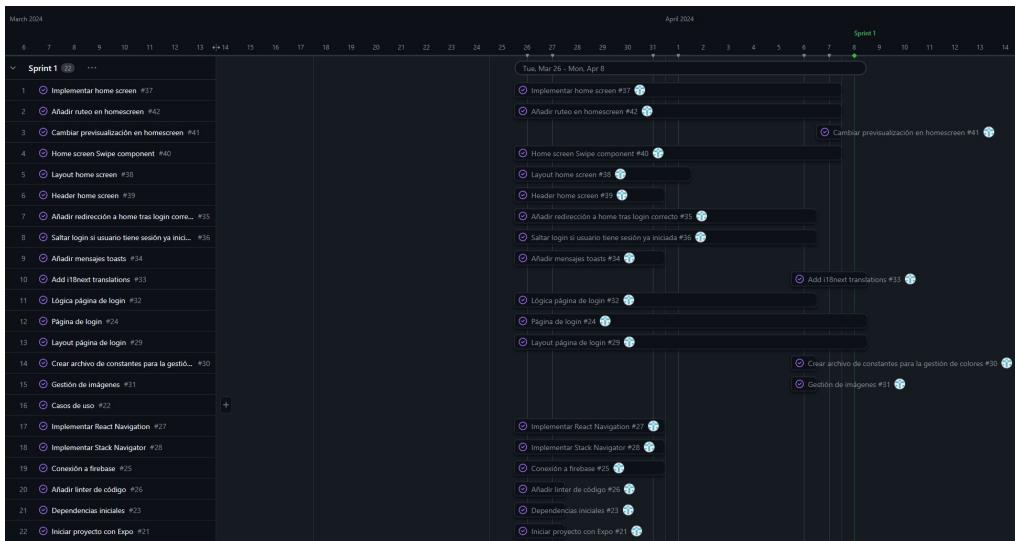


Figura A.3: Sprint 1 Roadmap

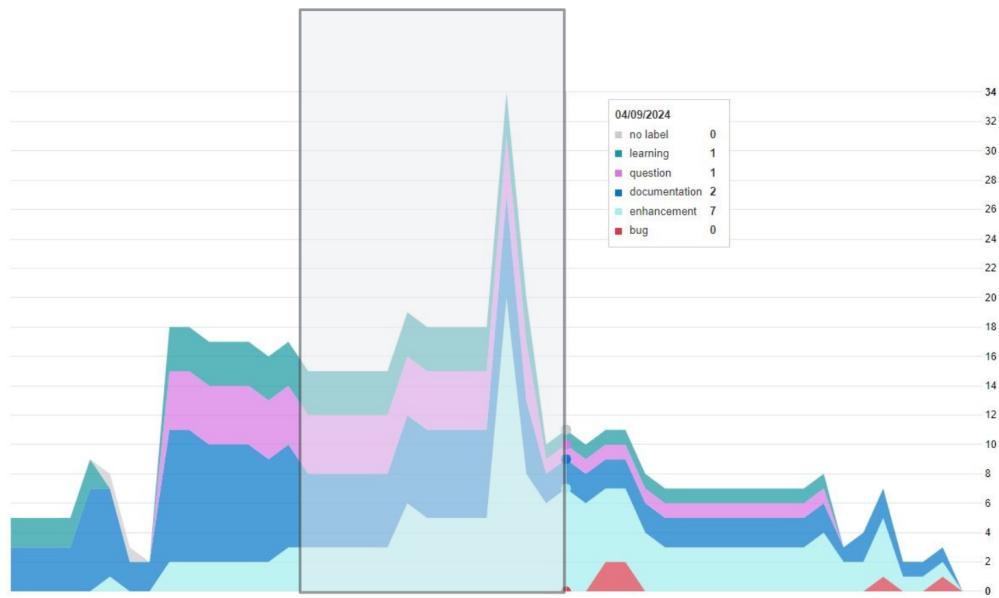


Figura A.4: Sprint 1 Gráfica

Sprint 2: 9 de abril a 23 de abril

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 José Antonio Barbero Aparicio. Los objetivos principales del sprint fueron:

- Implementar AddProductScreen. #43.
- Implementar CameraScreen. #45.
- Implementar HomeManagementScreen. #46.
- Implementar ConsumeProductScreen. #48.
- Implementar StorageScreen. #50.
- Release 1.0.0 de la aplicación. **Release 1.0.0**.

Se empiezan a corregir los primeros bugs que van apareciendo en la aplicación tras la codificación: #55 y #56

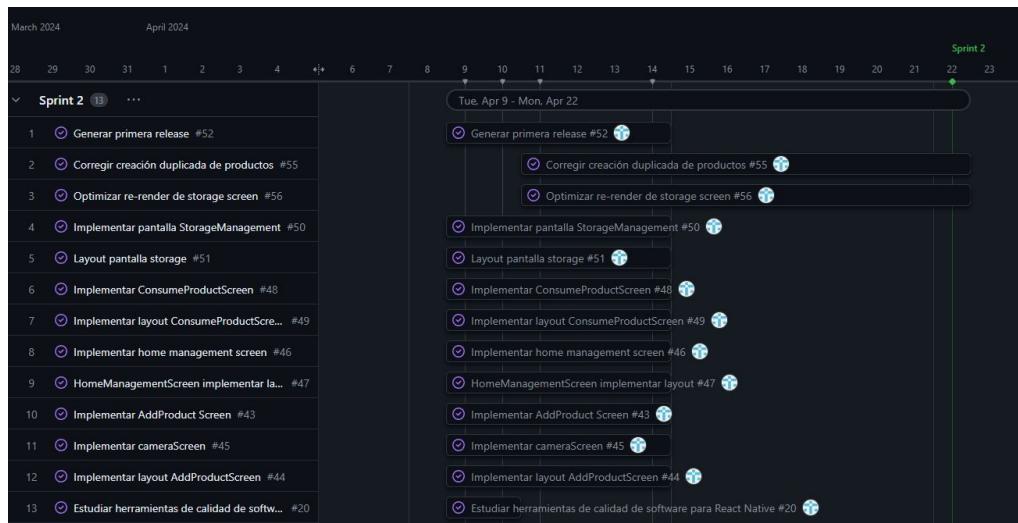


Figura A.5: Sprint 2 Roadmap

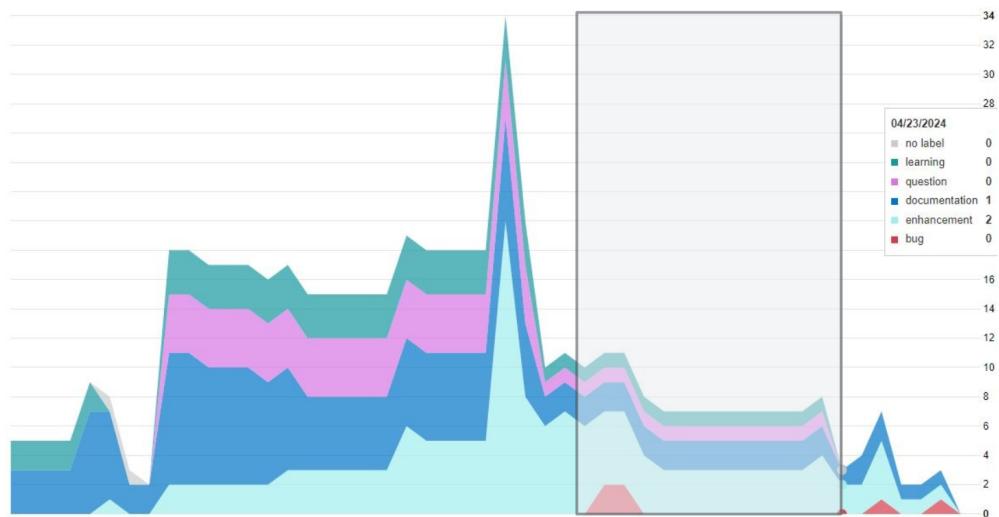


Figura A.6: Sprint 2

Sprint 3: 23 de abril a 6 de mayo

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. Los objetivos principales del sprint fueron:

- Añadir test a la aplicación. [#17](#).
- Añadir pantalla onboarding a la aplicación. [#53](#).
- Añadir rama y estructura de develop. [#57](#).
- Mejoras a ventana de storage.
- Generación de la Release 2.0.0. [Release 2.0.0](#).
- Documentación.
- Correcciones y nuevas Releases bugfix. [Release 2.0.1](#).

La pantalla de onboarding no estaba planificada originalmente pero se cambió el alcance de la aplicación ya que consideró un buen añadido para la usabilidad de los usuarios que entran por primera vez a la aplicación.

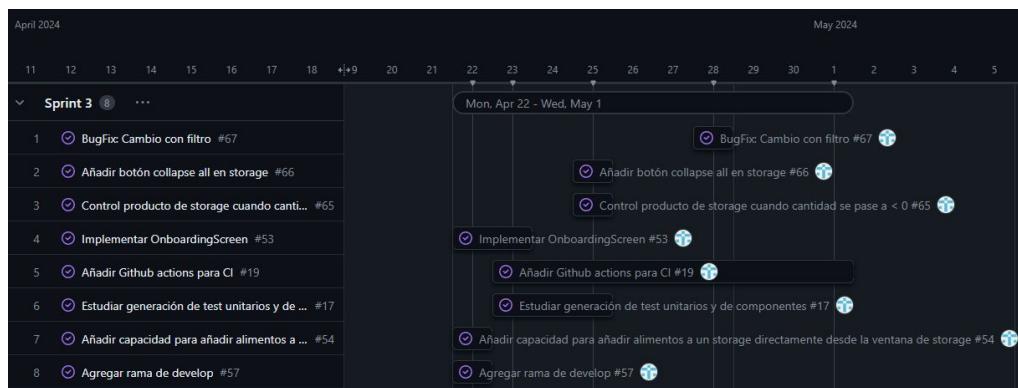


Figura A.7: Sprint 3 Roadmap

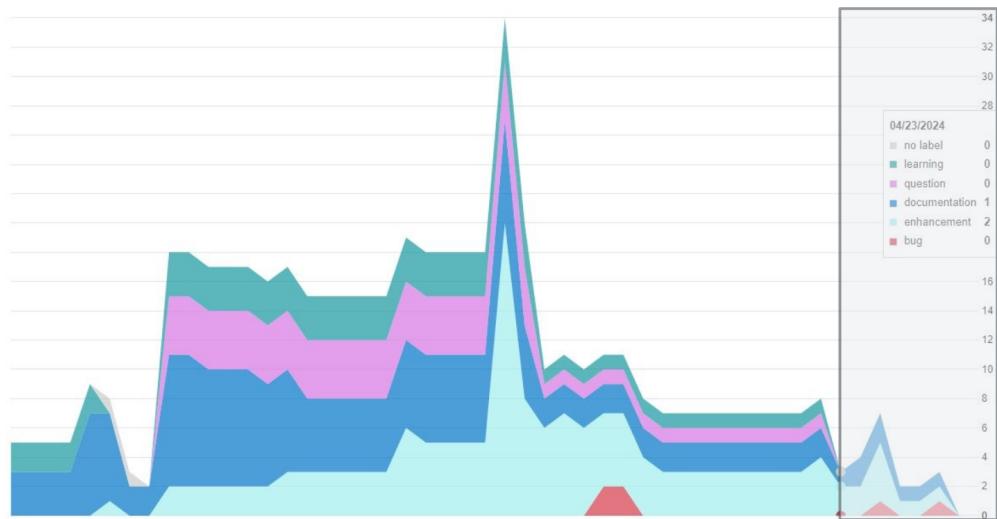


Figura A.8: Sprint 3

Resumen

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 A.1: Resumen del repositorio



Figura A.9: Resumen milestones

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se analiza la viabilidad económica del proyecto si se hubiera realizado en una empresa real.

Primero se analizarán los costes salariales:

Concepto	Coste
Salario Total	2000€
Contingencias comunes (23,60 %)	472€
Desempleo (5,5 %)	110€
Fondo de Garantía Salarial (0,2 %)	4€
Formación profesional (0,6 %)	12€
Mecanismo de Equidad Intergeneracional (0,58 %)	11,6€
Salario neto empleado	1390,4€
Total meses	2
Coste total salarial	4000€

Tabla A.2: Costes totales salariales

Los datos de los porcentajes relativos a las cotizaciones han sido obtenidos de la siguiente fuente de la Seguridad Social. [14].

También se añaden los costes materiales y de software:

Concepto	Coste
Ordenador	700€
Licencia Windows 10	259€
Smartphone	300€
Internet	75€
Luz	50€
Coste total material y software	1384€

Tabla A.3: Costes totales material y software

Costes totales del proyecto

Concepto	Coste
Coste total salarial	4000€
Coste total material	1384€
Costes totales del proyecto	5384€

Tabla A.4: Costes totales del proyecto 5384€

Beneficios

El proyecto está considerado como Open Source de tal manera que cualquier persona con la documentación de la aplicación pueda hostear su propio servicio por lo que no se monetizaría por esa vía. La vía por la que se podría monetizar la aplicación es ofreciendo una subscripción anual en la Play Store de tal manera que la gente que no tenga los conocimientos informáticos suficientes para hostear la aplicación pueda seguir obteniendo los beneficios de esta. La licencia anual de la aplicación por usuario podría seguir el siguiente patrón:

Concepto	Licencia anual general (1 hogar)	Por hogar extra
Precio	1,99€	0,99€

Tabla A.5: Resumen posible precio de licencias

Estas licencias anuales servirían para contratar el servicio prepago de *Firebase* y obtener beneficio por encima de este.

Viabilidad legal

En esta sección se estudiarán los aspectos legales de las licencias de los distintos software utilizados así como los recursos y los datos almacenados de los usuarios según la LOPD.

Licencias

A continuación se resumen los software utilizados junto con sus licencias:

Software	Licencia
Expo	MIT
Visual studio code	MIT
NPM	npm license

Tabla A.6: Resumen de licencias software

Licencias de todos los paquetes npm utilizados:

Paquete	Licencia
Babel	MIT
Expo	MIT
Firebase Sdk	Apache-2.0 y BSD-3-Clause
i18next	MIT
react-i18next	MIT
Jest	MIT
React	MIT
React-Native	MIT y CC-BY-4.0
use-debounce	MIT

Tabla A.7: Resumen de licencias paquetes npm

Recursos

En el apartado de recursos, solo ha sido necesario utilizar iconografía para la aplicación.

Iconografía

Los iconos usados en la aplicación se han obtenido de la web [Flaticon](#), estos están sujetos a la licencia de *Creative Commons* con atribución, por lo tanto se pueden compartir o adaptar en aplicaciones tanto personales como comerciales siempre que se atribuya la autoría [3]. A continuación se atribuyen todos los iconos utilizados en la aplicación:

- [Icono más y menos](#)
- [Icono código de barras](#)
- [Icono estantería](#)
- [Icono casa](#)
- [Icono editar](#)
- [Icono borrar](#)
- [Icono cámara](#)
- [Icono cancelar](#)
- [Icono actualizar](#)
- [Icono QR](#)

El logo de la aplicación fue generado mediante el uso de la inteligencia artificial DALL-E el cuál nos indica que cualquier imagen generada nos pertenece independientemente del plan contratado [13].

LOPD

En la aplicación solo se trata un dato relativo a los usuarios y es el email. El email está considerado un dato personal dada su definición: «*Los datos personales son cualquier información relativa a una persona física viva identificada o identifiable. Las distintas informaciones, que recopiladas pueden llevar a la identificación de una determinada persona, también constituyen datos de carácter personal*» [2].

La naturaleza de la aplicación por el momento es open source siendo el hosteado de los datos llevaba a cabo en *Firebase* bajo la responsabilidad de cada persona que quiera utilizar el código, la aplicación no aporta una cuenta ni base de datos de *Firestore* por defecto. Se puede encontrar la aplicación como prueba cerrada en [Play Store](#) y con acceso limitado solo para facilitar la descarga a los usuarios elegidos durante el testing. En caso de querer comercializar y exponer de manera pública la aplicación tendremos que tener en cuenta lo siguiente relativo a los datos personales de los usuarios [4]:

1. Licitud, lealtad y transparencia
2. Limitación de la finalidad
3. Minimización de los datos
4. Exactitud
5. Limitación del plazo de conservación
6. Integridad y confidencialidad

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se desarrolla la especificación de requisitos del sistema. Se introducirán unos objetivos generales que debe alcanzar la aplicación y se recopilará un catálogo de distintos requisitos funcionales y no funcionales junto con sus casos de uso.

B.2. Objetivos generales

A continuación, se enumeran los objetivos generales del proyecto.

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 la lectura de códigos de barras de los productos usando 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 forma sencilla mediante generaciones y lecturas de códigos QR.

B.3. Catálogo de requisitos

A continuación, se enumeran los requisitos funcionales del proyecto partiendo de los objetivos generales de este.

Requisitos funcionales

- RF-1 Autenticación y registro: La aplicación debe permitir a los usuarios autenticarse o loguearse.
 - RF-1.1 Login con email: La aplicación debe permitir a los usuarios loguearse con email y contraseña.
 - RF-1.2 Creación de cuenta: La aplicación debe permitir a los usuarios crear una cuenta con email y contraseña.
 - RF-1.2.1 Envío de email: La aplicación debe enviar un mensaje de confirmación al correo electrónico que el usuario indique al crear la cuenta con un vínculo de confirmación.
 - RF-1.2.2 Validación de email: La aplicación debe validar la cuenta del usuario una vez haya abierto el vínculo de confirmación.
- RF-2 Gestión de productos: La aplicación debe poder gestionar los productos del usuario.
 - RF-2.1 Añadir productos: La aplicación debe permitir al usuario agregar productos con las propiedades: código de barras, marca, nombre, cantidad, ubicación y fecha de caducidad.
 - RF-2.2 Consumir productos: La aplicación debe permitir al usuario consumir una cantidad determinada de los productos que tiene en stock.
 - 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-2.4 Recomendar productos: La aplicación debe dar recomendaciones al usuario basadas en su propio historial de productos a la hora de añadir y en los productos en stock a la hora de consumir.
 - RF-2.5 Editar fechas: La aplicación debe permitir editar las fechas de caducidad de los productos que están introducidos en el stock.

- RF-2.6 Editar cantidades: La aplicación debe permitir editar las cantidades de los productos que están introducidos en el stock.
 - RF-2.7 Editar datos: La aplicación debe permitir editar el nombre, marca y código de barras de un producto
 - RF-2.8 Cálculo y clasificación de caducidad: La aplicación debe calcular y clasificar los productos según los días restantes hasta su caducidad.
- RF-3 Gestión de hogares: La aplicación debe poder gestionar los hogares del usuario.
- RF-3.1 Mis hogares: La aplicación debe poder gestionar los hogares ya asociados al usuario.
 - RF-3.1.1 Visualizar hogares: La aplicación debe permitir al usuario visualizar todos los hogares que tenga añadidos.
 - RF-3.1.2 Cambiar nombre al hogar: La aplicación debe permitir personalizar el nombre de cada hogar de manera individual al usuario sin afectar al resto de usuarios.
 - RF-3.1.3 Establecer hogar por defecto: La aplicación debe permitir establecer un hogar por defecto al abrir la aplicación
 - RF-3.1.4 Salir de un hogar: La aplicación debe permitir a un usuario salir de un hogar.
 - RF-3.1.5 Ver Id del hogar: La aplicación debe permitir a un usuario visualizar el Id asociado a un hogar.
 - RF-3.1.6 Copiar Id del hogar: La aplicación debe permitir a un usuario copiar el Id asociado a un hogar.
 - RF-3.1.7 Ver QR del hogar: La aplicación debe permitir visualizar un código QR asociado al Id de un hogar del usuario.
 - RF-3.2 Añadir hogares: La aplicación debe poder permitir añadir nuevos hogares al usuario.
 - RF-3.2.1 Agregar hogar existente: La aplicación debe permitir añadir un hogar ya existe al usuario.
 - RF-3.2.2 Escanear hogar existente: La aplicación debe permitir escanear un código QR de un hogar ya existente.
 - RF-3.2.3 Crear nuevo hogar: La aplicación debe permitir poder crear nuevos hogares.
- RF-4 Gestión de ubicaciones:

- RF-4.1 Visualizar ubicaciones: La aplicación debe permitir visualizar todas las ubicaciones del usuario junto con sus productos asociados de manera anidada.
 - RF-4.2 Añadir ubicaciones: La aplicación debe permitir añadir nuevas ubicaciones.
 - RF-4.3 Eliminar ubicaciones: La aplicación debe permitir eliminar ubicaciones existentes junto con sus productos asociados.
 - RF-4.4 Editar ubicación: La aplicación debe permitir editar el nombre de una ubicación.
 - RF-4.5 Ordenar ubicaciones: La aplicación debe permitir ordenar las ubicaciones de manera alfabética o por fecha de caducidad. Tendrá en cuenta una ordenación atendiendo a los productos que esta tenga.
 - RF-4.6 Buscar en ubicaciones: La aplicación debe permitir una búsqueda global de tal forma que se tenga en cuenta tanto el texto de la ubicación como el texto de los productos.
 - RF-4.7 Mover productos entre ubicaciones: La aplicación debe permitir mover productos entre ubicaciones.
- RF-5 Resumen de productos: La aplicación debe facilitar un resumen editable en fechas con opciones predeterminadas donde se muestre la información de los productos que caducan en ese periodo. También permitirá la edición de estos.

Requisitos no funcionales

- RNF-1 Internacionalización: La aplicación debe estar preparada para ser traducida a distintos idiomas.
 - RNF-1.1 Inglés: La aplicación debe estar traducida al idioma inglés.
- RNF-2 Interfaz de usuario: La aplicación debe tener una interfaz sencilla y minimalista enfocándose en la agilidad de uso.
- 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-5 Uso de la cámara: La aplicación debe solicitar acceso a la cámara del usuario y recordar sus preferencias.
- RNF-6 Persistencia de datos: La aplicación debe poder conservar los datos del usuario de manera independiente al dispositivo que este esté usando.

B.4. Especificación de requisitos

Diagrama de casos de uso

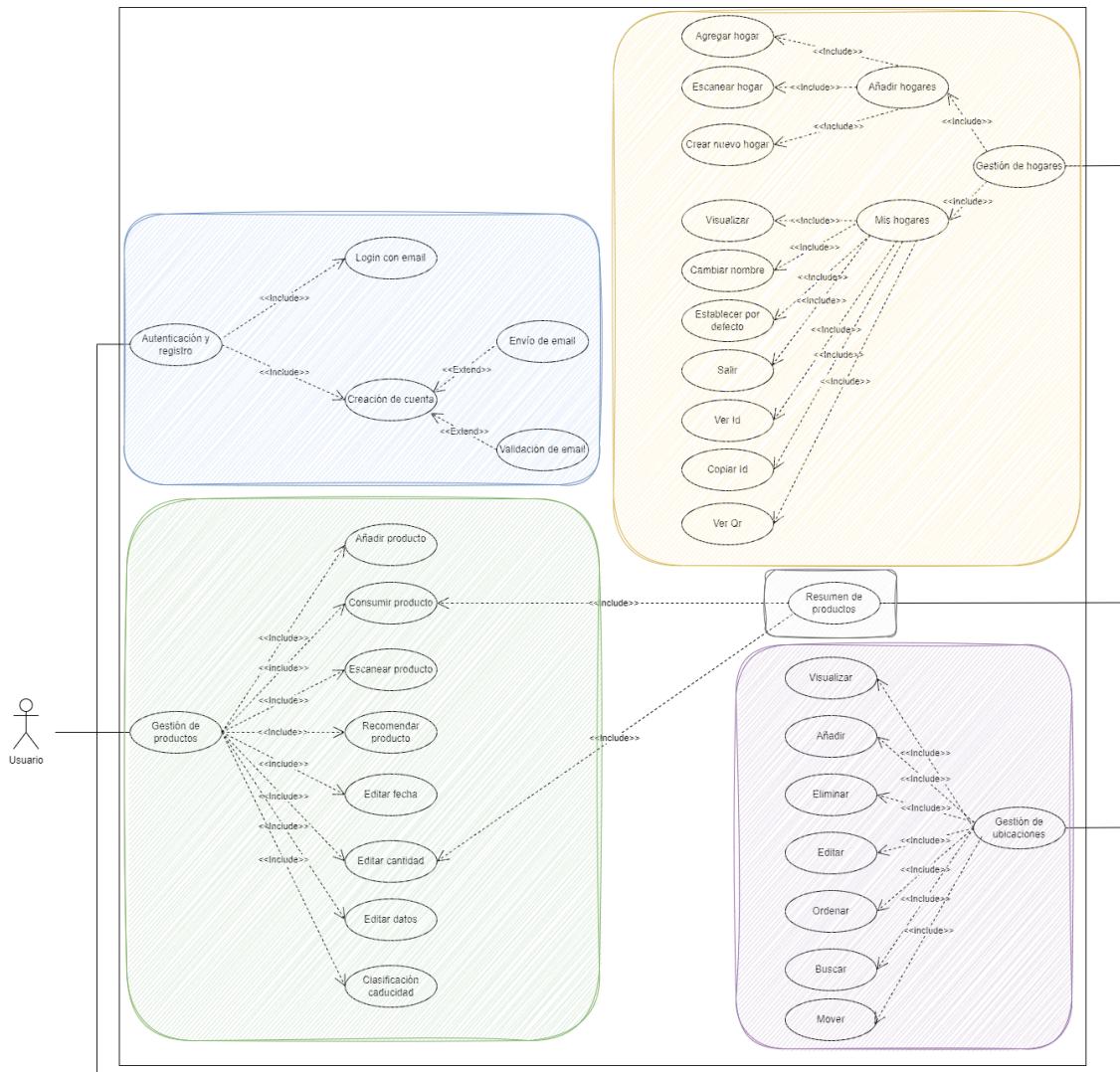


Figura B.1: Diagrama de casos de uso.

Casos de uso

CU-1	Autenticación y registro
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-1, RF-1.1, RF-1.2, RF-1.3, RF-1.4.
Descripción	Permite al usuario registrarse o loguearse.
Precondición	Existe conexión a firebase.
Acciones	<ol style="list-style-type: none"> 1. El usuario abre la aplicación. 2. Se muestra la pantalla de autenticación y registro.
Postcondición	El usuario visualiza la pantalla de autenticación.
Excepciones	Excepciones. <ul style="list-style-type: none"> ■ No hay conexión con Firebase.
Importancia	Alta.

Tabla B.1: CU-1 Autenticación y registro.

CU-2	Login con email
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-1.1.
Descripción	Permite al usuario loguearse con su email.
Precondición	El usuario tiene cuenta.
Acciones	<ol style="list-style-type: none">1. El usuario abre la aplicación.2. Se muestra la pantalla de autenticación y registro.3. El usuario introduce email correcto.4. El usuario introduce contraseña correcta.
Postcondición	El usuario accede a la aplicación con su perfil.
Excepciones	<p>Excepciones.</p> <ul style="list-style-type: none">■ Email inválido.■ Email no registrado.■ Email no validado.■ Contraseña incorrecta.
Importancia	Alta.

Tabla B.2: CU-2 Login con email.

CU-3	Creación de cuenta
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-1.2, RF-1.2.1, RF-1.2.2.
Descripción	Permite al usuario registrarse.
Precondición	El usuario no tiene cuenta.
Acciones	<ol style="list-style-type: none"> 1. El usuario abre la aplicación. 2. Se muestra la pantalla de autenticación y registro. 3. El usuario introduce email no asociado a una cuenta. 4. El usuario introduce una contraseña válida.
Postcondición	Se envía un email de verificación.
Excepciones	<p>Excepciones</p> <ul style="list-style-type: none"> ▪ Email inválido. ▪ Email ya registrado. ▪ Contraseña débil.
Importancia	Alta.

Tabla B.3: CU-3 Creación de cuenta.

CU-4	Envío de email
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-1.2.1.
Descripción	Envía un email con la validación de cuenta al usuario.
Precondición	El usuario ha se ha registrado.
Acciones	<ol style="list-style-type: none"> 1. El usuario abre su email. 2. El usuario tiene un correo a nombre de la aplicación.
Postcondición	Se valida el email del usuario.
Excepciones	<p>Excepciones</p> <ul style="list-style-type: none"> ■ Email se va a bandeja de Spam.
Importancia	Alta.

Tabla B.4: CU-4 Envío de email.

CU-5	Validación de email
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-1.2.2
Descripción	Asocia el vínculo del email a la cuenta del usuario.
Precondición	El usuario ha recibido el email de registro.
Acciones	<ol style="list-style-type: none"> 1. El usuario abre el mail de registro. 2. El usuario tiene un correo a nombre de la aplicación.
Postcondición	Se valida el email del usuario.
Excepciones	Excepciones
Importancia	Alta.

Tabla B.5: CU-5 Validación de email.

CU-6	Gestión de productos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2, RF-2.1, RF-2.2, RF-2.3, RF-2.4, RF-2.5, RF-2.6, RF-2.7, RF-2.8
Descripción	Permite al usuario gestionar sus productos.
Precondición	El usuario tiene un hogar asociado.
Acciones	<ol style="list-style-type: none"> 1. El usuario abre la aplicación. 2. El usuario tiene las opciones añadir o consumir productos.
Postcondición	El usuario puede elegir una opción.
Excepciones	El usuario no tiene un hogar asociado.
Importancia	Alta.

Tabla B.6: CU-6 Gestión de productos.

CU-7	Añadir productos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.1
Descripción	Permite al usuario añadir un producto al stock.
Precondición	El usuario tiene un hogar asociado.
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona en la opción añadir producto. 2. El usuario añade los datos del producto. 3. El usuario presiona en el botón añadir producto.
Postcondición	Se añade el producto.
Excepciones	El usuario no rellena los datos del producto.
Importancia	Alta.

Tabla B.7: CU-7 Añadir productos.

CU-8	Consumir productos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.2.
Descripción	Permite al usuario consumir un producto del stock.
Precondición	El usuario tiene productos en stock.
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona en la opción consumir producto. 2. El usuario busca el producto a consumir. 3. El usuario presiona en el botón consumir producto. 4. El usuario añade la nueva cantidad de unidades.
Postcondición	Se consume el producto.
Excepciones	El usuario introduce más stock del que había.
Importancia	Alta.

Tabla B.8: CU-8 Consumir productos.

CU-9	Escanear productos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.3
Descripción	Permite al usuario introducir un producto a través de la cámara.
Precondición	El usuario tiene un hogar asociado.
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona en la opción añadir o consumir producto. 2. El usuario presiona en el botón escanear con cámara. 3. El usuario escanea el código de barras. 4. Si el producto existe, se rellenan los datos, si no; se solicita rellenarlos.
Postcondición	Se escanea el producto.
Excepciones	No hay acceso a la cámara.
Importancia	Media.

Tabla B.9: CU-9 Escanear productos.

CU-10	Recomendar productos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.4
Descripción	Permite al usuario obtener recomendaciones de productos.
Precondición	El usuario ya tiene productos en stock.
Acciones	<ol style="list-style-type: none"> 1. El usuario presiona en la opción añadir producto. 2. El usuario presiona en buscar. 3. El usuario introduce uno de los siguientes: Nombre o marca o código de barras.
Postcondición	Se sugieren productos al usuario.
Excepciones	No hay productos en el historial.
Importancia	Baja.

Tabla B.10: CU-10 Recomendar productos.

CU-11	Editar fechas
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.5
Descripción	Permite editar una fecha de caducidad de un producto.
Precondición	El usuario tiene productos en stock.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú almacenamiento 2. El usuario selecciona un producto. 3. El usuario presiona en editar fecha. 4. El usuario introduce una nueva fecha.
Postcondición	Se cambia la fecha de caducidad del producto.
Excepciones	Error al cambiar la fecha.
Importancia	Alta.

Tabla B.11: CU-11 Editar fechas.

CU-12	Editar cantidades
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.6
Descripción	Permite editar la cantidad de unidades de un producto.
Precondición	El usuario tiene productos en stock.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú almacenamiento. 2. El usuario selecciona un producto. 3. El usuario presiona en editar cantidades. 4. El usuario introduce una nueva cantidad. 5. El usuario desliza un producto en el menú principal. 6. El usuario presiona en editar cantidades. 7. El usuario introduce una nueva cantidad.
Postcondición	Se cambian las unidades del producto.
Excepciones	Unidades negativas.
Importancia	Alta.

Tabla B.12: CU-12 Editar cantidades.

CU-13	Editar datos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.7.
Descripción	Permite editar los datos básicos de un producto.
Precondición	El usuario tiene productos en stock.
Acciones	<ol style="list-style-type: none">1. El usuario accede al menú almacenamiento.2. El usuario selecciona un producto.3. El usuario presiona en editar datos.4. El usuario introduce los nuevos datos.
Postcondición	Se cambian los datos del producto.
Excepciones	Error al cambiar los datos.
Importancia	Media.

Tabla B.13: CU-13 Editar datos.

CU-14	Cálculo y clasificación de caducidad
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-2.8.
Descripción	Permite visualizar la clasificación de caducidad de un producto.
Precondición	El usuario tiene productos en stock.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú principal. 2. El usuario visualiza los días restantes y su riesgo basado en color. 3. El usuario accede al menú almacenamiento. 4. El usuario visualiza los días restantes, el ícono y su riesgo basado en color.
Postcondición	Se visualizan la caducidad del producto.
Excepciones	Error al clasificar fecha.
Importancia	Alta.

Tabla B.14: CU-14 Cálculo y clasificación de caducidad.

CU-15	Gestión de hogares
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3, RF-3.1, RF-3.2.
Descripción	Permite al usuario gestionar sus hogares o añadir nuevos.
Precondición	El usuario tiene cuenta.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares.
Postcondición	El usuario visualiza las distintas opciones disponibles.
Excepciones	
Importancia	Alta.

Tabla B.15: CU-15 Gestión de hogares.

CU-16	Mis hogares
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1, RF-3.1.1, RF-3.1.2, RF-3.1.3, RF-3.1.4, RF-3.1.5, RF-3.1.6, RF-3.1.7.
Descripción	Permite al usuario gestionar sus hogares.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de mis hogares.
Postcondición	El usuario visualiza las opciones de sus hogares.
Excepciones	El usuario no tiene hogares asociados.
Importancia	Alta.

Tabla B.16: CU-16 Mis hogares.

CU-17	Visualizar hogares
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.1
Descripción	Permite al usuario visualizar sus hogares.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de mis hogares. 3. El usuario presiona en el combo de hogares
Postcondición	El usuario visualiza los nombres de sus hogares.
Excepciones	El usuario no tiene hogares asociados.
Importancia	Alta.

Tabla B.17: CU-17 Visualizar hogares.

CU-18	Cambiar nombre al hogar
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.2
Descripción	Permite al usuario cambiar el nombre a un hogar.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none">1. El usuario accede al menú gestión de hogares.2. El usuario accede al tab de mis hogares.3. El usuario presiona en el combo de hogares.4. El usuario elige un hogar.5. El usuario presiona en cambiar nombre.6. El usuario introduce un nuevo nombre.
Postcondición	Se cambia el nombre al hogar.
Excepciones	Nombre inválido.
Importancia	Media.

Tabla B.18: CU-18 Cambiar nombre al hogar.

CU-19	Establecer hogar por defecto
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.3.
Descripción	Permite al usuario establecer un hogar por defecto.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none">1. El usuario accede al menú gestión de hogares.2. El usuario accede al tab de mis hogares.3. El usuario presiona en el combo de hogares.4. El usuario elige un hogar.5. El usuario presiona en establecer por defecto.
Postcondición	Se establece como hogar por defecto.
Excepciones	Error al establecer por defecto.
Importancia	Alta.

Tabla B.19: CU-19 Establecer hogar por defecto.

CU-20	Salir de un hogar
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.4.
Descripción	Permite al usuario salir de un hogar.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de mis hogares. 3. El usuario presiona en el combo de hogares. 4. El usuario elige un hogar. 5. El usuario presiona en salir del hogar.
Postcondición	Se sale del hogar y se borra si no hay más usuarios en el hogar.
Excepciones	Error al salir del hogar.
Importancia	Alta.

Tabla B.20: CU-20 Salir de un hogar.

CU-21	Ver Id del hogar
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.5.
Descripción	Permite al usuario ver el Id asociado a un hogar.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none">1. El usuario accede al menú gestión de hogares.2. El usuario accede al tab de mis hogares.3. El usuario presiona en el combo de hogares.4. El usuario elige un hogar.
Postcondición	Se muestra un texto con el Id del hogar.
Excepciones	
Importancia	Alta.

Tabla B.21: CU-21 Ver Id del hogar.

CU-22	Copiar Id del hogar
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.6.
Descripción	Permite al usuario copiar el Id de un hogar.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de mis hogares. 3. El usuario presiona en el combo de hogares. 4. El usuario elige un hogar. 5. El usuario presiona en el texto del Id de hogar o en el código QR.
Postcondición	Se copia el Id del hogar al portapapeles.
Excepciones	Error al copiar al portapapeles.
Importancia	Baja.

Tabla B.22: CU-22 Copiar Id del hogar.

CU-23	Copiar Id del hogar
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.1.7.
Descripción	Permite al ver un código QR asociado al hogar.
Precondición	El usuario tiene hogares asociados.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de mis hogares. 3. El usuario presiona en el combo de hogares. 4. El usuario elige un hogar.
Postcondición	Se muestra el QR asociado al hogar.
Excepciones	Error al renderizar el QR.
Importancia	Baja.

Tabla B.23: CU-23 Ver QR del hogar.

CU-24	Añadir hogares
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.2, RF-3.2.1, RF-3.2.2, RF-3.2.3.
Descripción	Permite al usuario añadir nuevos hogares.
Precondición	El usuario está registrado y ha iniciado sesión.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de añadir hogares.
Postcondición	Se visualizan las opciones de añadir hogares.
Excepciones	
Importancia	Alta.

Tabla B.24: CU-24 Añadir hogares.

CU-25	Agregar hogar existente
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.2.1.
Descripción	Permite al usuario agregar un hogar existente.
Precondición	El usuario no está ya registrado en ese hogar.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de añadir hogares. 3. El usuario escribe el Id de un hogar existente. 4. El usuario presiona en añadir hogar existente.
Postcondición	Se añade al usuario a un hogar existente.
Excepciones	El usuario ya está en el hogar.
Importancia	Media.

Tabla B.25: CU-25 Agregar hogar existente.

CU-26	Escanear hogar existente
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.2.2.
Descripción	Permite al usuario escanear un hogar existente.
Precondición	El usuario tiene acceso a un código QR de hogar.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de añadir hogares. 3. El usuario presiona en escanear QR.
Postcondición	Se añade automáticamente el Id del hogar.
Excepciones	Error al leer QR.
Importancia	Media.

Tabla B.26: CU-26 Escanear hogar existente.

CU-27	Crear nuevo hogar
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-3.2.3.
Descripción	Permite al usuario crear un nuevo hogar.
Precondición	El usuario no está ya agregado a ese hogar.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de hogares. 2. El usuario accede al tab de añadir hogares. 3. El usuario escribe un nombre de hogar. 4. El preciona en crear hogar.
Postcondición	Se crea un hogar para el usuario.
Excepciones	Hogar duplicado.
Importancia	Alta.

Tabla B.27: CU-27 Crear nuevo hogar.

CU-28	Gestión de ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4, RF-4.1, RF-4.2, RF-4.3, RF-4.4, RF-4.5, RF-4.6, RF-4.7.
Descripción	Permite al usuario gestionar sus ubicaciones.
Precondición	El usuario tiene un hogar asociado.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones.
Postcondición	Se muestran las ubicaciones del usuario.
Excepciones	El usuario no tiene hogar asociado.
Importancia	Alta.

Tabla B.28: CU-28 Gestión de ubicaciones.

CU-29	Visualizar ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.1.
Descripción	Permite al usuario visualizar las ubicaciones creadas en el hogar.
Precondición	El hogar del usuario tiene ubicaciones asociadas.
Acciones	<ul style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones.
Postcondición	Se muestran las ubicaciones del usuario.
Excepciones	El usuario no tiene hogar asociado.
Importancia	Alta.

Tabla B.29: CU-29 Visualizar ubicaciones.

CU-30	Añadir ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.2.
Descripción	Permite al usuario añadir nuevas ubicaciones al hogar.
Precondición	El usuario tiene un hogar asociado.
Acciones	<ul style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones. 2. El usuario presiona el botón añadir ubicaciones. 3. El usuario introduce un nombre para la ubicación. 4. El usuario presiona en validar.
Postcondición	Se añade una nueva ubicación.
Excepciones	Ubicación no repetida.
Importancia	Alta.

Tabla B.30: CU-30 Añadir ubicaciones.

CU-31	Eliminar ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.3.
Descripción	Permite al usuario eliminar una ubicación junto con sus productos.
Precondición	El hogar del usuario tiene ubicaciones asociadas.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones. 2. El usuario presiona el botón editar. 3. El selecciona una o varias ubicaciones. 4. El usuario presiona en eliminar.
Postcondición	Se eliminan las ubicaciones seleccionadas junto con sus productos.
Excepciones	Error al eliminar.
Importancia	Alta.

Tabla B.31: CU-31 Eliminar ubicaciones.

CU-32	Editar ubicación
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.4.
Descripción	Permite al usuario editar el nombre de una ubicación.
Precondición	El hogar del usuario tiene ubicaciones asociadas.
Acciones	<ol style="list-style-type: none">1. El usuario accede al menú gestión de ubicaciones.2. El usuario presiona el botón editar.3. El selecciona una ubicación.4. El usuario presiona en editar nombre.5. El usuario introduce un nuevo nombre.
Postcondición	Se edita el nombre de la ubicación.
Excepciones	Nombre incorrecto o repetido.
Importancia	Baja.

Tabla B.32: CU-32 Editar ubicación.

CU-33	Ordenar ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.5.
Descripción	Permite al usuario ordenar sus ubicaciones junto con sus productos.
Precondición	El hogar del usuario tiene ubicaciones asociadas y productos en ellas.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones. 2. El usuario presiona un tipo de ordenación: Fecha o alfabética.
Postcondición	Se ordenan las ubicaciones por fecha o de forma alfabética.
Excepciones	Error al ordenar.
Importancia	Alta.

Tabla B.33: CU-33 Ordenar ubicaciones.

CU-34	Ordenar ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.6
Descripción	Permite al usuario buscar en sus ubicaciones.
Precondición	El hogar del usuario tiene ubicaciones asociadas y productos en ellas.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones. 2. El usuario escribe en el menú de búsqueda valores existentes.
Postcondición	Se muestran las ubicaciones o productos que coincidan con la búsqueda.
Excepciones	Error al buscar.
Importancia	Baja.

Tabla B.34: CU-34 Buscar en ubicaciones.

CU-35	Mover productos entre ubicaciones
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-4.7.
Descripción	Permite al usuario mover productos entre ubicaciones.
Precondición	El hogar del usuario tiene ubicaciones asociadas y productos en ellas.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú gestión de ubicaciones. 2. El usuario presiona el botón editar. 3. El usuario selecciona uno o varios productos. 4. El usuario presiona en el botón mover. 5. El usuario selecciona la ubicación de destino.
Postcondición	Se mueven los productos a la nueva ubicación.
Excepciones	Error al mover productos.
Importancia	Media.

Tabla B.35: CU-35 Mover productos entre ubicaciones.

CU-36	Resumen de productos
Versión	1.0.
Autor	Juan José Santos Cambra.
Requisitos asociados	RF-5.
Descripción	Permite al usuario visualizar un resumen de los productos.
Precondición	El hogar del usuario tiene ubicaciones asociadas y productos en ellas.
Acciones	<ol style="list-style-type: none">1. El usuario accede a la página principal de la aplicación.
Postcondición	Se muestra un resumen de productos.
Excepciones	No hay productos.
Importancia	Alta

Tabla B.36: CU-36 Resumen de productos.

B.5. Mockups de interfaces

En la fase de especificación de requisitos se han generado un conjunto de mockups de la aplicación usando la Web *Figma*, se muestran a continuación:

Login

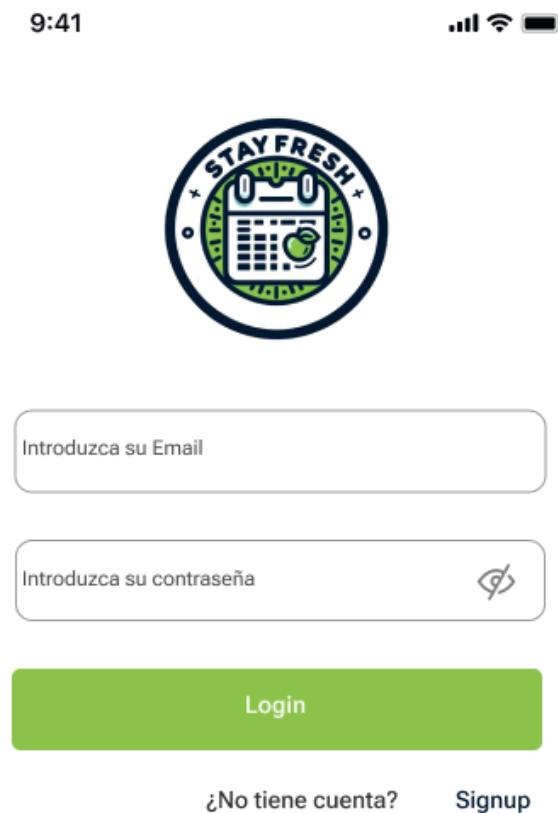


Figura B.2: Login de la APP.

Home



Figura B.3: Página principal.

Añadir producto

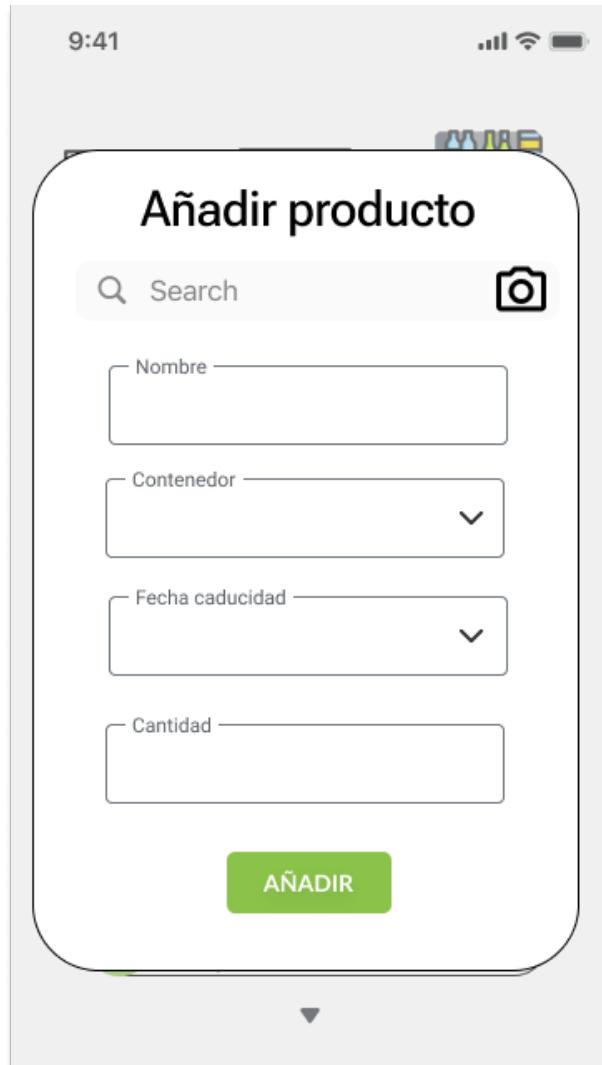


Figura B.4: Página añadir producto.

Consumir producto

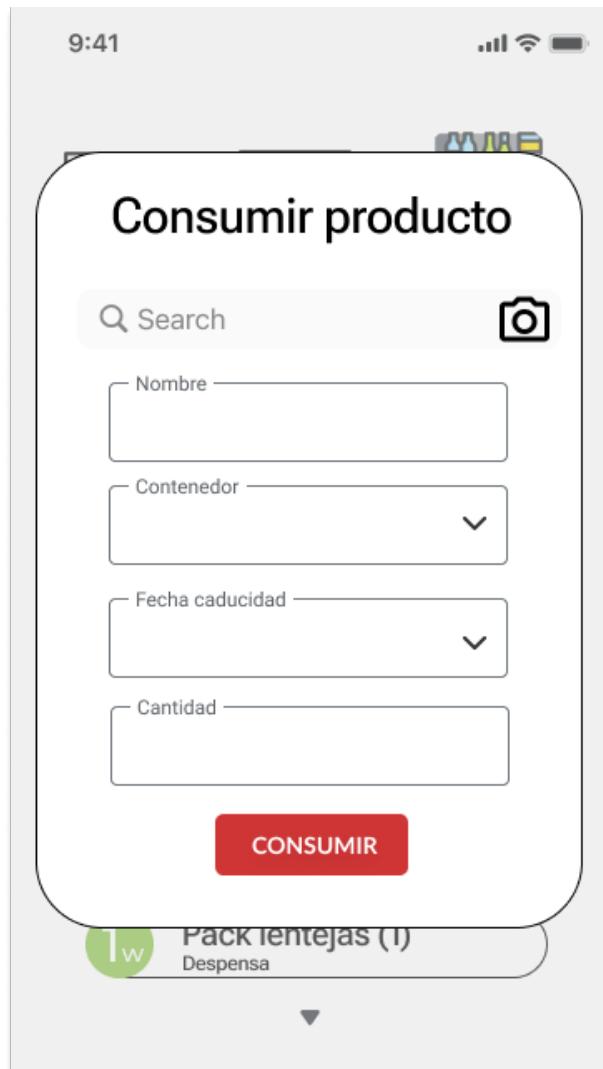


Figura B.5: Página consumir producto.

Gestión de hogares



Figura B.6: Página gestión de hogares.

Gestión de ubicaciones



Figura B.7: Página gestión de ubicaciones.

Opciones de ubicaciones

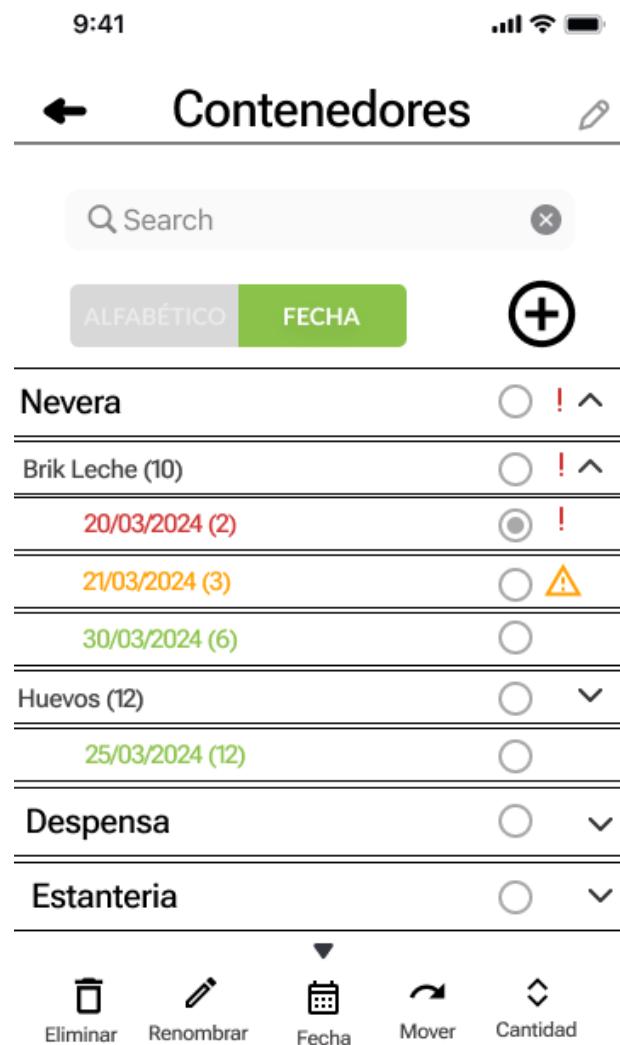


Figura B.8: Opciones en gestión de ubicaciones.

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se detalla cómo se han resuelto los objetivos y especificaciones previamente presentados. Se describe el diseño de los datos, el diseño procedural de la aplicación y el diseño arquitectónico.

C.2. Diseño de datos

La aplicación dispone de las siguientes entidades:

- **Usuarios (Users)**: Tienen un Id y un conjunto de hogares asociados los cuales tienen un nombre diferente entre usuarios.
- **Hogares (Homes)**: Tienen un Id y un conjunto de ubicaciones.
- **Ubicaciones (Storages)**: Tienen un Id, un nombre y un conjunto de productos.
- **Productos (Products)**: Tienen un Id, una fecha de inserción, un código de barras, una marca, una fecha de caducidad, un nombre y una cantidad.

A la hora de diseñar el modelo de datos en *Firestore* se han tenido en cuenta las siguientes propiedades de esta base de datos, algunas de estas características son propias al motor de *Firestore* en sí mismo y otras debido al hecho que es una base de datos de documentos, NO-SQL.

- **Búsquedas en colecciones:** *Firestore* no implementa queries entre distintas colecciones, solo se pueden realizar queries a documentos de una misma colección o de múltiples colecciones las cuales comparten el mismo Id. Carece del concepto *join* de las bases de datos relacionales.
- **Listado de colecciones:** *Firestore* requiere de un admin API para poder recuperar el número de colecciones que tiene un documento así como sus nombres.
- **Tamaño máximo de documento:** *Firestore* tiene un tamaño máximo de documento de 1 MiB, pero carece de límite en lo que a cantidad de documentos se refiere.
- **Indexación:** En caso de necesitar realizar queries que usen un filtrado por más de un campo, *Firestore* requiere la creación de un índice con los campos por los que se quiera filtrar.
- **Límite diario:** Al utilizar la versión gratuita de *Firestore*, nos encontramos las siguientes restricciones:
 - Datos almacenados 1 GB
 - Lecturas de documentos: 50.000 por día
 - Escrituras de documentos: 20.000 por día
 - Eliminaciones de documentos: 20.000 por día
 - Transferencia de datos salientes: 10 GiB por mes

Por lo tanto, para cumplir con las restricciones de *Firebase* tanto a nivel estructural como a nivel cuantitativo, se ha realizado el siguiente diseño de los documentos de *Firestore* con el afán de reducir el número de operaciones necesarias:



Figura C.1: Diagrama de Firebase

Podemos encontrar en el primer nivel de documentos las siguientes rutas:

- **products:** Tener los productos en el primer nivel nos permite realizar queries por los campos más básicos como son: hogar, fecha de caducidad, consumido y fecha de inserción. Estas son claves básicas en las búsquedas. Encontramos dentro del FK del almacenamiento evitando tener que consultar otro documento más lo cual aumentaría el número de lecturas.
- **homes:** Contiene dos arrays de objetos destinados a evitar la búsqueda en más documentos intermedios y por lo tanto desbordar el límite gratuito de *Firebase*. En storages podemos encontrar una definición de los almacenamientos que tiene el hogar. En users se definen los usuarios que tiene el hogar. Esto nos permite realizar comprobaciones de usuarios sin tener que cruzar contra otros documentos.
- **users:** Dentro encontramos un array de objetos del tipo userHomes donde se encuentra el FK de cada hogar, el nombre que el usuario le ha dado al hogar, un booleano para identificar si está establecido como hogar por defecto y otras configuraciones.

Como hemos podido observar, la base de datos se encuentra desnormalizada por los motivos mencionados anteriormente, principalmente:

- No existencia de joins.
- Límite de lecturas diarias en la versión gratuita: Lo que se ha buscado con el diseño es reducir el número de lecturas teniendo que aumentar el número de escrituras al realizar operaciones de edición de datos. Al ser las escrituras menos frecuentes que las lecturas, nos aseguramos nos sobrepasar los límites establecidos.
- Falta de iteración en colecciones y sus nombres.

C.3. Diseño procedural

En esta sección se detallan las secuencias de comunicación entre los distintos componentes y los servicios que se encargan de la navegación, comunicación con *Firebase* y gestionar las traducciones. *React* incluye una serie de funciones especiales denominadas *Hooks* las cuales nos permiten manejar tanto el ciclo de vida como el contexto de un componente. Por convención, todos los *Hooks* comienzan por el prefijo *use*.

Navegación

Para la navegación se utiliza un componente StackNavigator para no tener que renderizar de nuevo cada componenete ya visitado.

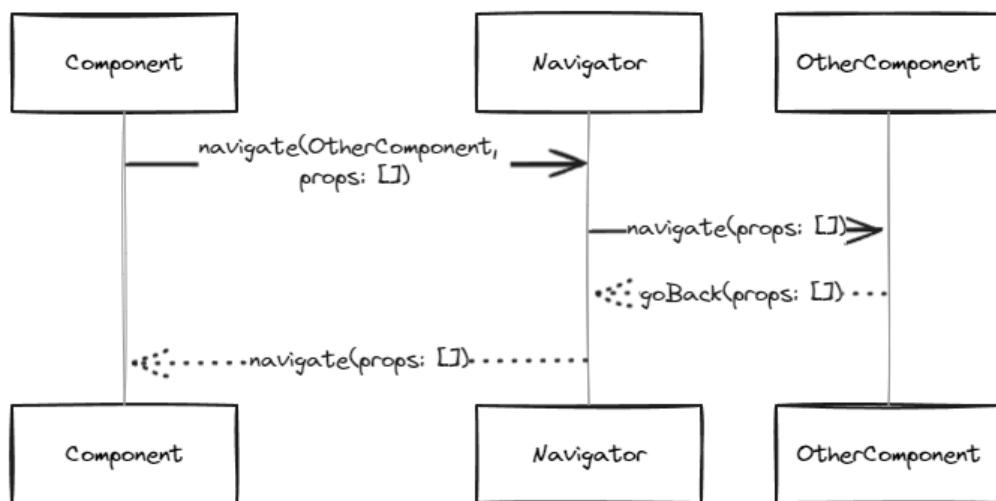


Figura C.2: Navegacion

Componente login

El componente login es el encargado de gestionar el inicio de sesión del usuario.

Proceso de login:

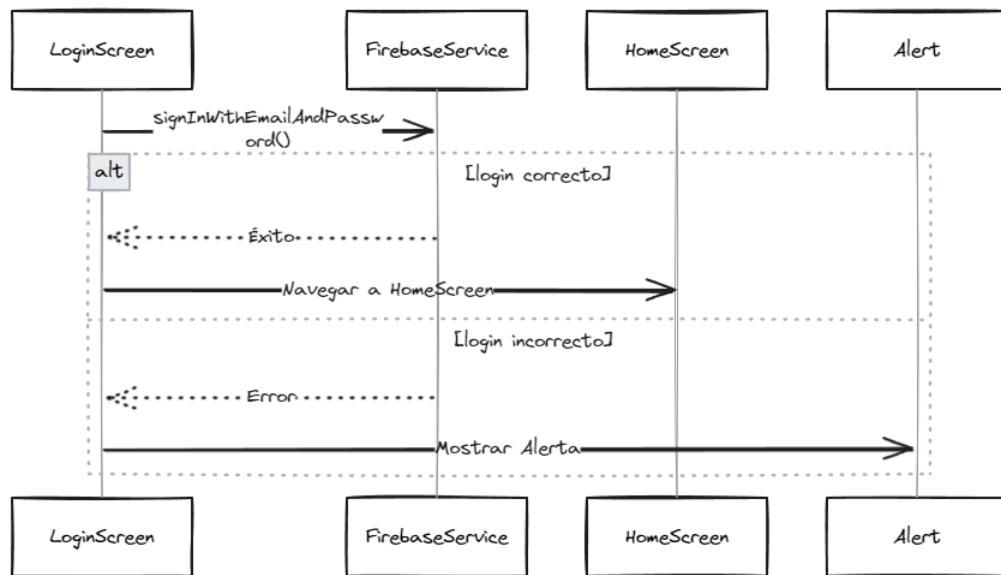


Figura C.3: Diagrama componente login

Componente home

El componente home es el principal en la aplicación desde el cuál se cargan los datos iniciales, se crea su perfil si no tiene y se navega al resto.

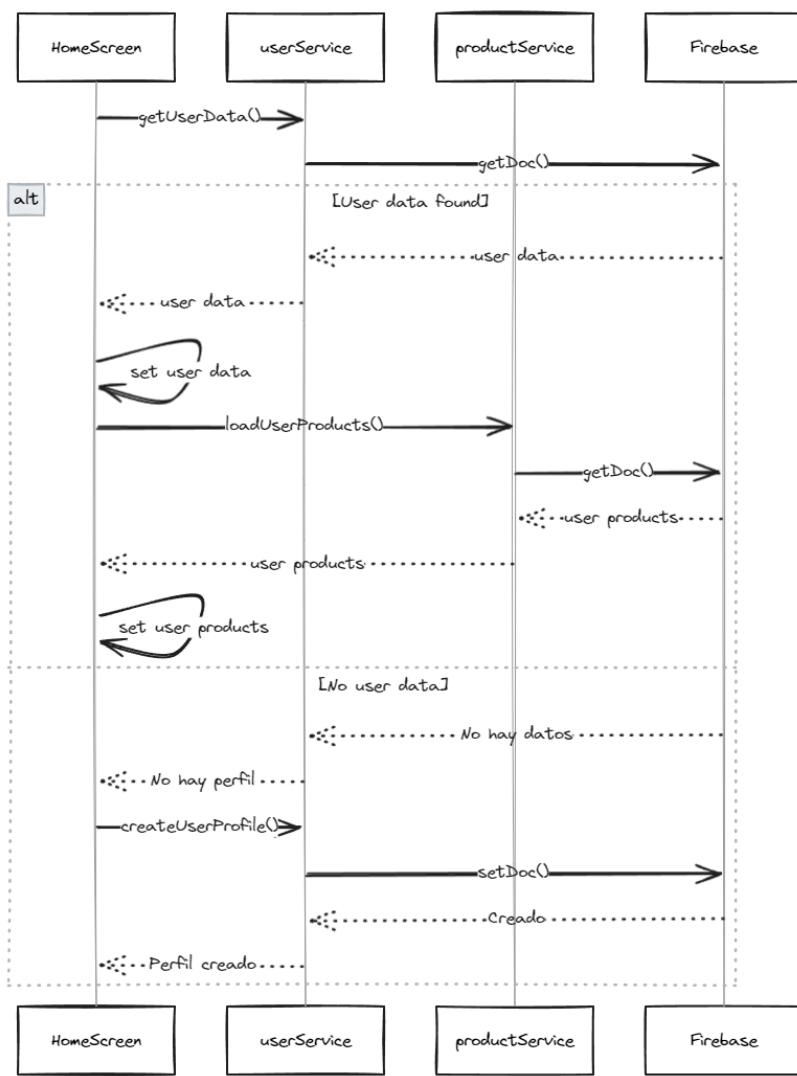


Figura C.4: Diagrama componente home

Hook useTranslate

El hook useTranslate es transversal a todos los demás componentes, es por ello que se representa un componente general.

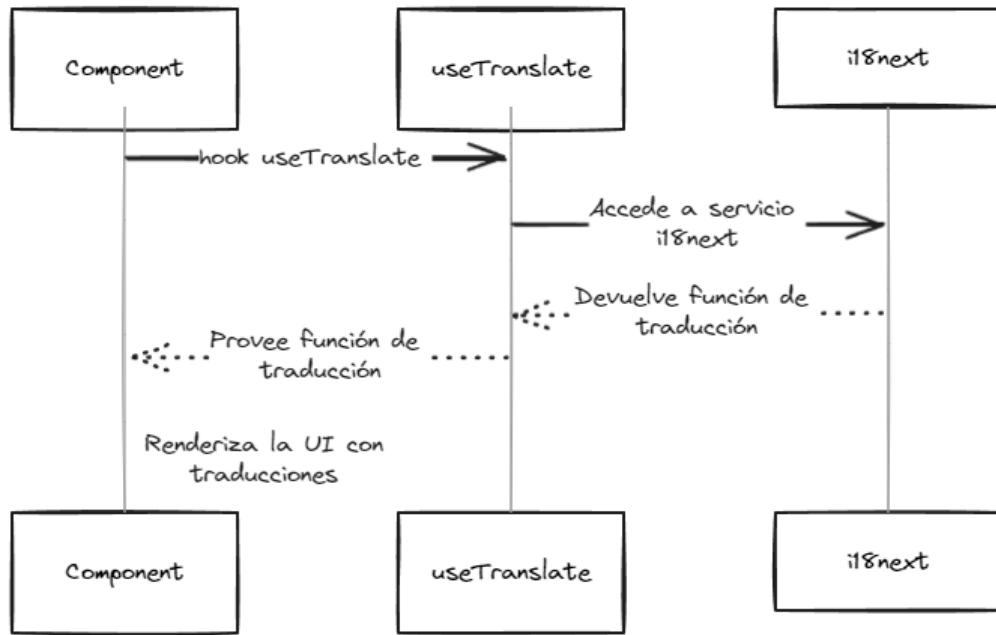


Figura C.5: Diagrama useTranslate

Componente addProduct

El componente addProduct se encarga de añadir productos al hogar del usuario

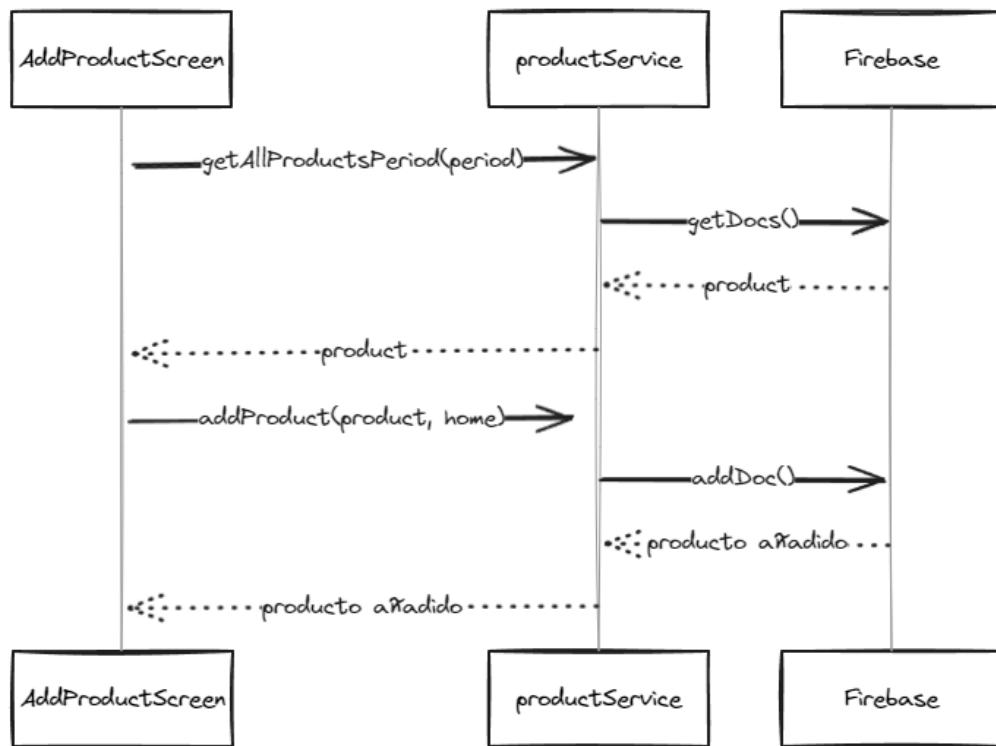


Figura C.6: Diagrama componente addProduct

Componente consumeProduct

El componente consumeProduct se encarga de consumir productos del hogar del usuario

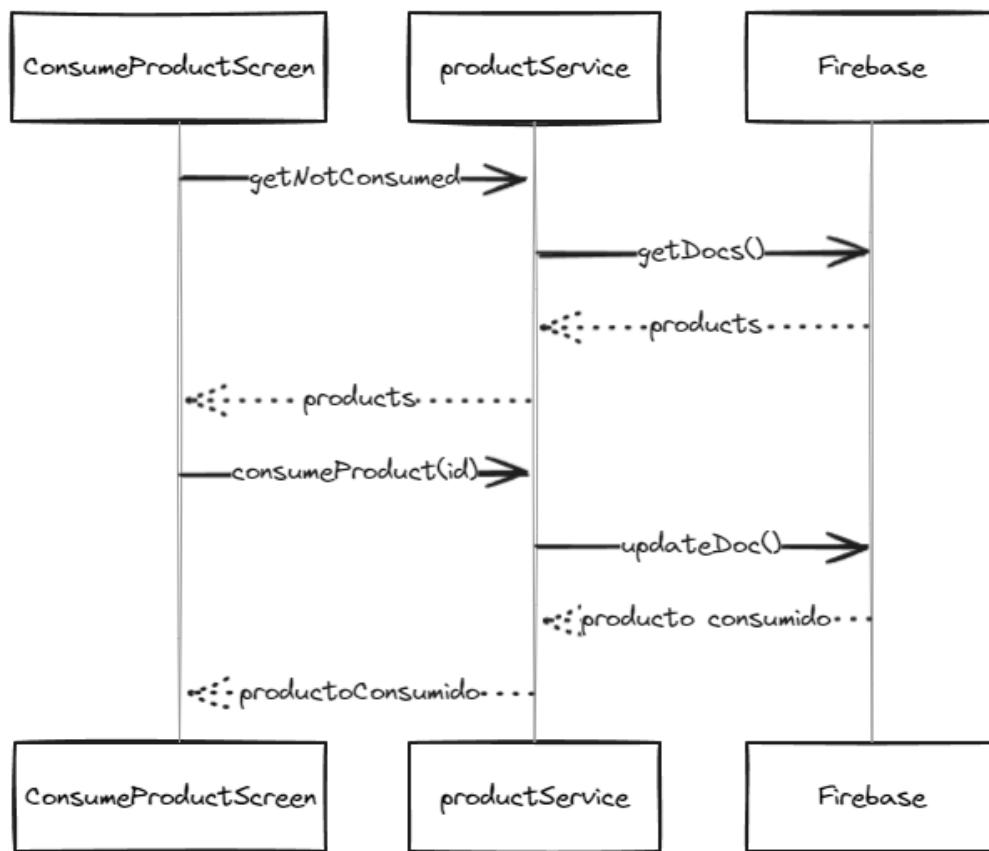


Figura C.7: Diagrama componente consumeProduct

Componente camera

El componente camera realiza las lecturas tanto de QR como de código de barras y devuelve el dato al componente anterior.

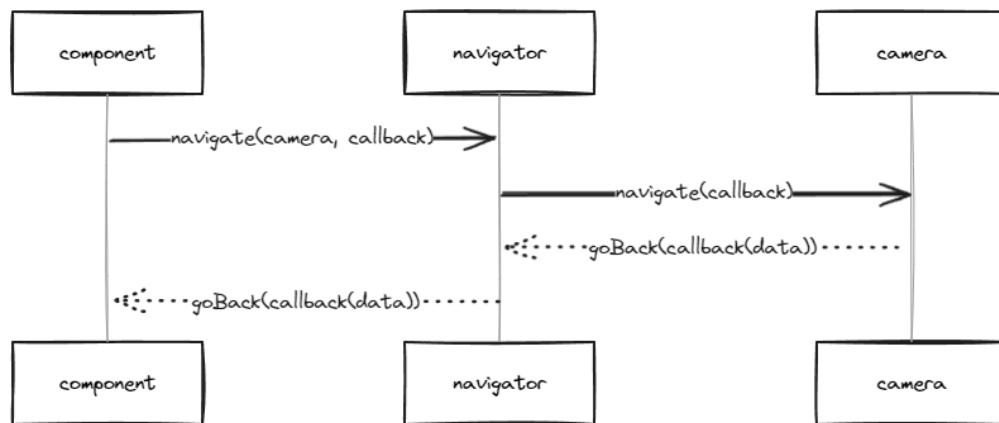


Figura C.8: Diagrama componente camera

Componente homeManagement

El componente homeManagement se encarga de la gestión de hogares del usuario, editar, añadir, eliminar...

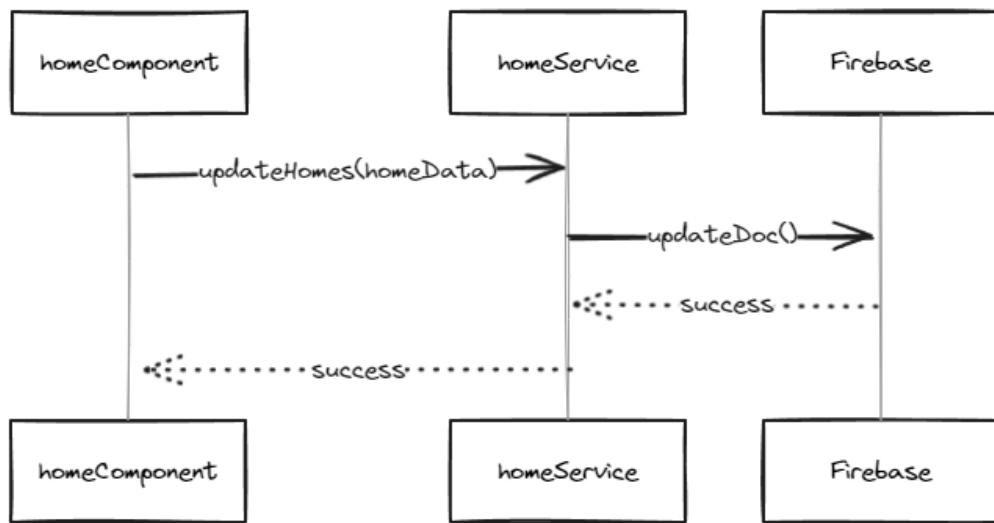


Figura C.9: Diagrama componente homeManagement

Componente storage

El componente homeManagement se encarga de la visualización de los productos ordenador por storage

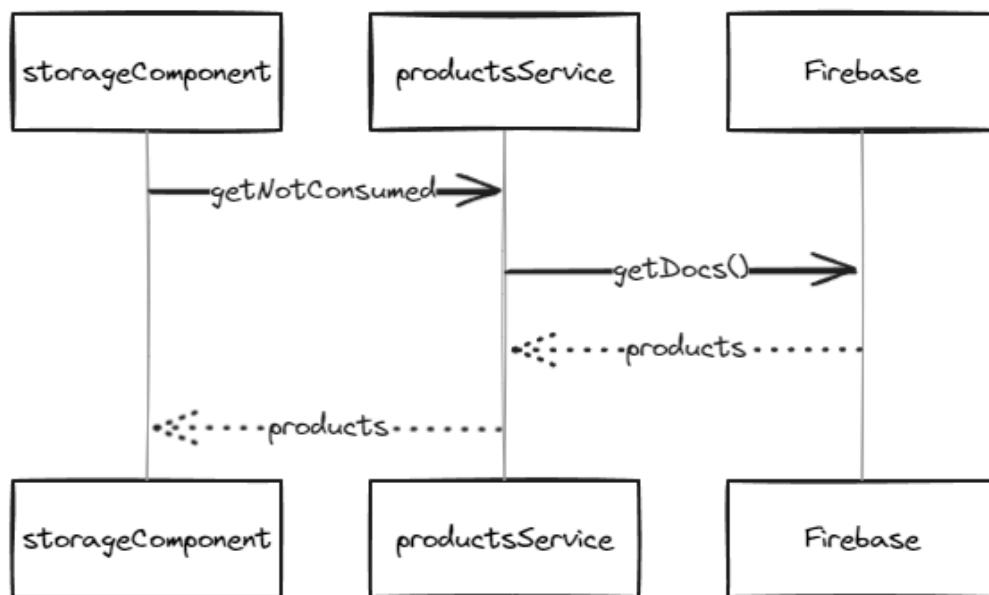


Figura C.10: Diagrama componente storage

C.4. Diseño arquitectónico

La solución está basada en *React*, por lo tanto, las decisiones de diseño y los patrones aplicados están estrechamente relacionados con esta tecnología.

Arquitectura general

La aplicación utiliza los servicios de backend de *Firebase*, tanto para la autenticación como el almacenamiento de datos, a continuación podemos encontrar el diagrama general de la arquitectura:

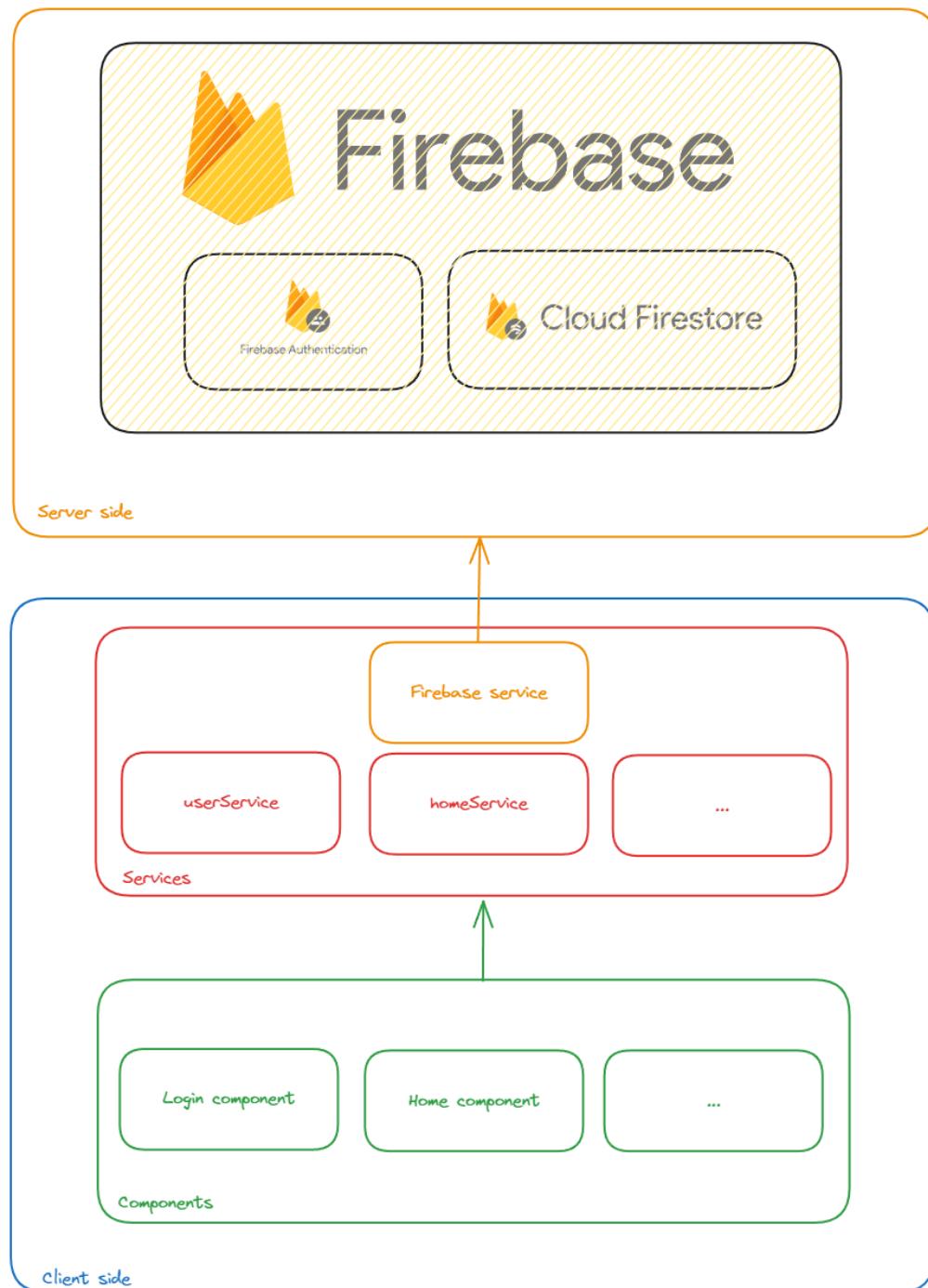


Figura C.11: Arquitectura general

Patrones

A continuación se detallan los patrones usados:

Component Composition

Este patrón de diseño implica utilizar múltiples sub-componentes para crear componentes más complejos. Esto permite que el código sea más modular y escalable así como un alto grado de personalización que se consigue a través del uso de una propiedad de *React* denominada *Props* la cual nos permite pasar o recuperar datos entre distintos niveles.

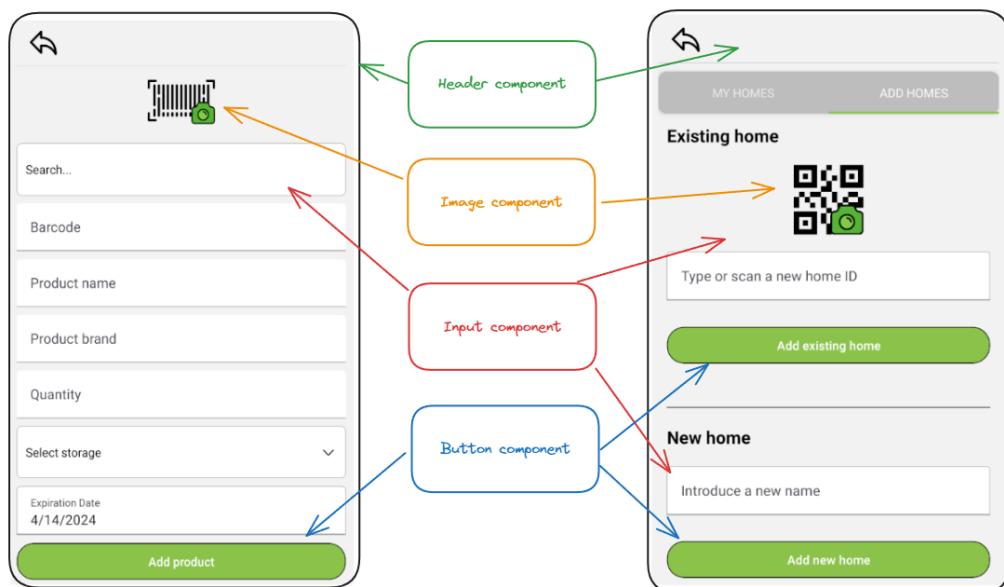


Figura C.12: Component composition

Container/Presentational Components

Separa la lógica de la vista. La idea detrás de este patrón es tener dos tipos de componentes: contenedor/servicio y presentación. Los componentes contenedor y los servicios son responsables de manejar la lógica y la obtención de datos, mientras que los de presentación son responsables de renderizar la interfaz de usuario. Esto lo podemos ver representado en la solución en aplicaciones como la siguiente:

- **storageComponent**: Este sería nuestro componente de presentación el cuál posee otros subcomponentes de UI.
- **productsService**: Este sería nuestro servicio encargado de la obtención y la lógica de datos.

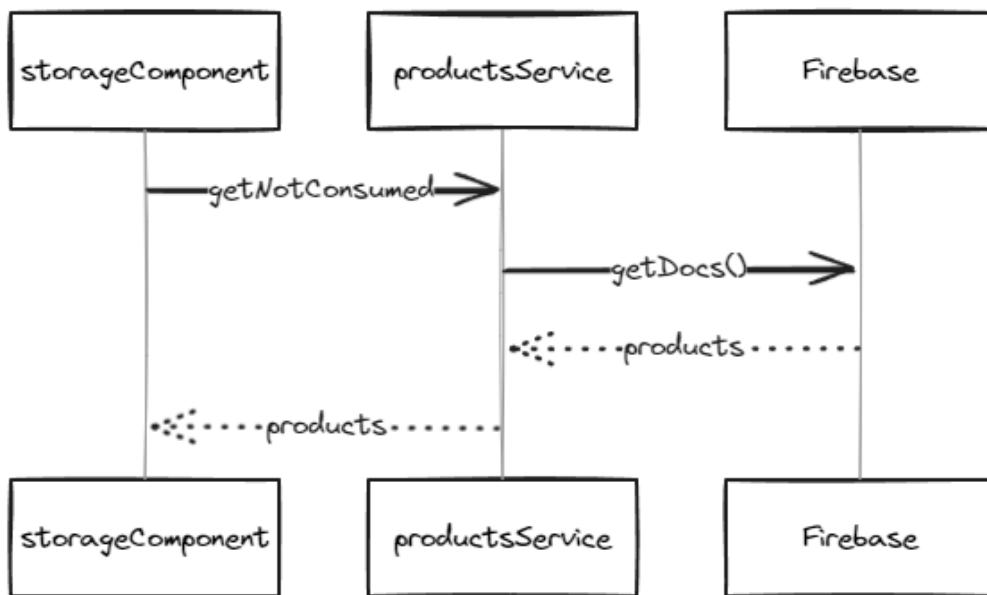


Figura C.13: Diagrama componente storage

Atomic Design

Es un patrón de diseño que implica descomponer la aplicación en componentes más pequeños y reutilizables. Este patrón implica la creación de componentes a nivel atómico, como átomos, moléculas y organismos. Los átomos son los componentes más pequeños, como botones y campos de entrada, mientras que las moléculas son grupos de átomos, como un formulario. Los organismos son grupos de moléculas, como una barra de navegación. Este patrón ayuda a hacer que el código sea más modular y reutilizable. Esto lo podemos ejemplificar con los componentes: Input, button, image... Cada uno de ellos son atómicos pero al juntarse crean lo denominado como molécula que da vida a la componente final screen que sería el organismo.

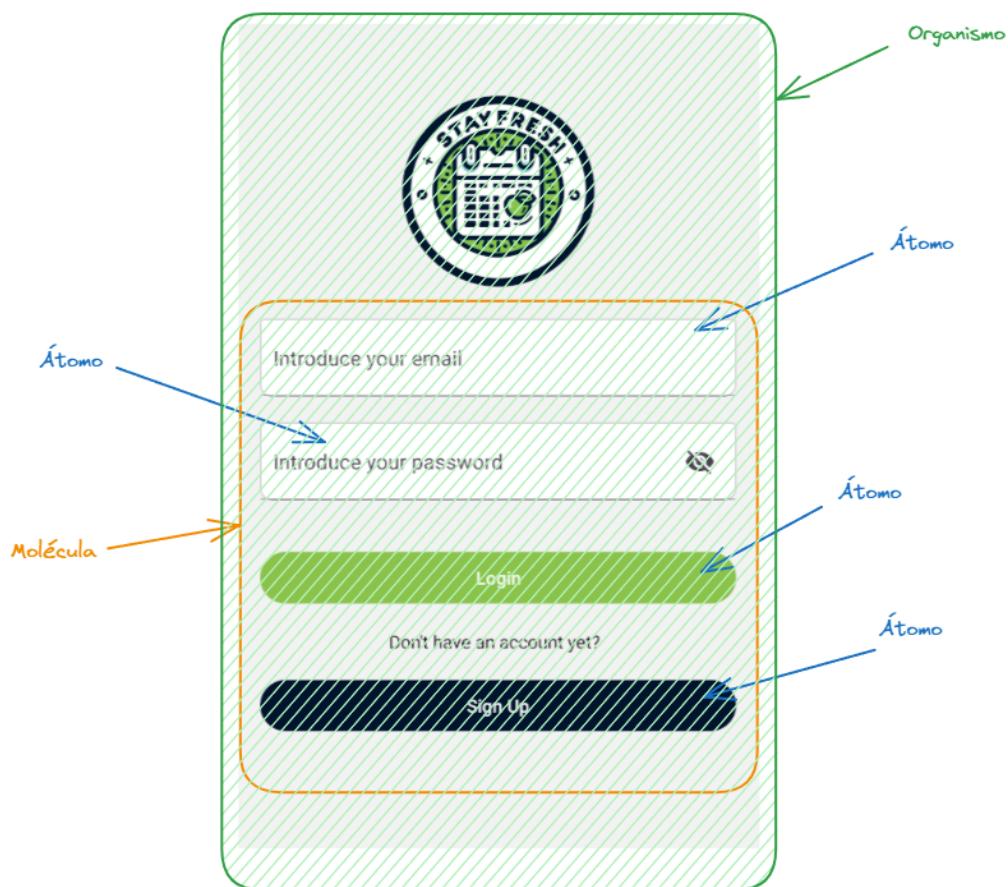


Figura C.14: Atomic design

Dependencias

En el desarrollo de la aplicación se ha utilizado el *Node Package Manager* también conocido como *NPM* como gestor de dependencias o paquetes. A continuación se detallan todos los paquetes usados en la aplicación.

- **react-native-async-storage/async-storage:** Servicio. Se encarga de almacenar en la memoria local distintos datos. En el caso de esta aplicación, se almacenan estados como el inicio de sesión (Auth).
- **react-native-community/datetimepicker:** Componente. Muestra un calendario diferente dependiendo del sistema operativo y devuelve el valor seleccionado en una función callBack.
- **expo:** En un marco de trabajo que proporciona un conjunto de herramientas integradas que permiten simplificar el desarrollo y configuraciones.
- **expo-barcode-scanner:** Componente. Se encarga de la gestión de lecturas de códigos de barras.
- **expo-blur:** Componente. Se aplica en los modales de carga para difuminar el fondo de la aplicación y dar una sensación de estado de carga.
- **expo-camera:** Componente. Se encarga de mostrar la cámara del usuario una vez que se han conseguido los permisos.
- **expo-clipboard:** Servicio. Permite acceder al portapapeles del dispositivo y realizar acciones como copiar o pega.
- **expo-status-bar:** Componente. Permite la manipulación de la barra de estado del dispositivo del usuario.
- **firebase:** Servicio. Se encarga de la comunicación con la base de datos y autenticación de *Firebase*.
- **react:** Paquete base de la librería de React. Expone todas las funciones necesarias: useState, useEffect...
- **react-i18next:** Servicio. Gestiona el estado de las traducciones y expone el hook useTranslate.
- **react-native:** Paquete base de React para desarrollo de aplicaciones móviles. Proporciona distintos elementos o componentes adecuados al sistema operativo como: Alert, Image, LogBox...

- **react-native-dropdown-picker**: Componente. Funciona como un *combo box* el cuál nos permite configurarlo mediante *Props*.
- **react-native-nested-listview**: Componente. Representa una lista anidada que es configurable mediante *Props*
- **react-native-pager-view**: Componente. Proporciona la capacidad de desplazarse entre componente deslizando.
- **react-native-paper**: Componente. Es una librería de componentes personalizados que pueden ser adecuados mediante al uso de *Props*.
- **react-native-svg**: Componente. Permite el dibujado de elementos con formato svg.
- **react-native-qrcode-svg**: Componente. Permite el renderizado de códigos QR.
- **react-native-screens**: Servicio. Permite distintos tipos de navegaciones entre componentes.
- **react-native-swipe-list-view**: Componente. Muestra una lista con capacidades de deslizar cada elemento.
- **react-native-tab-view**: Componente. Permite la navegación tipo tabs. Configurable por *Props*.
- **react-native-toast-message**: Componente. Muestra mensajes tipo toast. Configurable por *Props*.
- **react-native-toggle-element**: Componente. Muestra un elemento toggle. Configurable por *Props*.
- **react-native-uuid**: Servicio. Genera códigos UUID randomizados.
- **react-native-vector-icons**: Componente. Permite el acceso a varias librerías de iconos.
- **use-debounce**: Servicio. Expone varios hook para el manejo del debounce de la entrada del usuario.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo se detalla la documentación técnica relacionada con el entorno, programación y despliegue de la aplicación. Se verán detalles técnicos como la estructura de directorios, dependencias del entorno y compilación.

D.2. Estructura de directorios

- **/**: Carpeta raíz donde estarán contenidas todas las demás carpetas. Incluye los puntos de entrada a la aplicación y archivos de configuración.
- **/assets**: Almacena los archivos de imagen estáticos para la generación del instalador, logo y splash screen de la aplicación en Android.
- **/node_modules**: Aquí se encuentran todas las dependencias necesarias para el proyecto. Esta carpeta no la encontramos en *Github* ya que está añadida al *.gitignore* por contener muchos archivos. Sin embargo, se genera automáticamente al realizar ciertos comandos que veremos más adelante.
- **/src**: Es la carpeta contenedora del código fuente principal de la aplicación.

- **/src/assets:** Almacena los archivos de imagen estáticos necesarios en los distintos componentes de la aplicación.
- **/src/components:** En esta carpeta se encuentran todos los componentes que pueden ser reutilizados siguiendo el modelo atómico.
- **/src/config:** Aquí podemos encontrar la configuración de la aplicación y de la API de *Firebase*. Las API Keys no se encuentran en la carpeta por ser secretas.
- **/src/constants:** Contiene los archivos que definen las constantes comunes a toda la aplicación.
- **/src/helpers:** Contiene archivos que implementan funciones de asistencia, comprobaciones de datos...
- **/src/locales:** Contiene los archivos *.json* relativos a las traducciones de la aplicación.
- **/src/mockup:** Contiene mockup de ejemplo de datos para validar componentes sin utilizar *Firebase*.
- **/src/screens:** Componentes completos compuestos que representan una pantalla de la aplicación.
- **/src/services:** Servicios que manejan la lógica de la aplicación.
- **__test__:** Son las carpetas que contienen los test asociados a la carpeta padre.

D.3. Manual del programador

Esta sección tiene como objetivo explicar cómo preparar el entorno de la aplicación, descargar el código fuente y compilar para poder realizar modificaciones sobre la aplicación.

Entorno

Las dependencias mínimas para poder trabajar con el proyecto son las siguientes:

- Git.
- Node.

- Npm.
- Expo.
- Firebase.

Además, se recomienda también tener las siguientes dependencias instaladas para facilitar el desarrollo:

- Visual Studio Code.
- Emulador.

Git

Para poder trabajar sobre el proyecto es necesario poder clonar este y sincronizar los cambios que se realicen. El programa *Git* nos permitirá realizar todas estas acciones. Para realizar la descarga iremos a la URL: [15]

Node v20.11.1

Es un entorno de ejecución de *Javascript* construido sobre el motor V8 de *Google Chrome*. Permite la ejecución de código *Javascript* fuera del navegador. La versión necesaria la podemos encontrar en la siguiente ruta [8]

Npm

También conocido como *Node Package manager* es el manejador de paquetes de *Node* que nos permitirá instalar y compartir paquetes. Será la herramienta que utilicemos para descargar las dependencias de la aplicación. Su instalación está incluida dentro de *Node*. Podremos invocar a *Npm* mediante el comando *npm* desde la terminal.

Expo

Es una plataforma de código abierto para la creación universal de aplicaciones nativas de *Android* e *iOS* y para la web utilizando *Javascript* y *React*. Una vez tengamos *Node* instalado, podremos crear aplicaciones de *Expo* mediante el comando *npx create-expo-app*.

Firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles lanzada en 2011 y adquirida por Google en 2014.1. *Firebase* nos proporcionará los servicios necesarios para la autenticación y almacenamiento de datos. Los servicios que utilizaremos de *Firebase* son:

- *Firebase* Auth.
- *Firebase* Cloud Firestore.

La web es accesible en el siguiente Link [9]

No se dará acceso a la base de datos de *Firebase* ya que está ligado directamente a una cuenta personal de Google, pero a continuación se explica cómo replicar el entorno para un posible *Fork* del proyecto. Primero será necesario crear un proyecto:



Figura D.1: Creación de proyecto en Firebase

Una vez tengamos el proyecto creado, es necesario habilitar la autenticación mediante email, para ello ir a *Authentication*, métodos de acceso, agregar proveedor nuevo, correo electrónico.

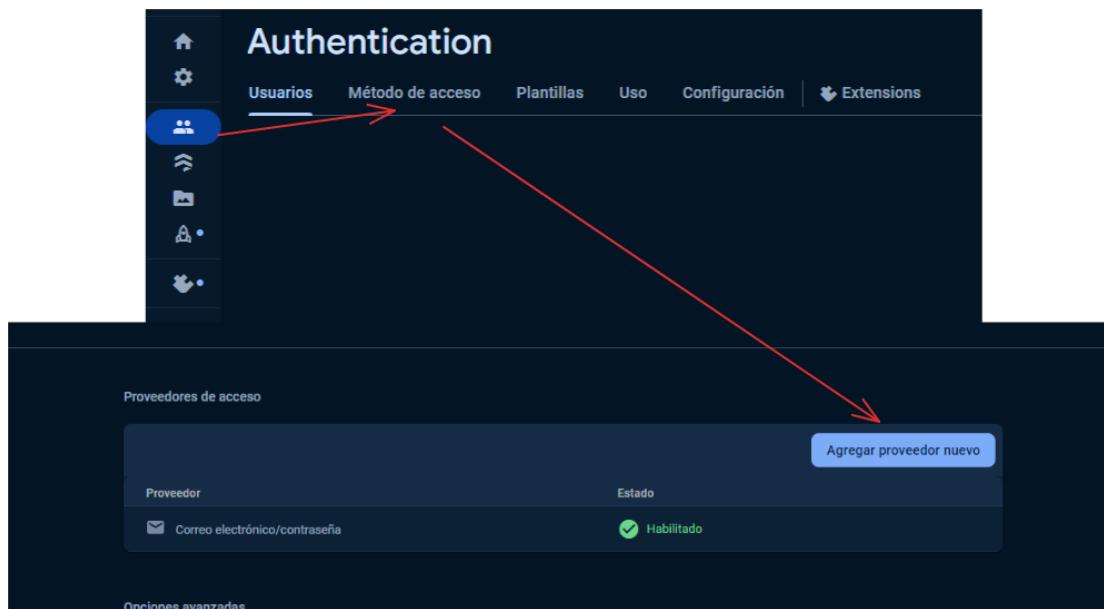


Figura D.2: Habilitar autenticación Firebase

Una vez preparada la autenticación, será necesario habilitar el *Firestore Database* y crear la estructura de base de datos vista en el capítulo de diseño. En el apartado índices añadiremos dos índices para poder realizar correctamente las queries de la aplicación:

ID de la colección	Campos indexados	Alcance de la consulta	Estado
products	home Ascendente, addedDate Ascendente, __name__ Ascendente	Colección	Habilitado
products	consumed Ascendente, home Ascendente, expirationDate Ascendente, __name__ Ascendente	Colección	Habilitado

Figura D.3: Indices de Firebase

Finalmente iremos a configuración del proyecto, configuración del sdk y copiaremos nuestra API Key para poder utilizarla en el proyecto.

Visual Studio Code

Visual Studio Code es un editor de código desarrollado por Microsoft para Windows, Linux, macOS y Web. Integra un marketplace de extensiones que nos son útiles durante el desarrollo, las extensiones recomendadas para trabajar con este proyecto son:

- React Native Tools: Esta extensión nos permitirá lanzar nuestra app de react en el emulador, depurarla y otras diversas opciones.
- Simple react snippets: Esta extensión nos permite mejorar nuestra productividad al contar con diversos snippets para abbreviar la generación del código.
- Prettier: Es un formateador automático de código que nos permite mantener un estándar unificado en toda la aplicación.

Para obtener Visual Studio Code lo podemos hacer a través del siguiente link: [12]

Emulación

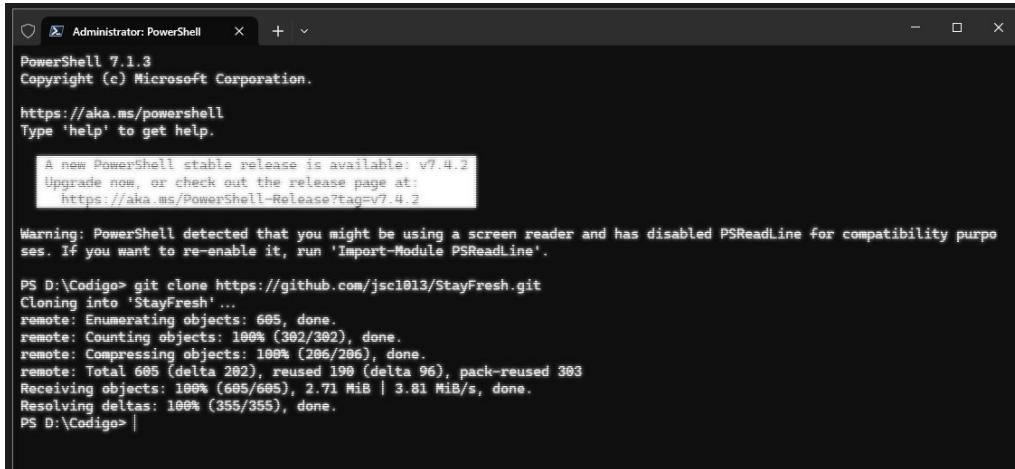
Existen dos opciones principales a la hora de emular la aplicación:

1. Emulator virtual: Podemos instalar un emulador virtual de un dispositivo android a través del *AVD* (Android Virtual Device) de Android Studio. Podemos encontrar el paso a paso detallado [11].
2. App de Expo: Expo proporciona una aplicación disponible en la *Play Store* que nos permite escanear un código QR en nuestro equipo para sincronizar la APP y poder ver en tiempo real los cambios. Podemos encontrar la aplicación en el siguiente enlace [5].

Clonación

Para realizar este paso será necesario tener instalado git. Teniendo instalado git en nuestra máquina, para clonar el repositorio y poder trabajar sobre él tendremos que ejecutar el siguiente comando:

```
git clone https://github.com/jsc1013/StayFresh.git
```



The screenshot shows a Windows PowerShell window titled 'Administrator: PowerShell'. The window displays the output of a 'git clone' command. The output includes a warning about PSReadLine being disabled for compatibility purposes, followed by the progress of cloning a repository named 'StayFresh' from the URL 'https://github.com/jsc1013/StayFresh.git'. The process shows objects being enumerated, counted, compressed, and received, along with a final message about resolving deltas.

```
A new PowerShell stable release is available: v7.4.2
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.4.2

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

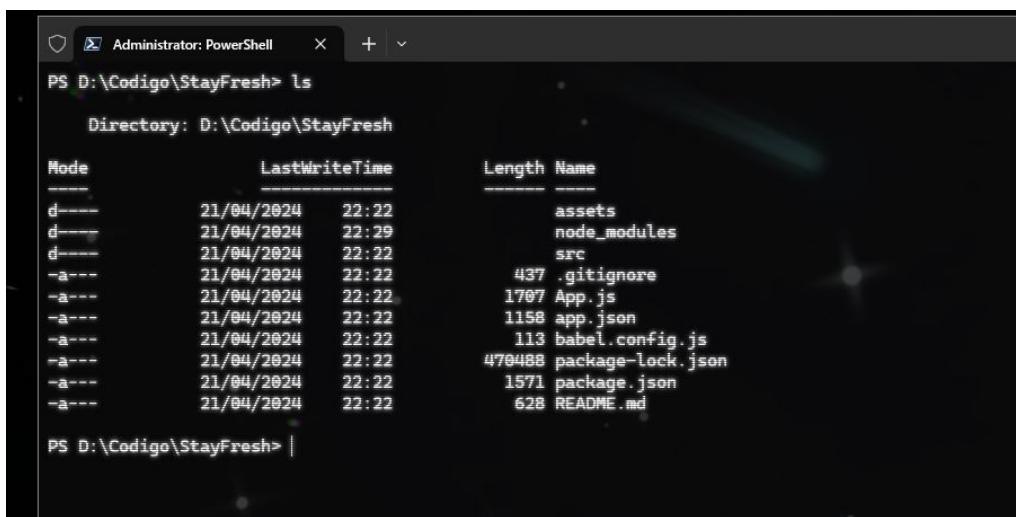
PS D:\Codigo> git clone https://github.com/jsc1013/StayFresh.git
Cloning into 'StayFresh'...
remote: Enumerating objects: 605, done.
remote: Counting objects: 100% (392/392), done.
remote: Compressing objects: 100% (206/206), done.
remote: Total 605 (delta 202), reused 190 (delta 96), pack-reused 303
Receiving objects: 100% (605/605), 2.71 MiB | 3.81 MiB/s, done.
Resolving deltas: 100% (355/355), done.
PS D:\Codigo> |
```

Figura D.4: Clonación de repositorio

Instalación de dependencias

Como se ha explicado en capítulos anteriores, la aplicación tiene ciertas dependencias. La información relativa a esta se encuentra en el archivo *package.json* para poder iniciar las dependencias, tendremos que colocarnos a la altura de la carpeta raíz del proyecto y ejecutar el comando:

npm install o *npm i* El proceso de instalación de las dependencias puede llevar unos minutos, una vez terminado podemos encontrar una carpeta denominada *node_modules*



```
PS D:\Codigo\StayFresh> ls

    Directory: D:\Codigo\StayFresh

Mode Name      Length
---- --       ----
d--- assets
d--- node_modules
d--- src
-a-- .gitignore 437
-a-- App.js     1707
-a-- app.json   1158
-a-- babel.config.js 113
-a-- package-lock.json 470488
-a-- package.json 1571
-a-- README.md   628

PS D:\Codigo\StayFresh> |
```

Figura D.5: Estructura tras inicialización de dependencias

Modificar la aplicación

La aplicación no admitirá modificaciones directas a la rama *main* o *master*, la realización de modificaciones se realizará a través del concepto *pull request* y se harán siempre a la rama de desarrollo. Los pasos son los siguientes:

1. Clonar repositorio con el comando *git clone https://github.com/jsc1013/StayFresh.git*.
2. Cambiar a la rama de desarrollo con el comando *git checkout develop*.
3. Generar una rama desde la rama de desarrollo con el comando *git checkout -b nombreDelFeature*.
4. Realizar los cambios necesarios.
5. Añadir los cambios con el comando *git add*.
6. Realizar un commit con el comando *git commit*
7. Realizar un push a la rama creada con los cambios con el comando *git push*.
8. Pedir un pull request desde Github a la rama de *develop*.

Podemos comprobar la estrategia de ramas visualmente en el siguiente gráfico:

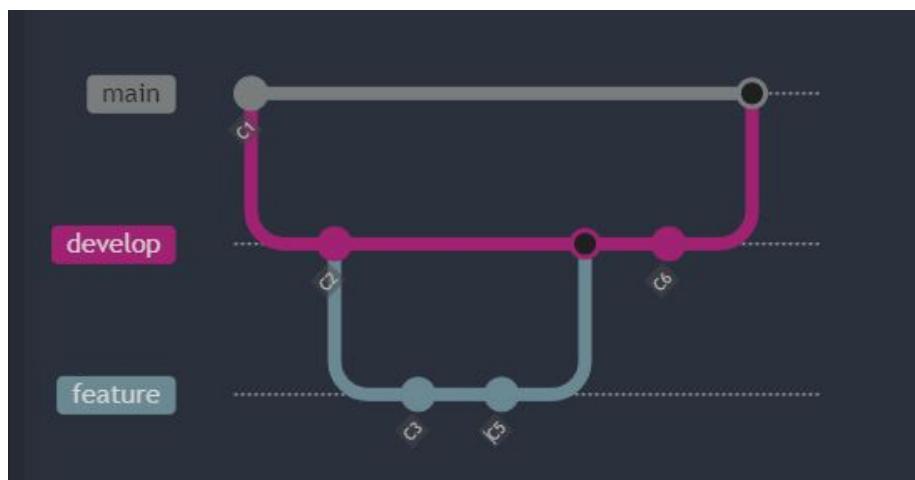


Figura D.6: Estrategia de ramas de desarrollo

Convención de commits

Para realizar un commit válido en la aplicación se deberá seguir un estándar determinado que es el que se ha usado durante todo el desarrollo de la aplicación, este estándar es el establecido por el equipo de *Angular* [1]. El estándar se resume en el uso de unos prefijos determinados en según el tipo de commit que se pueden ver a continuación:

- **build**: Cambios que afectan al sistema de compilación o dependencias externas
- **ci**: Cambios en archivos de configuración y scripts de CI
- **docs**: Cambios sólo en la documentación
- **feat**: Una nueva característica
- **fix**: Una corrección de errores
- **perf**: Un cambio de código que mejora el rendimiento
- **refactor**: Un cambio de código que no corrige un error ni añade una característica
- **style**: Cambios que no afectan al significado del código (espacios en blanco, formato, ausencia de punto y coma, etc.).
- **test**: Añadir pruebas que faltan o corregir pruebas existentes

Una vez tengamos el tipo de commit, se recomienda añadir un *scope* o el alcance del cambio entre paréntesis de tal manera que a modo de resumen se sepa a qué afecta. A modo de ejemplo, supongamos una corrección a un problema en la ventana de login:

fix(LoginScreen): Login error correction.

Todos los commits deben ser atómicos, es decir, incluir los mínimos cambios posibles para mejorar la legibilidad y seguimiento. Será necesario que se realicen en el idioma Inglés.

Ejecutar aplicación

Para ejecutar nuestra aplicación tendremos que hacerlo desde el terminal, se puede hacer directamente desde un terminal CMD o desde terminales tipo *Powershell* u otros como el embebido de Visual Studio Code que es el que mostraremos a continuación. El comando que nos levantará el bundle es: *npm start*. Este comando en realidad está definido en el archivo *package.json* y es un atajo para el comando *expo start*.



Figura D.7: npm start

Una vez que el proceso de arranque haya finalizado se nos indicará y se mostrará un código QR. En este momento surgen dos opciones. Utilizar un dispositivo real o un emulador.

- **Dispositivo real** Para poder probar la aplicación en un dispositivo real será necesario haber descargado como previamente se ha indicado

la aplicación *Expo* de la *Play Store*. Al abrir esta nos dará la opción de escanear un QR que será el que nos muestra el comando previo. Una vez realizado el vínculo, podremos usar la aplicación a tiempo real en nuestro smartphone. Para que este proceso funcione, será necesario que tanto el equipo que corre el servidor como el dispositivo móvil se encuentren en la misma red Wi-Fi.

- **Emulador** Como se ha indicado anteriormente, a través del *AVD Manager* de *Android Studio* se habrá generado un dispositivo virtual. Para arrancar este, haremos click en la esquina superior derecha de nuestro visual studio que nos abrirá un seleccionable con los distintos emuladores que tenemos instalados.

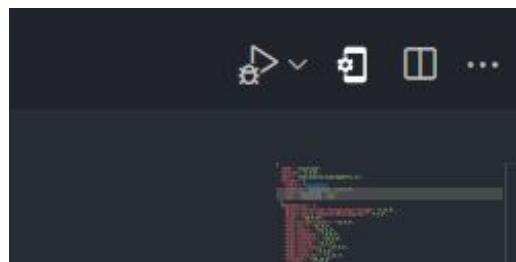


Figura D.8: Arrancar emulador Android

También podremos arrancar un emulador a través de la línea de comandos con el comando
`emulator -avd nombreDelEmulador` Una vez se haya arrancado el emulador, desde la terminal donde ejecutamos el comando `npm start`, pulsaremos la tecla a (*open Android*).

D.4. Compilación, instalación y ejecución del proyecto

Para poder compilar la aplicación a un archivo `.apk` necesitamos los siguientes requisitos:

- Tener una cuenta de Expo: Para crear una cuenta de Expo lo haremos a través del siguiente enlace <https://expo.dev/signup>
- Cliente EAS Build: *EAS (Expo Application Services)* es un servicio cloud de generación de binarios perteneciente a *Expo*. Seguiremos los siguientes pasos.
 1. Para poder hacer uso de sus diferentes funciones primero necesitaremos tener instalado el cliente. Esto lo realizaremos a través del comando
`npm install -g eas-cli.`
 2. Haremos login con la cuenta que hayamos creado de *Expo* mediante el comando
`eas login`
 3. Ejecutaremos el comando que nos preparará el proyecto para un bundle de Android.
`eas build:configure.`
 4. Finalmente generamos la aplicación con el comando.
`eas build --platform android.`

EAS nos generará un link mediante el cuál podremos seguir el proceso de compilación de nuestra aplicación. Una vez terminado nos dará la opción de subirlo a la *Play Store* o de descargar los binarios. En caso de optar por la opción de los binarios ya que la licencia de la *Play Store* es de pago, tendremos que generar el APK con la herramienta *bundletool* de Android. [10]

D.5. Buenas prácticas

La rama main de la aplicación está conectada a *Codeclimate* que ejecuta un análisis estático del código para evaluar la calidad de este, la URL a la web del análisis de la aplicación la podemos encontrar aquí:

<https://codeclimate.com/github/jsc1013/StayFresh>

Para prevenir diferencias de codificación entre los distintos posibles desarrolladores de la aplicación, se recomienda la utilización y aplicación de un Linter a tiempo real con un refuerzo de reglas de codificación. En este caso se recomienda la aplicación de las reglas de *standard*:

<https://github.com/standard/eslint-config-standard-react>

D.6. Pruebas del sistema

En esta sección se explica el procedimiento para realizar testing sobre la aplicación, el framework utilizado, estructura de carpetas y funcionamiento de los test.

Jest

El framework utilizado para la ejecución de pruebas de código es *Jest*. *Jest* es un framework de JavaScript diseñado para garantizar la corrección de cualquier código JavaScript. Permite escribir pruebas con una API accesible, familiar y rica en funciones que ofrece resultados rápidamente. [6]. Este módulo está configurado dentro del archivo *packages.json* como una dependencia de la aplicación, por lo tanto se instará al realizar un *npm install*.

```
"jest-expo": "~50.0.4",
"jest": "^29.3.1"
```

Figura D.9: Paquete jest

Carpetas

Como se ha podido ver en el apartado de estructura de carpetas, la aplicación contiene una serie de carpetas denominadas `__test__`. La filosofía de este tipo de testing es no tener una única carpeta donde se engloben todos los test o un solo archivo, sino que, cada apartado de la aplicación tenga asociada su carpeta de test con un *scope* acotado a esa sección. Podemos verlo ilustrado en la siguiente imagen:

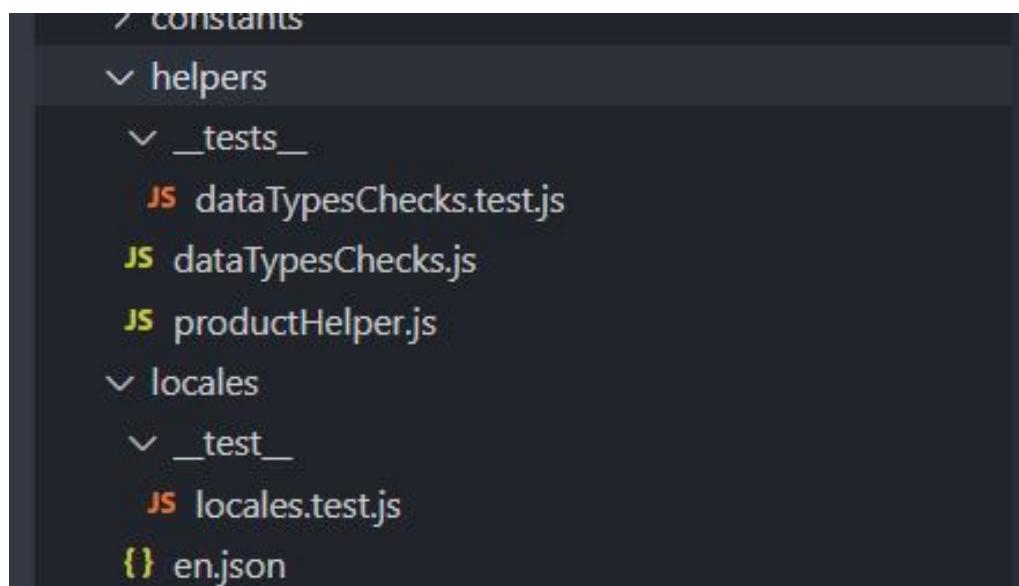


Figura D.10: Estructura de test

Como se puede apreciar, tanto la carpeta *helpers* como la carpeta *locales* tienen su propia carpeta `__test__`.

Archivos de test

Los archivos de test tendrán la siguiente estructura en su nombre: *nombreDeLaClaseAProbar.test.js*. Se creará un archivo diferente por cada clase o archivo del que queramos realizar pruebas. Los test seguirán el estándar de *Jest* que podemos encontrar en su documentación [7]. Podemos ver un ejemplo de test en la siguiente imagen:

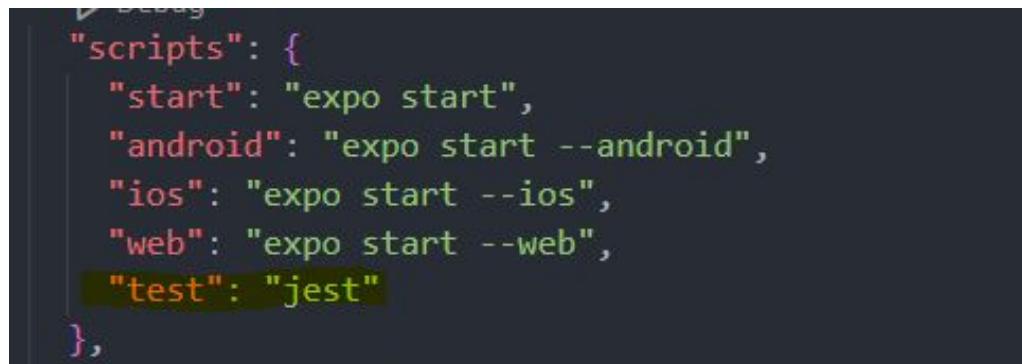
```
src > locales > __test__ > JS locales.test.js > ...
1 import en from "../en.json";
2 import es from "../es.json";
3
4 describe("Validación de formato JSON del módulo en", () => {
5   test("debe contener las propiedades esperadas", () => {
6     expect(en).toHaveProperty("components");
7     expect(en).toHaveProperty("components.login");
8     expect(en).toHaveProperty("components.home");
9     expect(en).toHaveProperty("components.addProduct");
10    expect(en).toHaveProperty("components.modal");
11    expect(en).toHaveProperty("components.homeManagement");
12    expect(en).toHaveProperty("components.consumeProduct");
13    expect(en).toHaveProperty("components.storage");
14    expect(en).toHaveProperty("general");
15  });
16})
```

Figura D.11: Ejemplo de test

En la imagen podemos ver que el *Scope* del test son los archivos locales que están en una ruta anterior a la carpeta `__test__`. En el apartado *describe* describiremos el test y luego añadiremos las condiciones. En este caso es una validación de que el archivo JSON de los locales está bien formado y contiene todos los datos necesarios.

Script

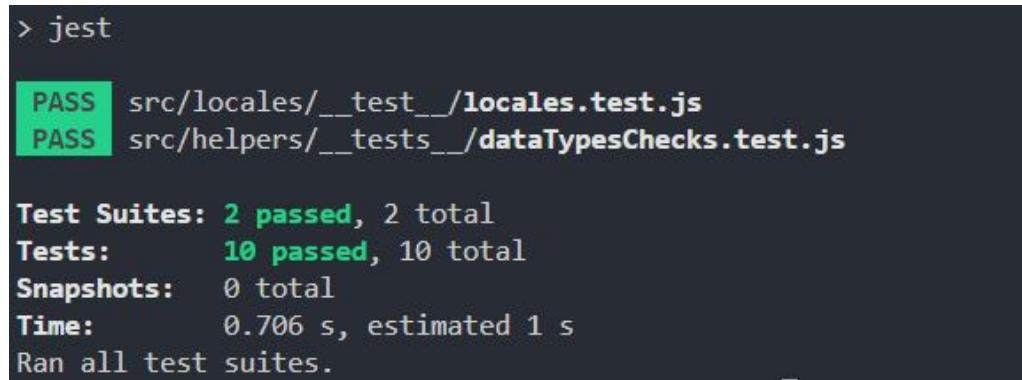
Para lanzar los test se ha añadido un script al archivo *packages.json*. Que podremos ejecutar a través del comando *npm test*.



```
"scripts": {  
  "start": "expo start",  
  "android": "expo start --android",  
  "ios": "expo start --ios",  
  "web": "expo start --web",  
  "test": "jest"  
},
```

Figura D.12: Test script

Este script nos ejecutará jest y nos arrojará los resultados de los test a través de la consola:



```
> jest  
  
PASS  src/locales/__test__/locales.test.js  
PASS  src/helpers/__tests__/dataTypesChecks.test.js  
  
Test Suites: 2 passed, 2 total  
Tests:       10 passed, 10 total  
Snapshots:   0 total  
Time:        0.706 s, estimated 1 s  
Ran all test suites.
```

Figura D.13: Resultado de test

Podremos ejecutar los test con un comando alternativo mediante el ejecutor de paquetes *npx*. El comando es: *npx jest*. Si además de comprobar que estamos pasando los test queremos también obtener la cobertura de la aplicación, podremos utilizar el comando *npx jest -coverage*

```
PASS  src/locales/_test_/locales.test.js
PASS  src/helpers/_tests_/dataTypesChecks.test.js
-----|-----|-----|-----|-----|-----|
File      | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files |    100 |     100 |     100 |     100 |
dataTypesChecks.js | 100 | 100 | 100 | 100 |
-----|-----|-----|-----|-----|-----|
Test Suites: 2 passed, 2 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        0.765 s, estimated 1 s
Ran all test suites.
```

Figura D.14: Resultado de cobertura

CI/Github Actions

Para controlar la integración continua de la aplicación, esta ha sido preparada para sincronizarse con *Github Actions*. Se ha incluido el siguiente workflow:

```
1   name: StayFresh Testing
2
3   on:
4     pull_request:
5       branches:
6         - main
7         - develop
8     workflow_dispatch:
9     push:
10       branches:
11         - main
12         - develop
13
14   jobs:
15     unit-test:
16       name: Run Tests
17       runs-on: ubuntu-latest
18       steps:
19         - uses: actions/checkout@v2
20         - uses: actions/setup-node@v1
21         with:
22           node-version: "20"
23         - run: npm install
24         - run: npm run test
```

Figura D.15: Github actions

A continuación se explica el script:

- pullrequest: Cada vez que se realice un pull request a las ramas main o develop, se ejecutará el workflow.
- workflow_dispatch: Nos permite realizar una ejecución manual del workflow.
- push: Cuando se realice un push a las ramas main o develop, se ejecutará el workflow.
- runs-on: Indica el S.O donde se realizarán las pruebas.
- steps: Indica los pasos a realizar
 1. Checkout al repo
 2. Preparar el entorno de *Node.js* con la versión 20 que es la que usa nuestro proyecto.
- npm install: Instalar todas las dependencias.
- npm run test: Ejecutar los test.

Si los test han pasado satisfactoriamente encontraremos la siguiente marca en el historial de commits:

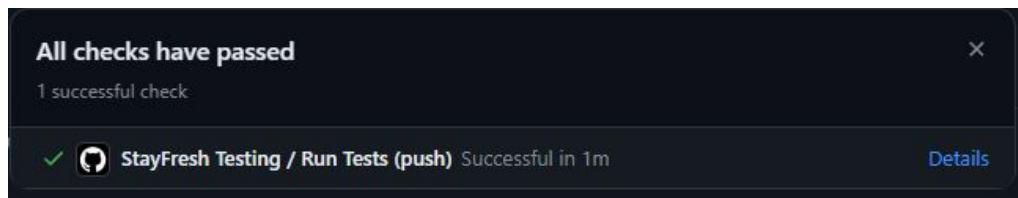


Figura D.16: Comprobaciones pasadas

Apéndice E

Documentación de usuario

E.1. Introducción

En este anexo se detalla los requisitos necesarios para la instalación de la aplicación en dispositivos con sistema operativo Android además del proceso paso a paso para la instalación.

E.2. Requisitos de usuarios

A continuación se listan los requisitos mínimos para poder ejecutar la aplicación:

- Dispositivo Android con una versión de sistema operativo mayor a Android 6 (Marshmallow).
- Habilitar instalación de aplicaciones de fuentes desconocidas en el Smartphone.
- Conexión a Internet.
- Cuenta de correo electrónico.

E.3. Instalación

A continuación se detallan los pasos para la instalación de la aplicación:

1. Permitir los orígenes desconocidos en el dispositivo:
 - a) Dirigirse a Configuración -> Seguridad.
 - b) Marcar la opción *Fuentes desconocidas*.
 - c) Confirmar los términos.
2. Acceder al apartado Releases del repositorio de Github de la aplicación:
<https://github.com/jsc1013/StayFresh/releases>
3. Localizar la última versión de la aplicación (Siempre tendrá un número de versión más alto que el resto y se situará en la parte superior de la página).
4. Seleccionar el archivo *StayFresh.apk*. A continuación se nos descargará automáticamente.

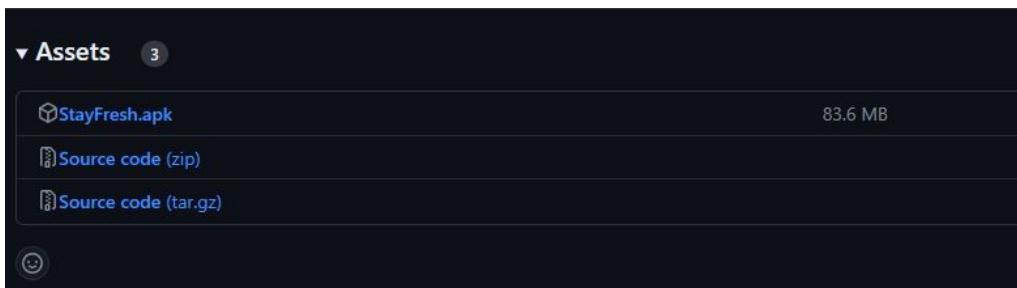


Figura E.1: Descarga de APK

5. Ejecutar el archivo .APK descargado y esperar a que finalice el proceso de instalación.

E.4. Manual del usuario

En esta sección se detalla cómo registrarse, primeros pasos y el funcionamiento de los distintos apartados de la aplicación.

Registro y login

Nada más abrir la aplicación nos encontraremos con el menú de login y registro. Para crear una cuenta si no tenemos, escribiremos nuestro email y contraseña que queramos para la cuenta.

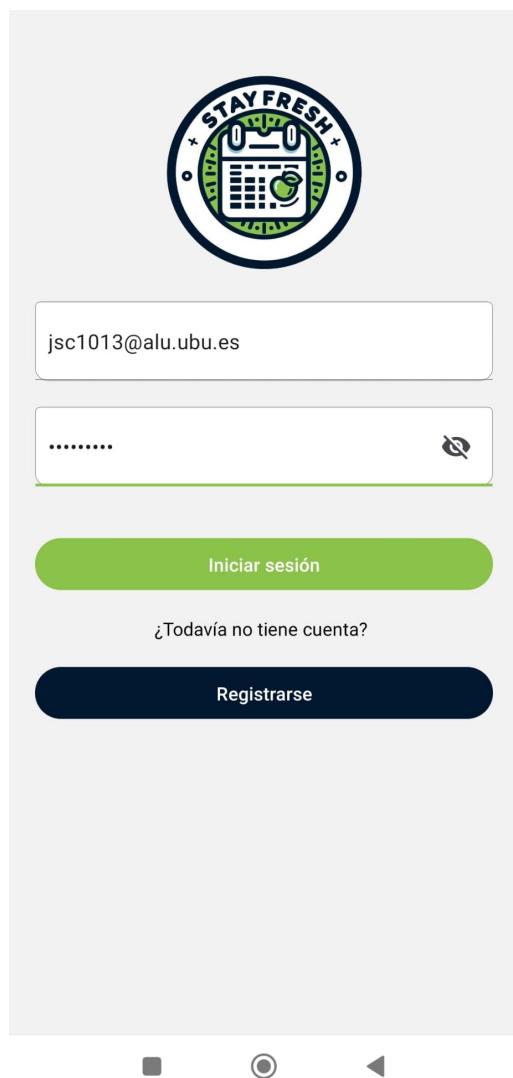


Figura E.2: Registro 1

Podemos ver la contraseña pulsando el ícono con el ojo en el borde derecho.

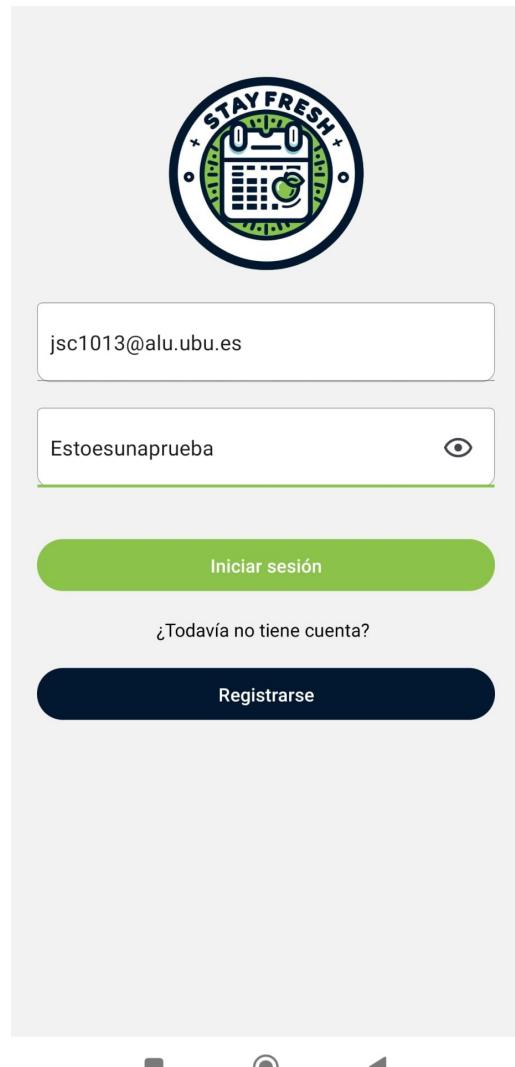


Figura E.3: Registro 2

Presionaremos el botón *Registrarse* y nos aparecerá un pop-up indicando que se ha enviado el email de validación de la cuenta. Tendremos que ir a nuestro email, comprobar la bandeja de entrada y buscar uno que provenga de la cuenta StayFresh. Dentro del mail encontraremos un link que hay que pulsar para validar la cuenta. Es importante revisar el buzón de *spam* si no encontramos el email en nuestra bandeja principal



Figura E.4: Registro 3

Tutorial

La primera vez que entremos a la aplicación, se mostrará un tutorial paso a paso del funcionamiento de esta. Para avanzar en el tutorial podemos deslizar hacia la derecha o presionar el botón siguiente.

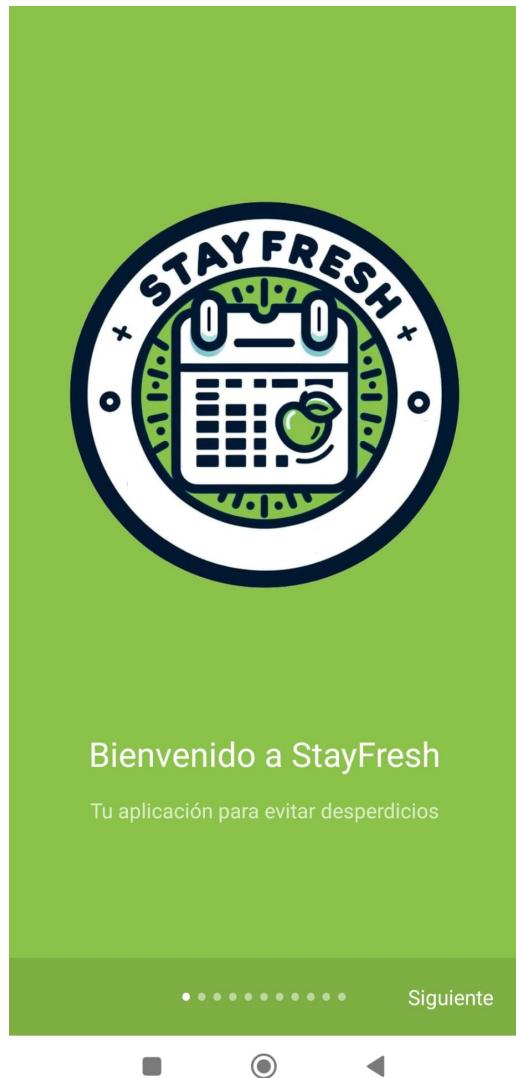


Figura E.5: Tutorial 1

Al llegar a la última pantalla del tutorial el botón siguiente cambiará y ya podremos cerrarlo.

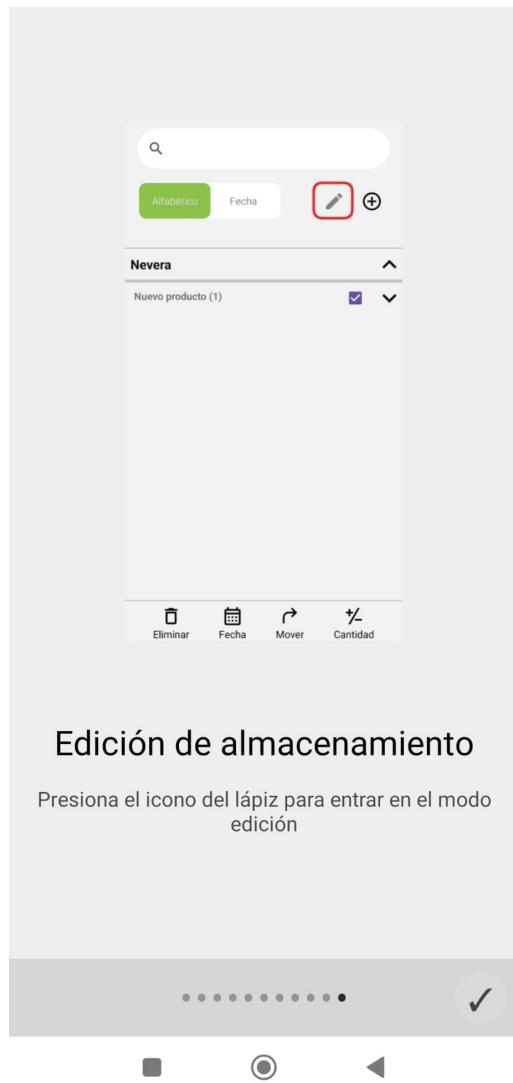


Figura E.6: Tutorial 2

Menú principal

Tras terminar el tutorial, se nos mostrará el menú principal de la aplicación. A continuación se detallan todos sus componentes.

- Gestión de hogares: Podemos encontrar este menú en el símbolo de la casa en la esquina superior izquierda. Nos permitirá comprobar el hogar actual, añadir nuevos hogares o salir de estos.
- Cerrar sesión: Se encuentra en la parte superior derecha.
- Añadir productos: Lo podemos encontrar con un símbolo de un código de barras y un símbolo más con el texto: Añadir. En este menú podremos añadir productos a nuestro stock.
- Consumir productos: Lo podemos encontrar con un símbolo de un código de barras y un menos más con el texto: Consumir. En este menú podremos retirar productos de nuestro stock.
- Almacenamiento: Lo podemos encontrar con un símbolo de unas estanterías y el texto almacenamiento. Nos permitirá gestionar de manera global tanto nuestros contenedores como nuestros productos.
- Resumen de productos: Debajo de los botones de los menús, encontramos un texto que indica los próximos X días que queremos previsualizar respecto a los alimentos que caducan. Si presionamos este texto, nos dejará cambiar los días por defecto. A su derecha encontramos un botón para refrescar la vista y justo debajo tendríamos el resumen de productos en caso de tener alguno que cumpla las condiciones de fecha. Los pasos que tendremos que dar son:
 1. Crear un hogar.
 2. Crear un contenedor.
 3. Añadir alimentos.

A continuación se muestra una imagen del menú principal

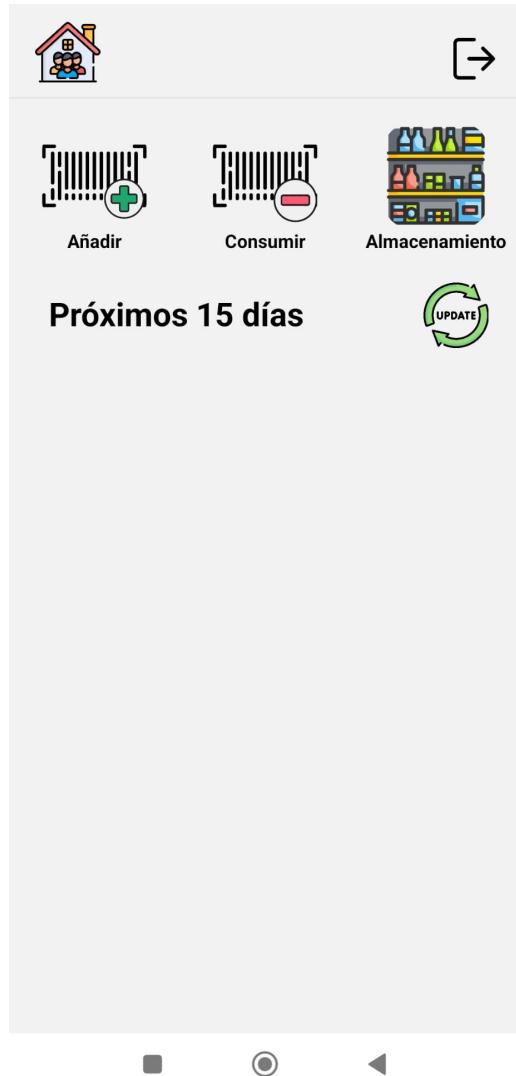


Figura E.7: Menú principal 1

Si intentamos acceder a los menús añadir, consumir o almacenamiento sin haber creado previamente un hogar, la aplicación nos mostrará un error.

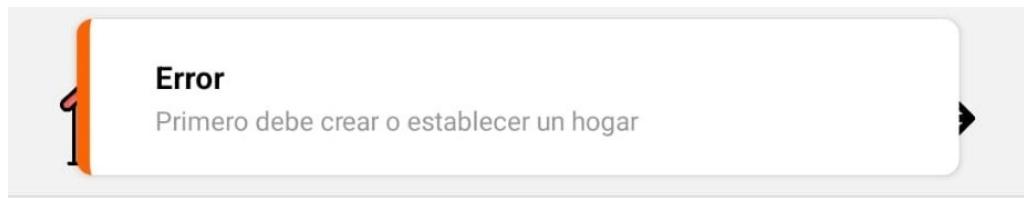


Figura E.8: Menú principal 2

Añadir un hogar

En este menú encontraremos dos submenús principales:

- Mis hogares: Nos permitirá gestionar los hogares que tenemos añadidos: Compartirlos, editar su nombre y salir de estos.
- Añadir hogares: Nos permitirá añadir nuestros hogares.

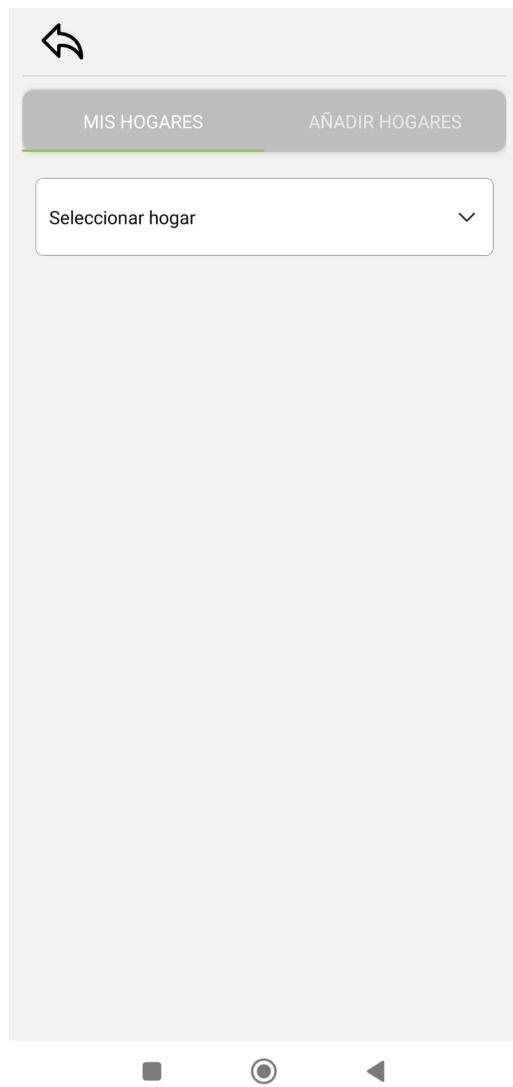


Figura E.9: Añadir un hogar 1

Primero crearemos un hogar, para ello vamos al submenú añadir hogares. Se nos presentan dos opciones:

- Hogar existente: Si queremos adjuntarnos a un hogar que haya creado otra persona, podremos usar esta opción. Para ello tendremos que escanear el código QR del hogar de la otra persona o copiar su identificador. Una vez introducido podremos renombrarlo sin afectar a otros usuarios.
- Nuevo hogar: Si queremos crear un nuevo hogar simplemente introduciremos el nombre de este.

Tras añadir un hogar, se nos mostrará un mensaje como el siguiente:

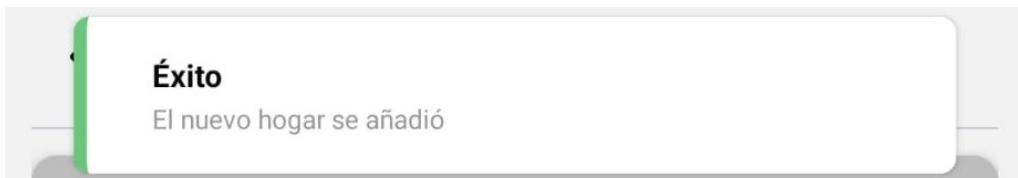


Figura E.10: Añadir un hogar 2

Al volver al submenú mis hogares podremos ver que el hogar ya nos aparece en el listado y podremos seleccionarlo. Se nos mostrarán distintas opciones:

- Establecer por defecto: Nos permitirá marca el hogar como el hogar principal.
- Editar nombre: Podremos cambiar el nombre del hogar sin afectar a otros usuarios.
- QR: Es el código que podremos facilitar a otros usuarios para que se incorporen a nuestro hogar. También se puede copiar el identificador al portapapeles presionando el QR.
- Salir del hogar: Nos permitirá abandonar el hogar seleccionado. Si no quedan más usuarios en el hogar, este será eliminado.



Figura E.11: Añadir un hogar 3

Establecer hogar por defecto

Si seleccionamos la opción establecer por defecto, se mostrará un mensaje de éxito, se añadirá un ícono en el lado izquierdo del hogar para diferenciarlo del resto y pasará a ser el hogar principal del usuario.



Figura E.12: Añadir un hogar 4

Al volver al menú principal el hogar marcado como principal se verá indicado en la parte superior izquierda y ya se permitirá la navegación al resto de menús. El siguiente paso será añadir un contenedor de almacenamiento.

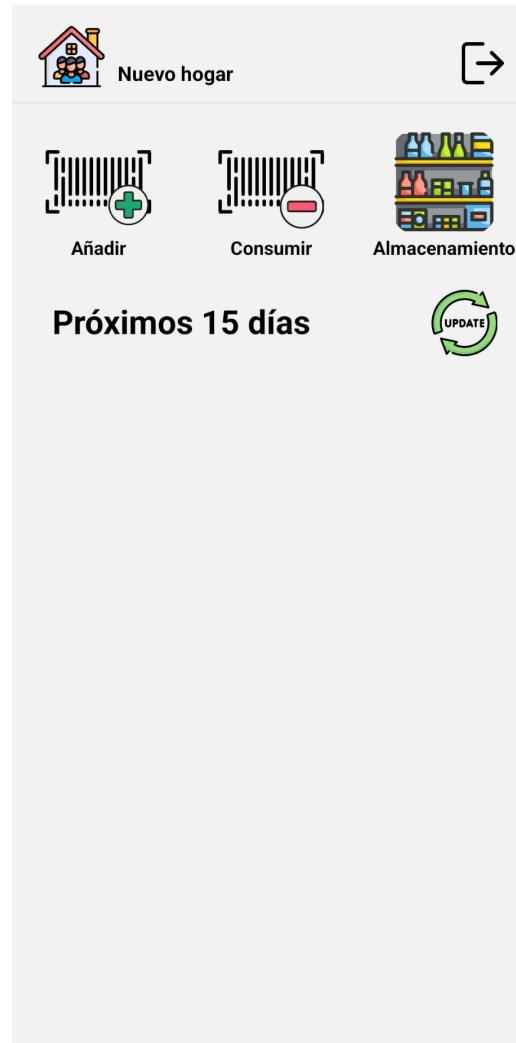


Figura E.13: Añadir un hogar 5

Añadir almacenamiento

Para añadir un nuevo almacenamiento que contenga nuestros productos iremos al menú almacenamiento.

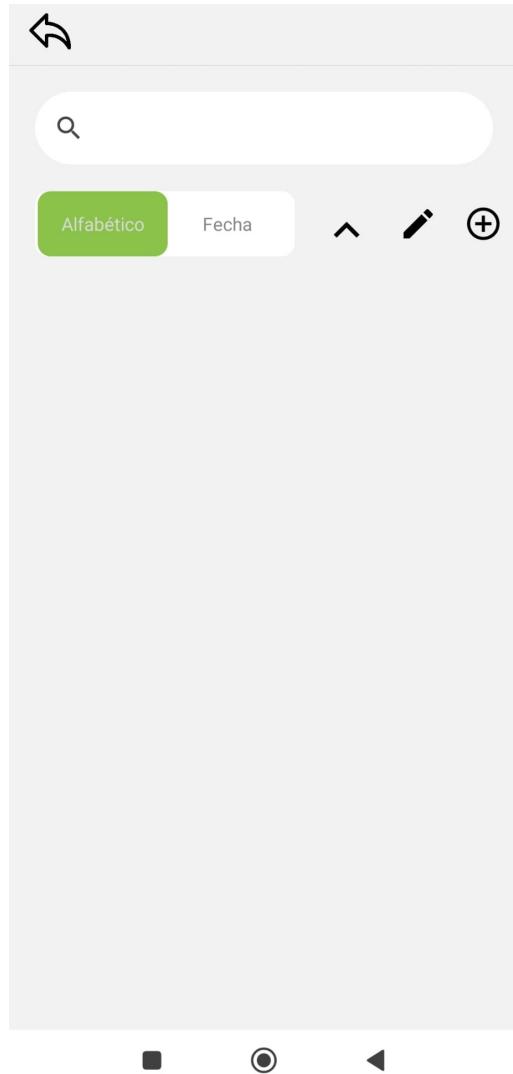


Figura E.14: Añadir almacenamiento 1

Una vez en este presionaremos el icono con el símbolo más e introduciremos el nombre para el almacenamiento.



Figura E.15: Añadir almacenamiento 2

Tras añadirlo se nos mostrará un mensaje y aparecerá en el listado inferior. Ya podremos añadir productos.

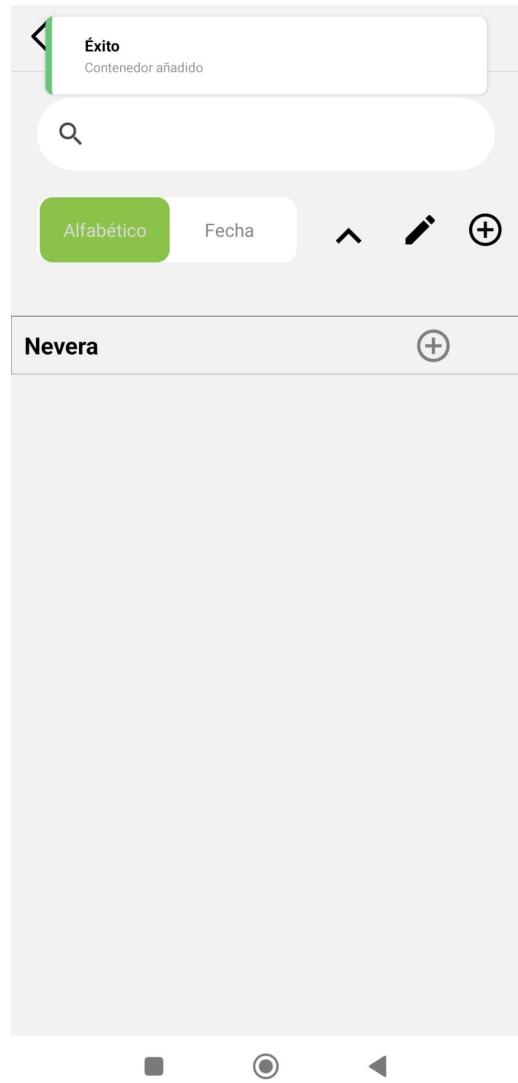


Figura E.16: Añadir almacenamiento 3

Añadir productos

Para añadir productos tenemos dos opciones:

- Ir a menú añadir: Salir del menú almacenamiento e ir al menú añadir donde tendremos que seleccionar de nuevo el almacenamiento.
- Presionar sobre el ícono más a la derecha de un almacenamiento: Se nos abrirá el menú añadir con el contenedor ya seleccionado.

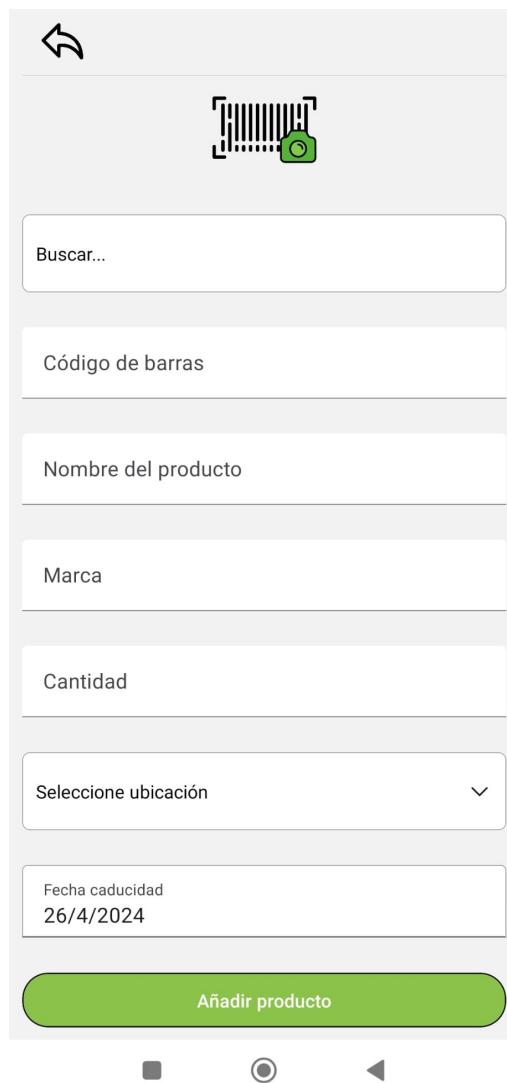


Figura E.17: Añadir productos 1

Para añadir un producto tenemos dos opciones:

- Que ya existe: Si el producto ya existe, podremos buscarlo en el menú de buscar o escanear su código de barras. La aplicación recuerda productos añadidos de los últimos seis meses.
- Que no existe: Si el producto no existe, podremos introducir sus datos manualmente, la introducción del código de barras se puede realizar a través de la cámara.

A continuación se muestra la segunda opción donde la aplicación nos informará y nos pedirá que rellenemos los datos:

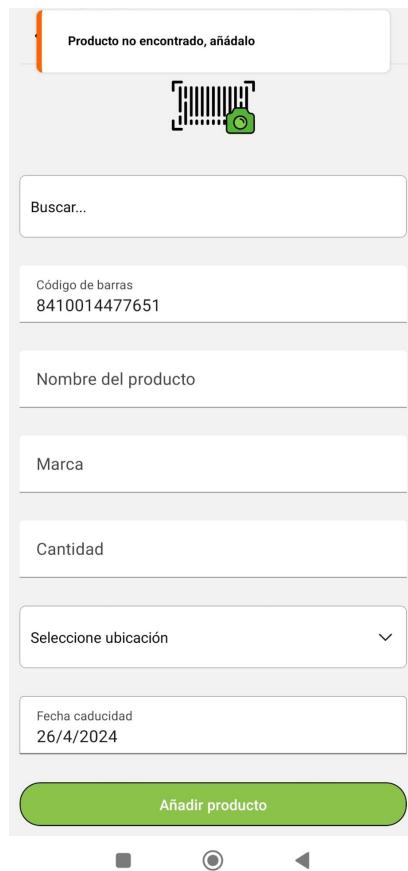


Figura E.18: Añadir productos 2

Complearemos los datos del producto y presionaremos añadir.

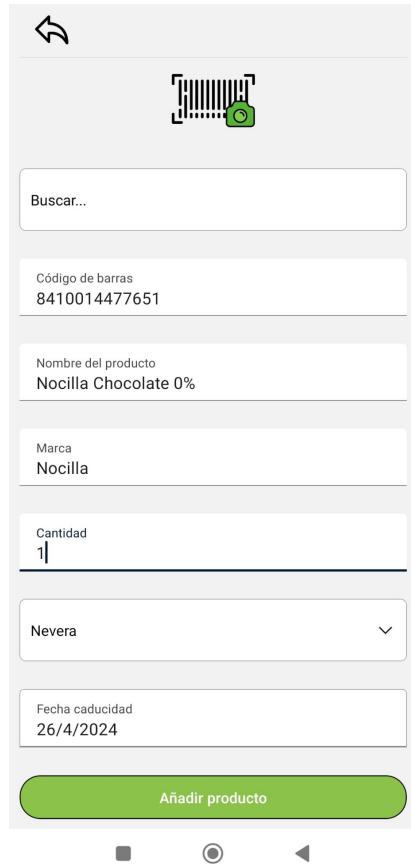


Figura E.19: Añadir productos 3

Se nos mostrará un mensaje de éxito tras añadirlo



Figura E.20: Añadir productos 5

Si tuviéramos un producto ya añadido, podríamos buscarlo directamente filtrando por:

- Nombre.
- Marca.
- Código de barras.

Los productos encontrados se verían a modo de listado y se podrían seleccionar:

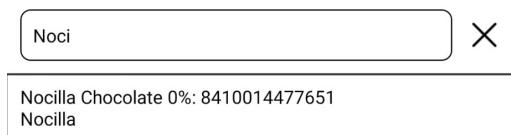


Figura E.21: Añadir productos 6

Resumen de productos

Al volver al menú principal, si el producto añadido cumple las condiciones del filtro establecido, se mostrará, en la siguiente imagen se muestra el producto que acabamos de añadir el cual tiene una caducidad del día actual. En el resumen se muestra una categorización de colores para los distintos productos en función de los días que queden para su caducidad. Los días los podemos ver en la parte izquierda: Si pasan de siete, se cambiará el modo a semanas.

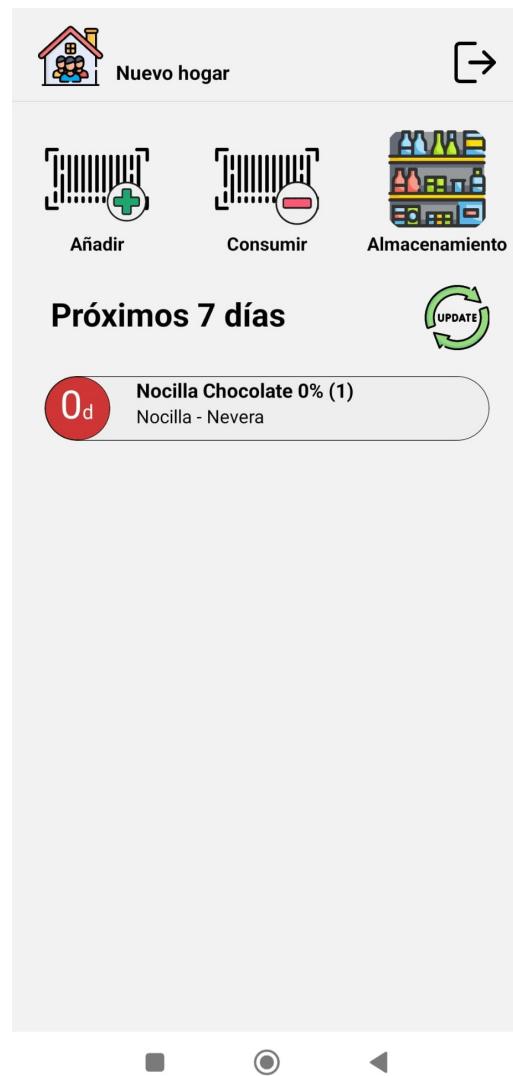


Figura E.22: Resumen de productos 1

Podremos cambiar los días de previsualización presionando el texto:

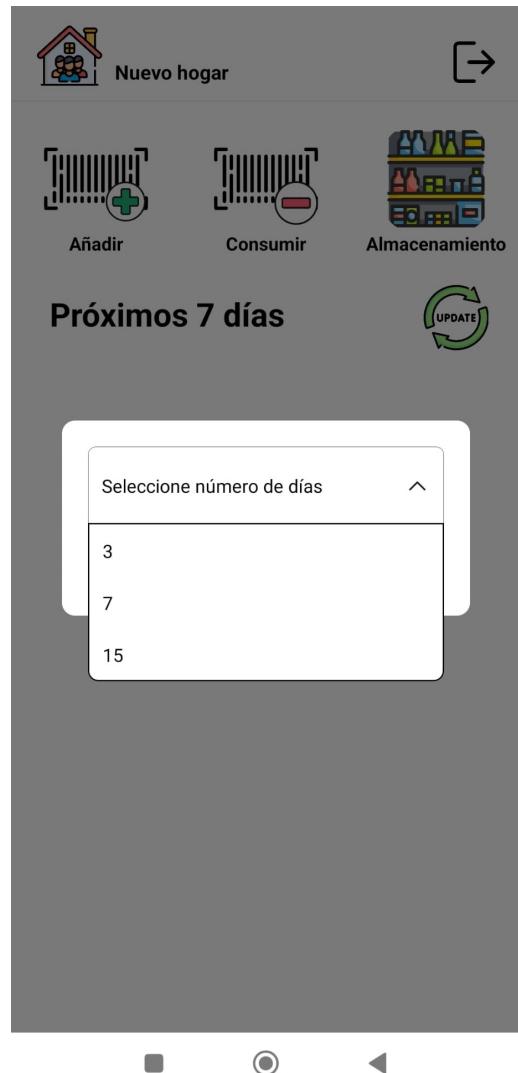


Figura E.23: Resumen de productos 2

Para editar rápidamente un producto en el listado, podremos deslizarlo hacia la izquierda y tendremos las opciones de cambiar unidades o eliminar producto.



Figura E.24: Resumen de productos 3

Gestión de almacenamiento

Teniendo el producto añadido, si vamos al almacenamiento, se mostrará este ordenado en el contenedor elegido. Podemos ordenar los productos por orden alfabético o por fecha de caducidad presionando sobre Alfabético o sobre Fecha.

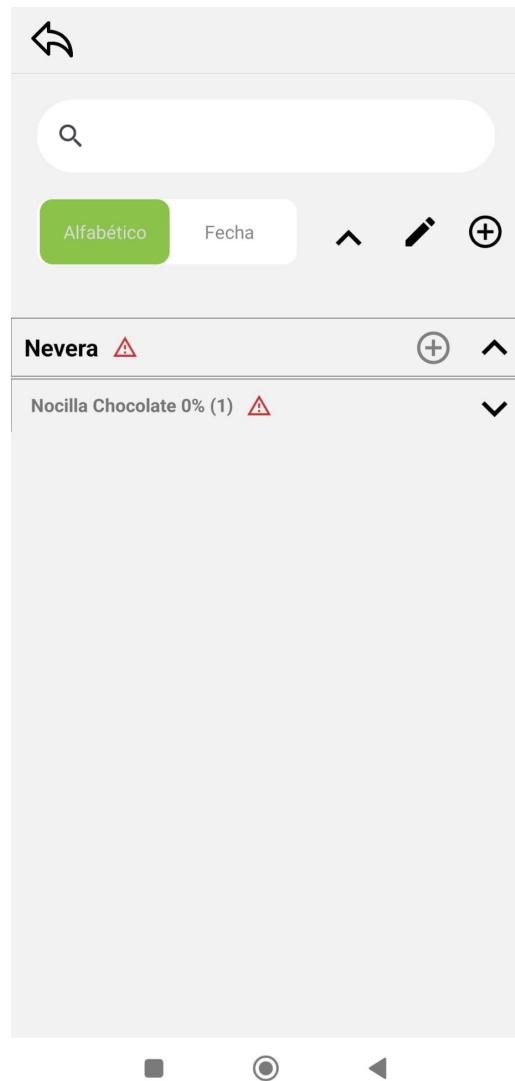


Figura E.25: Gestión de almacenamiento 1

Si presionamos el icono del lápiz en el menú, entraremos en el modo edición, podremos seleccionar distintos elementos y en función de la selección se mostrarán distintas opciones en la cinta inferior. Podremos:

- Eliminar: Nos permitirá eliminar tanto contenedores como productos.
Si se elimina un contenedor, se eliminará su contenido.
- Fecha: Nos permitirá cambiar la fecha de caducidad de un producto.
- Mover: Podremos mover productos entre contenedores.
- Cantidad: Podremos cambiar las unidades de un producto.

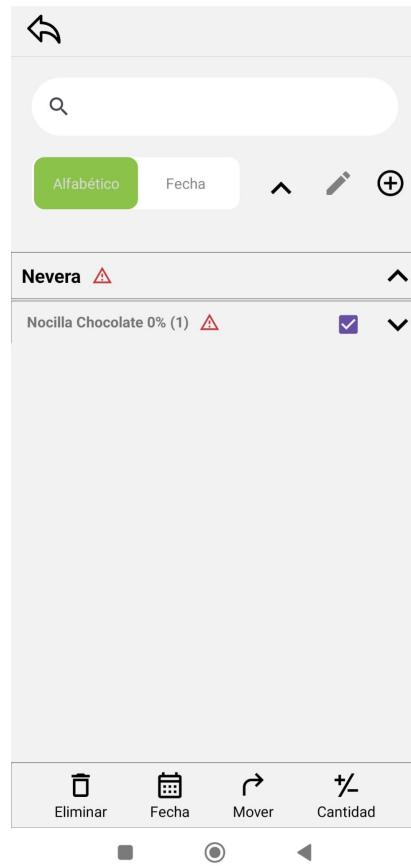


Figura E.26: Gestión de almacenamiento 2

Si tuviéramos varias unidades de un mismo producto bajo el mismo contenedor, éstas se agruparían:

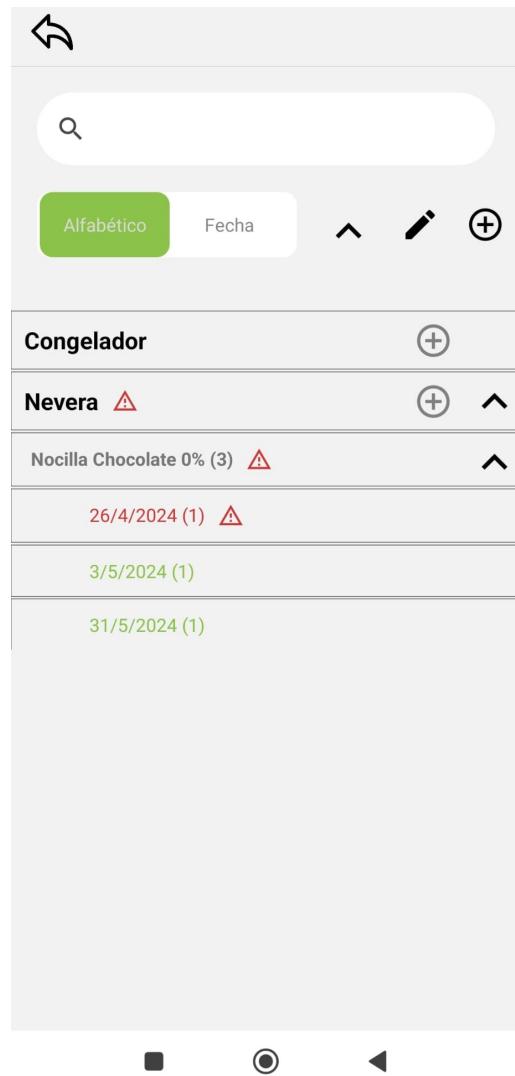


Figura E.27: Gestión de almacenamiento 3

Consumir productos

En este menú podremos retirar productos del stock, estos pueden ser buscados o escaneados.

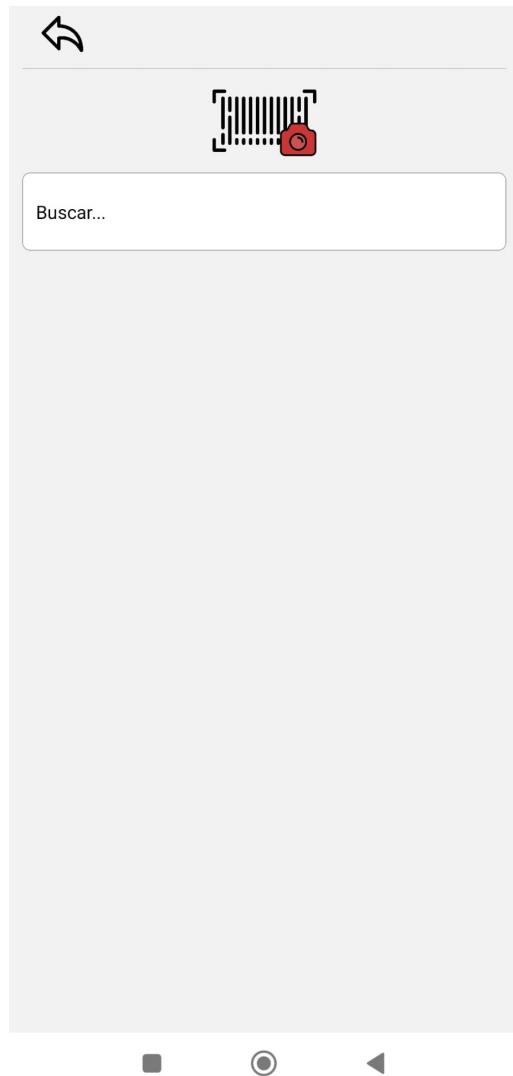


Figura E.28: Consumir productos 1

Al buscar o escanear un producto se nos mostrarán las coincidencias de este:

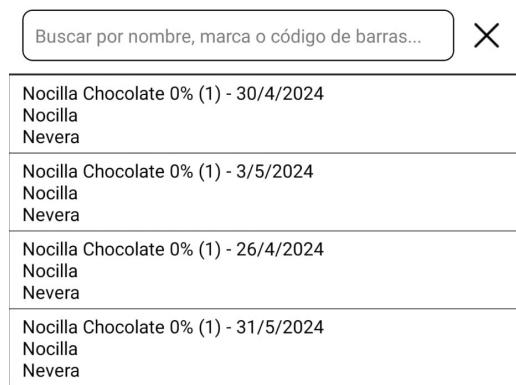


Figura E.29: Consumir productos 2

Categorización de productos

Los productos tendrán distintos colores basándose en los días restantes para la caducidad de estos:

- Negro: Productos ya caducados: Mostrará la fecha negativa de los días que lleva caducado.
- Rojo: Productos que van a caducar, menos de tres días.
- Naranja: Productos cercanos al riesgo de caducidad, de tres a siete días.
- Verde: Productos que están lejos de la caducidad, más de siete días.

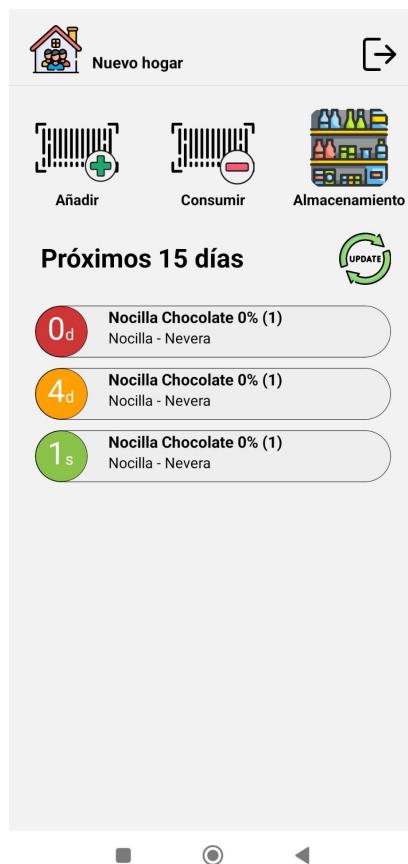


Figura E.30: Categorización de productos 1

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo detalla cómo la aplicación contribuye a los Objetivos de Desarrollo Sostenible (ODS) promulgados por las Naciones Unidas. La relevancia de este proyecto radica en su potencial para optimizar el uso de alimentos en el ámbito doméstico, reduciendo el desperdicio y así promover un consumo más responsable.

F.2. Objetivos relacionados

A continuación se presentan los principales Objetivos de Desarrollo Sostenible con los que se relaciona la aplicación.

- **ODS 2, hambre cero:** El objetivo 2 es crear un mundo libre de hambre para 2030. El problema global del hambre y la inseguridad alimentaria ha mostrado un aumento alarmante desde 2015, una tendencia exacerbada por una combinación de factores que incluyen la pandemia, los conflictos, el cambio climático y la profundización de las desigualdades [18]. Esta aplicación busca optimizar la utilización de los alimentos en los hogares evitando el desperdicio al tener un control sencillo sobre su estado lo cual puede contribuir al ODS2 con una reducción del hambre local al disminuir la cantidad de alimentos desechados innecesariamente.

- **ODS 12, producción y consumo responsables:** El Objetivo 12 pretende garantizar modalidades de consumo y producción sostenibles, algo fundamental para sostener los medios de subsistencia de las generaciones actuales y futuras. Nuestro planeta se está quedando sin recursos, pero el índice de población sigue creciendo. En caso de que la población mundial alcance los 9800 millones de personas en 2050, se podría necesitar el equivalente a casi tres planetas para proporcionar los recursos naturales necesarios para mantener los estilos de vida actuales [16]. La aplicación promueve hábitos de consumo más sostenibles ayudando a minimizar el impacto ambiental del desperdicio alimentario.
- **ODS 13, medidas urgentes para combatir el cambio climático y sus efectos:** El cambio climático afectará a todas las personas de todos los países de todos los continentes de alguna forma. Se avecina un cataclismo climático y no estamos preparados para las posibles consecuencias. El cambio climático se debe a las actividades humanas y amenaza la vida en la Tierra tal como la conocemos. Con el aumento de las emisiones de gases de efecto invernadero, el cambio climático evoluciona a un ritmo mucho más rápido de lo previsto. Sus efectos pueden ser devastadores y pueden provocar fenómenos meteorológicos extremos y cambiantes, así como la subida del nivel del mar [17]. Reducir el desperdicio de alimentos también implica una disminución en la cantidad de residuos que terminan en vertederos, donde generan metano, un potente gas de efecto invernadero. Al disminuir el desperdicio, la aplicación contribuye a la acción climática.

F.3. Adherencia

A la hora de establecer una rutina sea del tipo que sea, lo más importante es la adherencia a esta. Lo mismo se traslada a las aplicaciones, sus principios de diseño deben estar centrados en el usuario para asegurar su eficacia así como su facilidad de uso de tal manera que generar una rutina constante sea un hito fácilmente alcanzable. Desde la release 1.0.0 de la aplicación, se ha estado usando en mi entorno personal con el fin de recabar datos respecto a su funcionamiento e interés dando a conocer que su manejo es sencillo, rápido e intuitivo facilitando que el uso de la aplicación no sea tedioso y creando así una adherencia a la aplicación que termina convirtiéndose en rutina.

Lo indicado previamente es de suma importancia ya que si uno de los objetivos de la aplicación es aportar a los ODS, tiene que conseguir que los usuarios establezcan sus rutinas y no las abandonen.

F.4. Futuras líneas de trabajo

Actualmente la aplicación carece de apartados exclusivos relativos a los ODS que aporten datos al usuario de manera directa. Como futuras líneas de desarrollo, se podrá tener un módulo de estadísticas del usuario que indique a modo resumen cuántos productos ha consumido a tiempo y cuántos han caducado, de tal forma que se pueda dar un ranking al usuario y que cause competitividad en él para fomentar el uso de la aplicación.

Bibliografía

- [1] Angular. Angular commit convention. <https://github.com/angular/angular/blob/22b96b9/CONTRIBUTING.md#-commit-message-guidelines>, 2018. [Internet; accedido 21-abril-2024].
- [2] European commission. Datos personales. https://commission.europa.eu/law/law-topic/data-protection/reform/what-personal-data_es#:~:text=Los%20datos%20personales%20son%20cualquier,constituyen%20datos%20de%20car%C3%A1cter%20personal., 2024. [Internet; accedido 01-mayo-2024].
- [3] Creative Commons. Licenses. <https://creativecommons.org/licenses/by/4.0/deed.es>, 2024. [Internet; accedido 01-mayo-2024].
- [4] Agencia española de protección de datos. Principios relativos al tratamiento de datos. <https://www.aepd.es/preguntas-frecuentes/2-rgpd/3-principios-relativos-al-tratamiento/FAQ-0207-cuales-son-los-principios-relativos-al-tratamiento-de-datos>, 2024. [Internet; accedido 01-mayo-2024].
- [5] Expo. Expo app. https://play.google.com/store/apps/details?id=host.exp.exponent&hl=es_PY, 2024. [Internet; descargado 21-abril-2024].
- [6] OpenJS Foundation. Jest. <https://jestjs.io/es-ES/>, 2019. [Internet; accedido 23-abril-2024].
- [7] OpenJS Foundation. Jest documentación. <https://jestjs.io/es-ES/docs/getting-started>, 2019. [Internet; accedido 23-abril-2024].

- [8] Openjs foundation. Node js v20.11.1. <https://nodejs.org/en/blog/release/v20.11.1>, 2024. [Internet; descargado 21-abril-2024].
- [9] Google. Firebase. <https://firebase.google.com/?hl=es>, 2011. [Internet; accedido 21-abril-2024].
- [10] Google. Cómo compilar y probar un paquete de aplicación. <https://developer.android.com/tools/bundletool?hl=es-419>, 2024. [Internet; accedido 21-abril-2024].
- [11] Google. Cómo crear y administrar dispositivos virtuales. <https://developer.android.com/studio/run/managing-avds?hl=es-419>, 2024. [Internet; accedido 21-abril-2024].
- [12] Microsoft. Visual studio code. <https://code.visualstudio.com/download>, 2015. [Internet; descargado 21-abril-2024].
- [13] Openai. Dall-e licenses. <https://help.openai.com/en/articles/6425277-can-i-sell-images-i-create-with-dall-e>, 2024. [Internet; accedido 01-mayo-2024].
- [14] Seguridad Social. Bases y tipos de cotización 2024. <https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538>, 2024. [Internet; accedido 01-mayo-2024].
- [15] Linus Torvalds. Git download. <https://git-scm.com/downloads>, 2005. [Internet; descargado 21-abril-2024].
- [16] Naciones Unidas. Objetivo de desarrollo sostenible 12. <https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/>, 2024. [Internet; accedido 01-mayo-2024].
- [17] Naciones Unidas. Objetivo de desarrollo sostenible 13. <https://www.un.org/sustainabledevelopment/es/climate-change-2/>, 2024. [Internet; accedido 01-mayo-2024].
- [18] Naciones Unidas. Objetivo de desarrollo sostenible 2. <https://www.un.org/sustainabledevelopment/es/hunger/>, 2024. [Internet; accedido 01-mayo-2024].