HOMIES

Una app para familias o compañeros de piso

Sofía Cid González

UOC Proyecto final del Grado de Ingeniería Informática

CONTENIDO

TABLA DE ILUSTRACIONES Y GRÁFICOS

1. INTRODUCCIÓN

1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

Gestionar una casa no es un trabajo fácil. Está plagado de numerosas tareas que consumen mucho tiempo y requieren planificación, como comprar, hacer la comida o limpiar la casa. Este tipo de cuestiones son aún más complicadas cuando se trata de compartir casa, ya sea con familiares o amigos.

A las tareas cotidianas, se suma la necesidad de planificar quién hace qué y tomar decisiones con los demás, por ejemplo, quién organiza las tareas de limpieza o qué debe comerse cada día. Además, es fácil encontrar dificultades para coordinarse con el resto de la familia, por lo que es común encontrar varias listas de la compra independientes, calendarios personales con eventos distintos, etc. Estas situaciones, acaban generando dificultades en la gestión y la convivencia.

Tratando de solucionar estos problemas, surge ***Homies***, una app para **familias y compañeros de piso**, diseñada para ayudarte a **organizar tu hogar** desde un mismo sitio. De este modo, trata de dar respuesta a las principales necesidades de las diferentes familias o compañeros de piso permitiendo que todos los miembros de la familia sepan qué actividades hay que hacer y cómo pueden colaborar.

La app, por tanto, proporciona información sobre qué cosas hay que comprar o qué se comerá en los próximos días. Ofrece una única lista de la compra, un calendario común, y ayuda a los participantes de la familia a organizar las tareas de la limpieza, evitando así las peleas futuras sobre a quién le tocaba hacer qué. Además, con el fin de facilitar la gestión económica, especialmente en los compañeros de piso, permite anotar los gastos comunes y repartirlos entre el resto de la familia.

Actualmente, existen en el mercado apps destinadas a este fin, aunque la mayoría de pago o con funcionalidades limitadas. Por un lado, tendríamos algunas soluciones de gestión completa. Algunos ejemplos son:

- **Cozi:** Es una aplicación para la gestión familiar bastante potente. Como aspectos positivos, te permite tener un calendario compartido y disponer de listas comunes con bastante facilidad. Como aspectos negativos, el diseño es algo

anticuado y confuso, y le faltan algunas funcionalidades, por ejemplo, no permite asignar tareas de limpieza, o indicar qué comida tomarás hoy.

- **Dommus**: Una opción sencilla, aunque con pocas funcionalidades. Dispone fundamentalmente de listas y calendario. Carece de gestión económica, y de tareas del hogar, y tampoco dispone de recetario.

Por otro lado, tenemos un conjunto de aplicaciones que cubren parte de la funcionalidad que buscamos, algunas con mucho detalle. Buenos ejemplos de estas serían:

- **Splitwise**: Es un claro ejemplo de una buena aplicación para la gestión económica. Te permite anotar tus gastos, así como repartirlos entre los diferentes compañeros de piso. Lamentablemente, solo se orienta al aspecto económico, pero tiene un diseño muy logrado, es rápida y te ofrece grandes oportunidades de personalización.
- **ChoreMonster**: Orientada a las tareas del hogar, especialmente para los más pequeños, es un ejemplo de aplicación sencilla y divertida que busca encontrar la motivación para la realización de tareas de la casa.

1.2 OBJETIVOS DEL TRABAJO

Partiendo de las necesidades anterior, delimitamos los objetivos de nuestro proyecto, indicando un objetivo general a partir del cual desgranaremos un conjunto de objetivos específicos.

Los objetivos de este trabajo de fin de grado, comprenderían los siguientes apartados:

Objetivo general

OG1. Implementar una aplicación móvil para Android que facilite la gestión y organización de las familias o compañeros de piso. Para ello, la aplicación te permitirá crear un hogar e invitar a sus integrantes.

Objetivos específicos de la app

OE1. Proporcionar un calendario común sincronizado con Google. En este calendario también se verán las comidas y tareas planificadas (si las hubiera).

OE2. Permitir la creación de listas de la compra visibles para todos los usuarios del hogar. Estas listas deben poder organizarse por categorías.

OE3. Facilitar la planificación de comidas, pudiendo seleccionar, para cada día, la comida que se realizará.

OE4. Ofrecer al usuario la posibilidad de planificar las tareas de limpieza en el hogar, permitiendo añadir tareas pendientes, asignarlas, y programarlas para que reaparezcan en el futuro.

OE5. Realizar un sistema de recompensas, de forma que cada vez que un usuario realiza una tarea, reciba un número de puntos. Estos puntos deben poder canjearse por recompensas, previamente definidas por los usuarios miembros de la familia.

OE6. Proporcionar al usuario un apartado de gastos, donde las familias puedan apuntar los gastos comunes. Este apartado, debe permitir a los compañeros de piso dividir estos gastos, permitiendo saber en todo momento cuánto debes al hogar y cuánto te deben pagar a ti.

OE7. Diseñar un servicio REST que permita almacenar los datos de los usuarios necesarios y permita disponer de una app fluida.

Como objetivos secundarios, de carácter personal serían:

OS1. Poder poner en práctica los principios y características de *Material Design*.

OS2. Aprender acerca del funcionamiento de los calendarios de Google.

OS3. Realizar una app empleando las guías de código limpio, proporcionando documentación e intentando proporcionar un código genérico y modular, con el fin de facilitar la inclusión de nuevas características en el futuro.

OS4. Desarrollar una app estable y poco pesada, que tenga un diseño atractivo y fácil de usar.

1.3 ENFOQUE Y MÉTODO SEGUIDO

A nivel de enfoque, se ha decidido desarrollar la app sin emplear gestor de contenidos y sin partir de proyectos de Open Source ya comenzados. De este modo, podremos tener

un mayor control sobre la app producida. Esto nos va a permitir optimizar procesos y realizar un producto orientado exclusivamente a nuestros objetivos.

Respecto a la metodología de trabajo, ya que disponemos de una fecha de entrega próxima y que no se trata de un proyecto estructurado, llevaremos a cabo un desarrollo ágil. No obstante, al tratarse de un equipo unipersonal, es difícil emplear una metodología concreta como Scrum o Kanban, por lo que sencillamente, se aplicará una metodología a base de *Sprints*, empleando los tableros de Trello como soporte, a fin de gestionar las tareas pendientes.

A nivel de control de versiones, se usará Git con los repositorios gratuitos de BitBucket, pese a ser una única persona, con el fin de evitar pérdidas o cambios, y de mantener un seguimiento del proyecto.

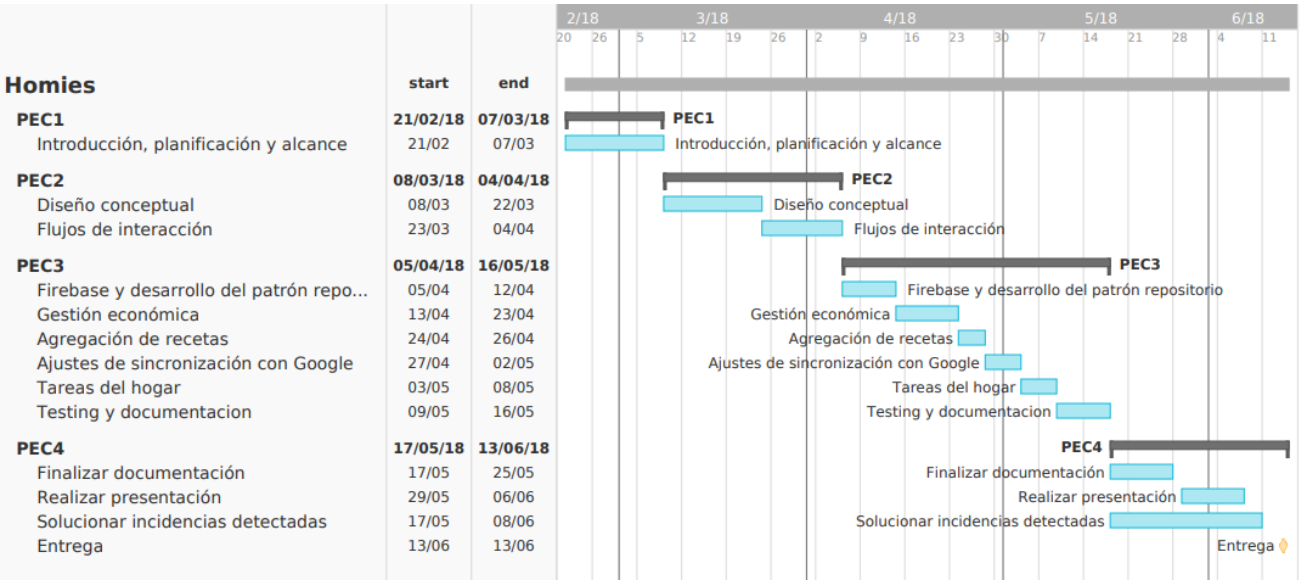
1.4 PLANIFICACIÓN DEL TRABAJO

A la hora de realizar este proyecto, no parte desde cero, ya que se comenzó en el primer cuatrimestre del curso lectivo 2017-2018. Los aspectos ya realizados del trabajo quedarían reflejados en la siguiente tabla:

Tarea	Porcentaje realizado
Decisiones de arquitectura	100%
Prototipo de la APP	100%
Casos de uso	100%
Análisis y definiciones funcionales de la app	100%
Diseño conceptual (necesita mejoras)	40%
Flujos de interacción (necesita mejoras)	30%
Desarrollo	
Conexiones a Firebase	100%
Login y registro	100%
Desarrollo de la BBDD Firebase	50%
Desarrollo del patrón repositorio	70%
Listas generales y listas de la compra	100%
Homepage	100%
Sistemas de puntos	70%
Tareas	80%
Gestión económica	20%
Planificación de comidas	80%

Agregación de recetas	30%
Calendario y sincronización con Google	70%

Por ello, se realiza una planificación temporal teniendo en cuenta que algunos aspectos ya se encuentran realizados. Así, dispondríamos del siguiente diagrama de Gantt:



Para la realización del diagrama, se ha tenido en cuenta que, a nivel de trabajo diario, es esperable poder dedicar entre 20-25 horas semanales al proyecto.

1.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS

Los productos proporcionados serán:

1. Copia del proyecto en Android, incluyendo el código fuente de la aplicación, así como todos sus archivos.
2. APK que permite instalar la app en un dispositivo Android.

Además, se entregará este documento como la memoria del proyecto, así como una presentación explicando la app y su desarrollo.

1.6 BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA

La memoria se organiza en capítulos. A continuación, se detallan los apartados creados y su relación con el proyecto:

1. **Introducción**: Sería el capítulo que acaba de concluir y permite disponer de una visión general del proyecto, su motivación y los objetivos que persigue.
2. **Análisis:** Se trataría de un análisis funcional de la app a desarrollar, indicando los diferentes *stakeholers* y las funcionalidades necesarias a un nivel más bajo. Los diagramas de UML que se realicen, así como los análisis de la base de datos, irían en este apartado.
3. **Arquitectura:** Se centra en explicar la tecnología aplicada. Detallado el IDE empleado, base de datos, así como versión de Android sobre la que trabajamos.
4. **Diseño gráfico**: Donde se incluiría el *mock up*, el diseño de la interfaz de usuario, logotipos, así como tipografía y colores escogidos.
5. **Conclusiones**: Englobarían las contingencias del proyecto, las dificultades encontradas en el desarrollo y las diferentes posibilidades de ampliación o aspectos en los que la APP podría mejorar.
6. **Bibliografía**: Influirá las referencias empleadas para el desarrollo del proyecto. Empleando el formato ISO 690.

2. ANÁLISIS

2.1 USUARIOS Y CONTEXTO DE USO

2.1.1 INTRODUCCIÓN AL DCU

El diseño centrado en usuario (DCU) es un enfoque a la hora de diseñar un software o aplicación. Se fundamenta en que el proceso de diseño debe basarse en la información proporcionada por las personas que van a hacer uso del sistema, es decir, los usuarios.

Uno de los mayores riesgos en el diseño de aplicaciones móviles es realizar un producto que sea poco usado o poco atractivo para los usuarios. Esto, sumado a la competitividad del mercado, supone que es necesario utilizar una metodología como DCU, con el fin de optimizar el resultado de cara al usuario y construir un software que se asegure de cubrir sus necesidades y adecuarse a sus gustos. De este modo, la captación de usuarios es más sencilla, la usabilidad es mayor y se garantiza que cubra, al menos, las necesidades principales de los usuarios.

El proceso se realiza en varias fases que mostraremos a continuación:

1. **Análisis**: Que busca identificar los contextos de uso, los usuarios y sus necesidades. En este documento, corresponde a este apartado.
2. **Diseño**: Englobaría los diferentes aspectos del diseño, tanto lógico como gráfico. En este documento, esta información se encuentra en el apartado Diseño.
3. **Evaluación**: Comprueba que se han alcanzado los objetivos y que el producto satisface a los usuarios. En este documento, se incluye en el apartado 3.6 Evaluación del diseño.

ILUSTRACIÓN 1. PROCESO DE DCU

2.1.2 MÉTODO DE INDAGACIÓN ESCOGIDO.

Existen numerosos métodos de indagación para obtener información de los diferentes usuarios potenciales de la app. No obstante, de cara a este proyecto hemos optado por utilizar un método cualitativo. En concreto, nos hemos decantado por las **entrevistas en profundidad** ya que nuestra aplicación es versátil y un método de indagación como éste abarca diversas dimensiones, permitiéndonos obtener información sobre las necesidades de los usuarios de manera amplia, directa y poco estricta.

2.1.3 ENTREVISTAS EN PROFUNDIDAD

Para la realización de las entrevistas en profundidad hemos realizado entrevistas a varias personas de nuestro entorno. Ya que nuestra app tiene un espectro de usuarios bastante amplio, hemos intentando escoger una muestra lo más uniforme posible. Para la entrevista, hemos decidido realizar un guion para facilitar la interacción con los entrevistados. No obstante, intentamos tener en cuenta que la conversación debe ser abierta, procurando evitar sesgos e influencias en las respuestas. De esta forma, estas preguntas se usan a modo de ejemplo, pero no se han realizado todas en todos los casos:

- Información general: Situación laboral y familiar, así como lugar de residencia.
- Frecuencia de uso de los dispositivos móviles. ¿Emplea la tecnología para la vida diaria?
- ¿Qué dificultades encuentra a la hora de gestionar su hogar?
- ¿Qué necesitaría en una app de gestión familiar?

- Explicación de la app. ¿Entiende para qué sirve? ¿Qué esperaría si se la acabara de instalar?
- ¿Qué aspectos positivos ve en la idea? ¿Qué funcionalidad echa en falta?
- ¿Usaría una app así? ¿Por qué sí? ¿Por qué no?

Hemos realizado 9 entrevistas. Incluimos a continuación un resumen de las tres más importantes.

> **Entrevista 1.** *Helena trabaja desde casa, tiene 26 años, vive en Salamanca y comparte piso. Emplea el móvil a diario y ha usado muchas apps para gestionar su hogar (listas de la compra, gestión económica, recetas, etc.). Considera indispensable tener un lugar donde gestionar las tareas del hogar para poder repartirlas entre sus compañeros, permitiendo monitorizar si se han realizado o no.*
>
> *Con el fin de no depender de otras apps, también quiere un lugar donde tener la lista de la compra actualizada, de forma que sus compañeros puedan verla. Cuando le explicamos las funcionalidades de la app, casi todas le parecen muy interesantes, aunque le preocupa que la funcionalidad de las recetas no sea intuitiva.*
>
> **Entrevista 2.** *Ana tiene 38 años, tiene dos hijas y vive en Madrid. Con el ritmo de la ciudad le resulta difícil gestionar su hogar. Usa el móvil diariamente, aunque no ha probado aplicaciones de gestión del hogar. Una de sus mayores preocupaciones es la planificación de las comidas, al final ella siempre se encarga de cocinar, pero le cuesta decidirse y realizar un menú equilibrado. Quiere un lugar donde apuntar sus recetas y poder planificarlas para cada día de la semana. Por otro lado, le cuesta que los niños colaboren en la casa, le parece importante que haya una recompensa para las tareas del hogar, con puntos o premios de algún tipo. Por último, cuando le explicamos la idea de la app le gusta, aunque le parece que son demasiadas funcionalidades por lo que cree que sería necesario un diseño muy intuitivo o bien un apartado de ayuda.*

Entrevista 2. *Mario vive solo, tiene 24 años y se independizó hace poco. Le cuesta organizarse en sus tareas, usa apps que le ayudan a gestionarse con objetivos. Lo que le interesa de una app para la gestión del hogar es el apartado económico, algo que le permita anotar sus gastos y registrar su economía. También le interesan las listas de la compra (pero muy sencillas) y la gestión de las tareas de limpieza, para que pueda organizarse.*

A partir de las respuestas obtenidas, hemos elaborado las siguientes conclusiones:

a) Casi todos los usuarios encuentran que la app tiene muchas funcionalidades, así que necesitarían o bien un tutorial o un diseño muy intuitivo. También conviene plantearse la simplificación de algunos de los objetivos de la app.
b) En general, muchos usuarios emplean las listas para organizarse. Las capacidades tecnológicas varían. La mayoría no ha usado ninguna app similar, aunque todos emplean los móviles en su vida diaria. Todos conocen Google Calendar y lo usan habitualmente.
c) Los principales usuarios son padres/madres o compañeros de piso, no obstante, también las unidades familiares de un único miembro podrían querer usarla.
d) La mayor parte de los usuarios tienen necesidades similares: planificar comidas, asignar tareas de limpieza y obtener recordatorios de los mismas parecen funcionalidades primordiales.

2.1.4 Fichas de usuario

A partir de los resultados de las entrevistas, hemos elaborado los siguientes perfiles de usuario:

Compañeros de piso
Experiencia en tecnología: Alta. **Funcionalidades más importantes**: Lista de la compra, tareas de limpieza.
Escenarios de uso: ***Escenario 1***

Contexto: *Está saliendo del trabajo y ve que tiene un supermercado al lado y que le sobran 5 minutos, así que quiere entrar y comprar algunas cosas. Las cosas que compra quiere marcarlas en la lista para que sus compañeros sepan que ya se han comprado.*

Objetivo: Ver la lista de la compra y marcar los elementos ya comprados.

Escenario 2

Contexto: *Es domingo, y quiere saber qué tareas de limpieza le toca a cada miembro de la casa la semana que viene.*

Objetivo: Comprobar las tareas de limpieza en la app. Asignarlas si es necesario. Ver si están o no realizadas.

Madre/Padre de familia.

Experiencia en tecnología: Usa el móvil a diario.

Funcionalidades más importantes: Recetario, planificación de comidas, reparto de tareas.

Escenario 1

Contexto: *Quiere llegar a mediodía, y poder mirar en la app lo que toca comer hoy, para tener las comidas planificadas de antemano.*

Objetivo: Disponer de un calendario con las comidas que se deben preparar cada día.

Escenario 2

Contexto: *Los niños no suelen querer realizar las tareas del hogar, quiere convencerles de que colaboren con la casa asignándoles tareas y premiándoles con beneficios.*

Objetivo: Asignar tareas del hogar a otros usuarios, asignar puntos a cada tarea y poder canjear los puntos por recompensas. Enseñar a los niños en la app los puntos que lleva y los premios que puede conseguir en función de cuánto colabore.

Escenario 3

Contexto: *Está intentando planificar las comidas de esta semana, pero así a botepronto es difícil que se acuerde de todas las cosas que sabe cocinar.*

Objetivo: Entrar en la app y ver su listado de recetas.

Miembro unifamiliar

Experiencia en tecnología: Usa el móvil a diario.

Funcionalidades necesarias: Planificación económica, planificación de las tareas del hogar.

Escenario 1

Contexto: *Quiere limpiar el baño con cierta regularidad, pero siempre se le olvida.*

Objetivo: Crear una tarea que sea limpiar el baño y que se sincronice con su calendario. Recibir algún tipo de penalización o fallo si no realiza la tarea.

Escenario 2

Contexto: *Ya no se acuerda de cuánto dinero ha gastado este mes. Lo ha ido incluyendo en la app, pero querría saber si ha gastado mucho o poco.*

Objetivo: Entrar en la app y ver cuánto dinero lleva gastado y si es más o menos que otros meses.

2. 1. 5 CONCLUSIONES DEL ANÁLISIS

A través de las fichas, podemos concluir que nuestros usuarios tienen un rango de edad diferente, que el uso de la app será realizado fundamentalmente por personas adultas, especialmente por personas con núcleos familiares de dos o más personas. El contexto de uso será principalmente en el hogar, si bien se usará la app en modo consulta fuera del mismo para interactuar con las listas de la compra o consultar el calendario.

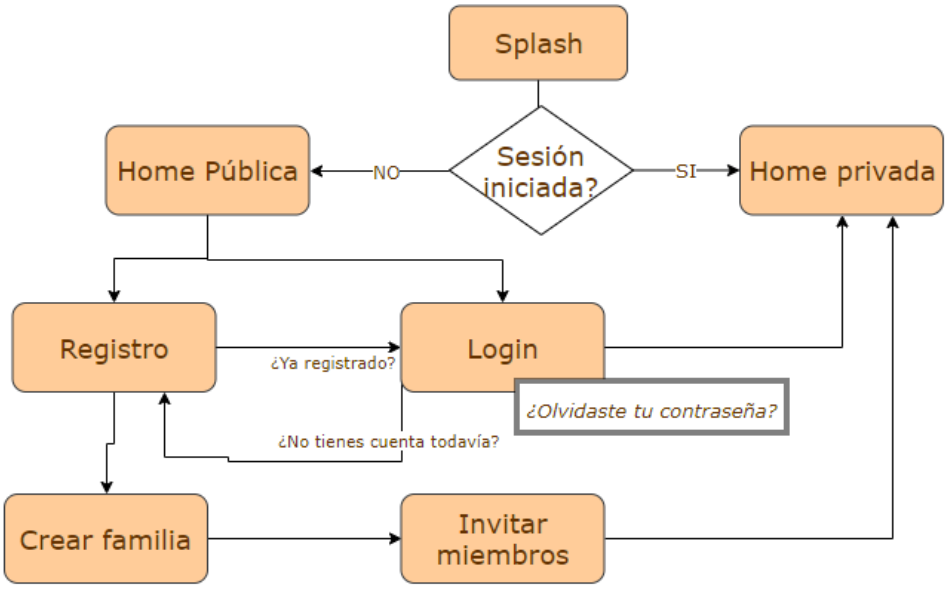
Tras las entrevistas, hemos añadido algunas funcionalidades importantes ya mentadas, como los resúmenes económicos, el recetario o la asignación aleatoria y programada de las tareas. También hemos visto a través de los perfiles de usuario que la lista de la

compra y la planificación de comidas parecen las funcionalidades más importantes, por lo que haremos hincapié en ellas en el diseño.

3. DISEÑO

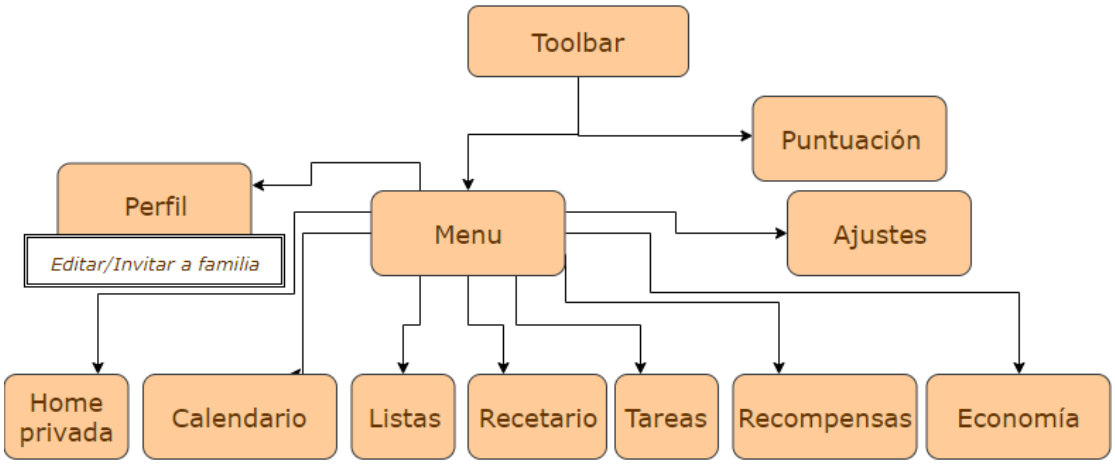
3.1 FLUJOS DE INTERACCIÓN.

Para los flujos de interacción, se ha optado por un diseño en el que cada elemento representa una pantalla. Los condicionales se marcan con un rombo cuando forman parte de la lógica interna y las operaciones disponibles en la misma pantalla se representan con un rectángulo en blanco.



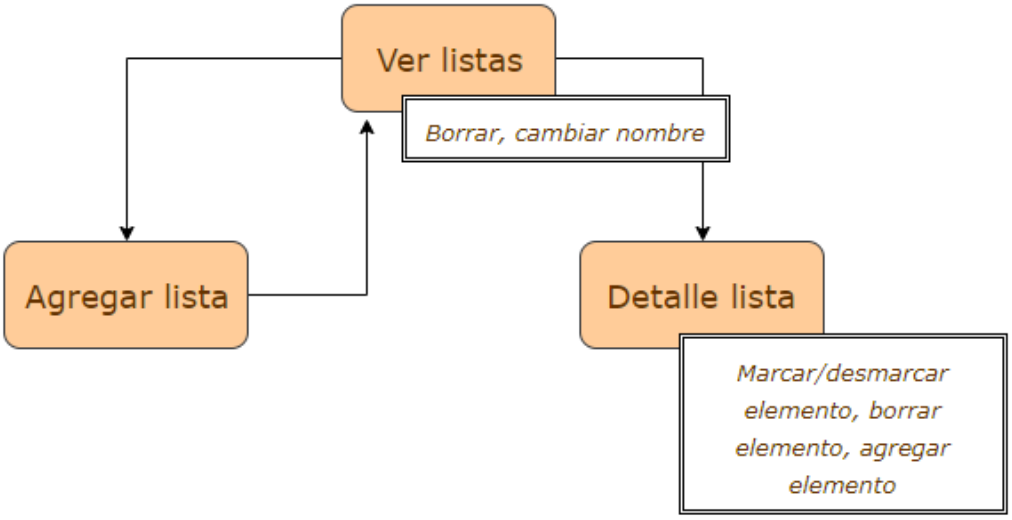
En este flujo, las interacciones entre registro y *login* se representan, en vez de con un rombo, como una relación, ya que el paso de una sección a otra se realizará con botones accionados por el usuario. Del mismo modo, la acción *Olvidaste tu contraseña* se representa como una acción ya que no tendrá una pantalla independiente, y no altera el flujo.

Es importante tener en cuenta que, desde las pantallas principales, una vez logado, el botón de atrás siempre te lleva a la *Home Privada*. Por otro lado, el menú siempre estará presente a través de la *toolbar* (Barra superior) de Android. El flujo de la *toolbar* y el menú sería el siguiente:

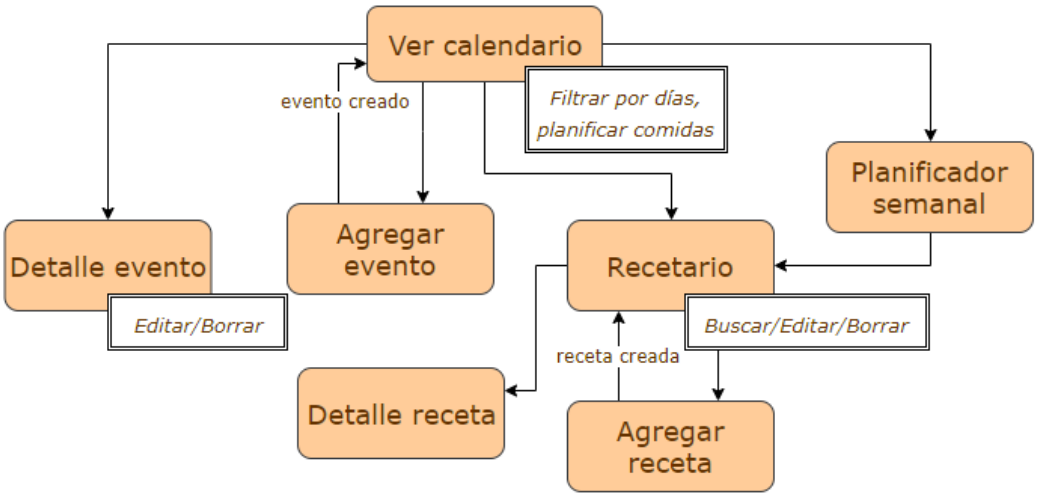


A continuación, se muestran los flujos de cada pantalla principal.

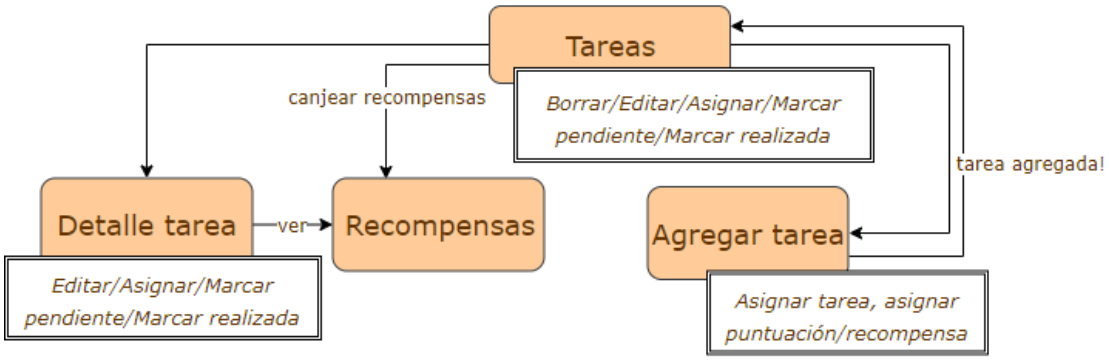
3.2 Flujos de interacción de Listas.



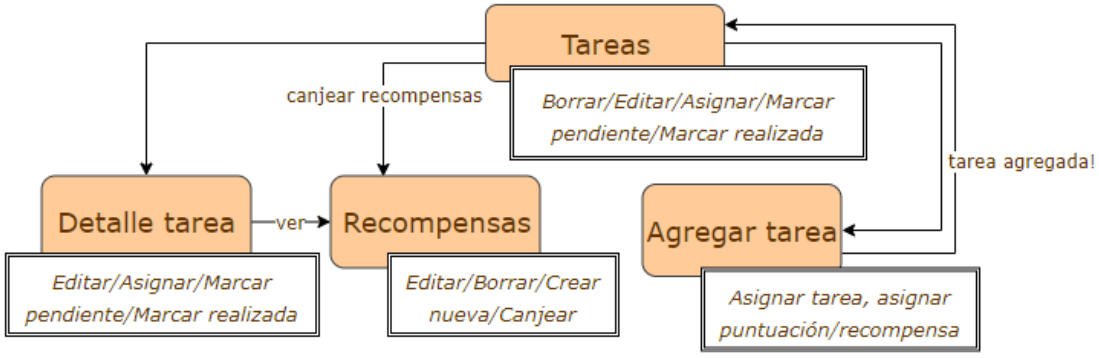
3.2 Flujos de interacción de Eventos.



3.3 Flujos de interacción de tareas.

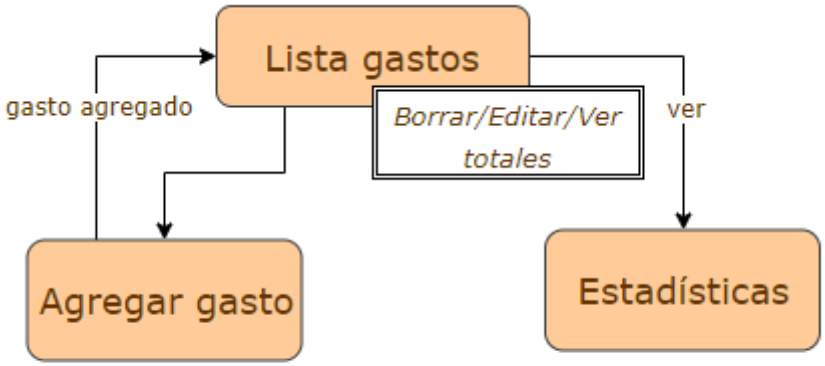


3.4 Flujos de Recompensas.



Las recompensas se crearán en la misma pantalla para evitar un flujo demasiado grande.

3.5 Flujos de Economía.



Es importante notar que, por un lado, no será posible editar o agregar estadísticas, ya que esta pantalla no será personalizable. Por otro lado, no habrá detalle de los gastos, toda la información se verá a través de la lista.

3.2 DISEÑO TÉCNICO

3.2.1 DIAGRAMA DE LOS CASOS DE USO

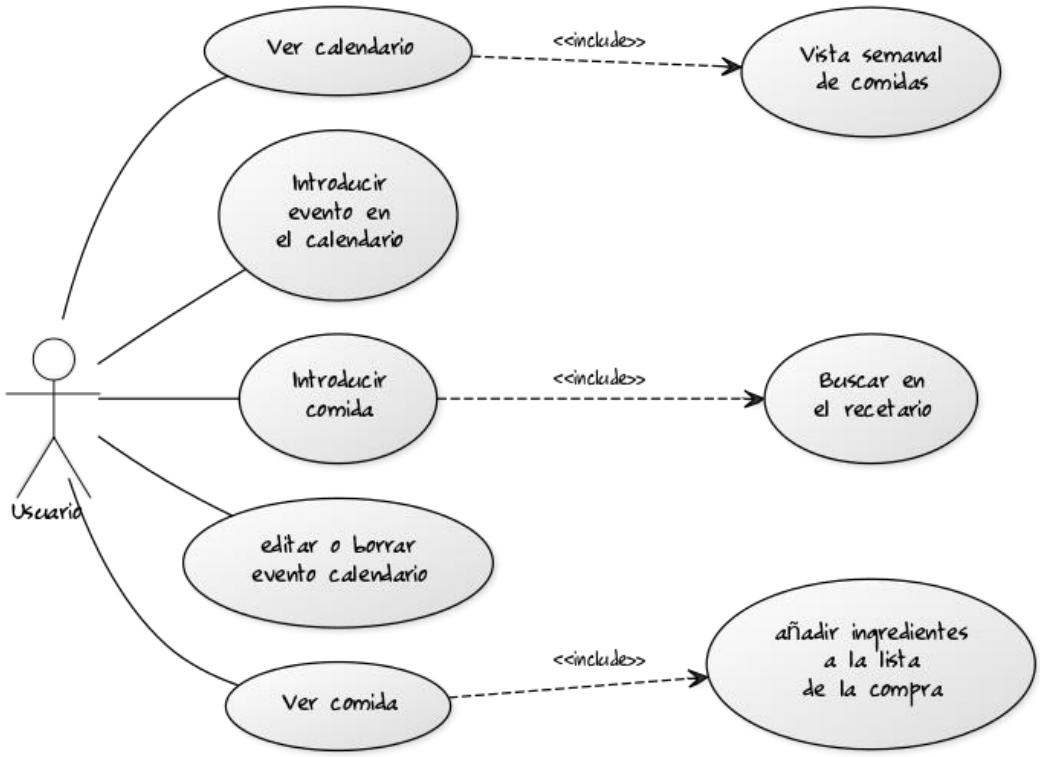


ILUSTRACIÓN 2. DIAGRAMA DE CASOS DE USO DEL CALENDARIO.

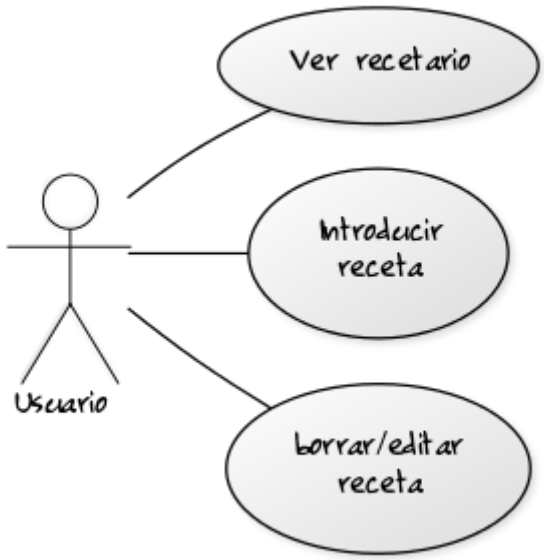


ILUSTRACIÓN 3. DIAGRAMA DE CASOS DE USO RECETARIO

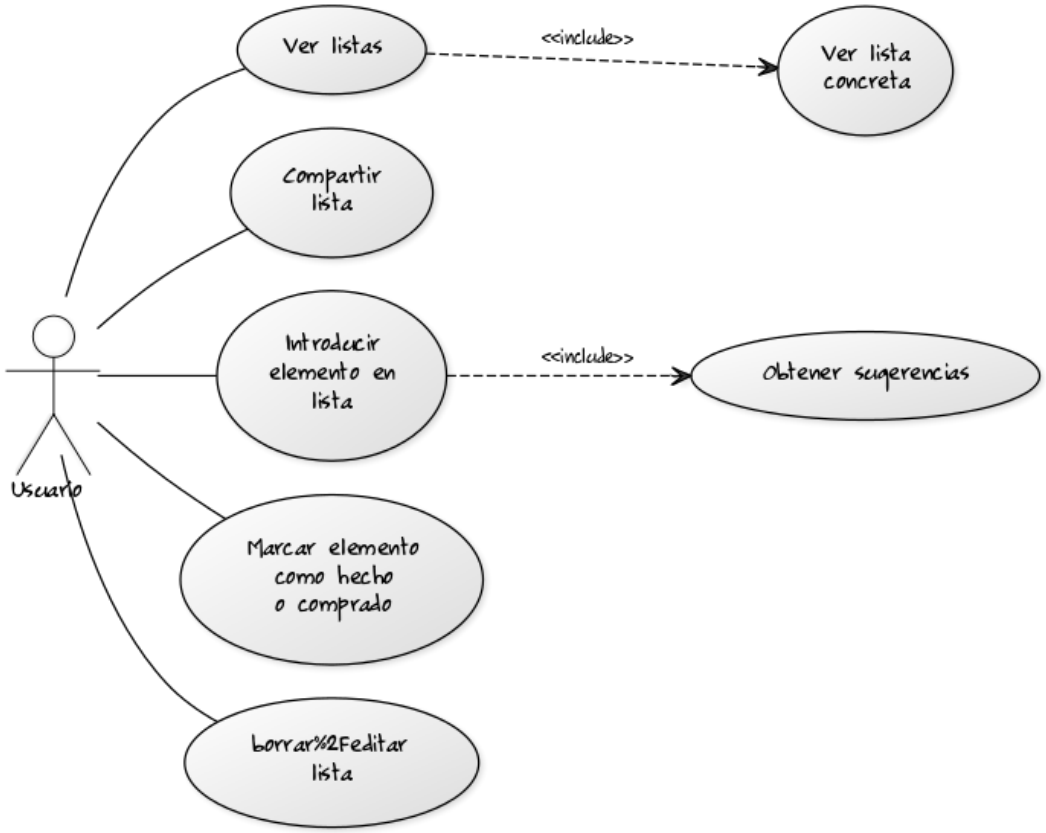


ILUSTRACIÓN 4. DIAGRAMA DE CASOS DE USO DE LISTAS

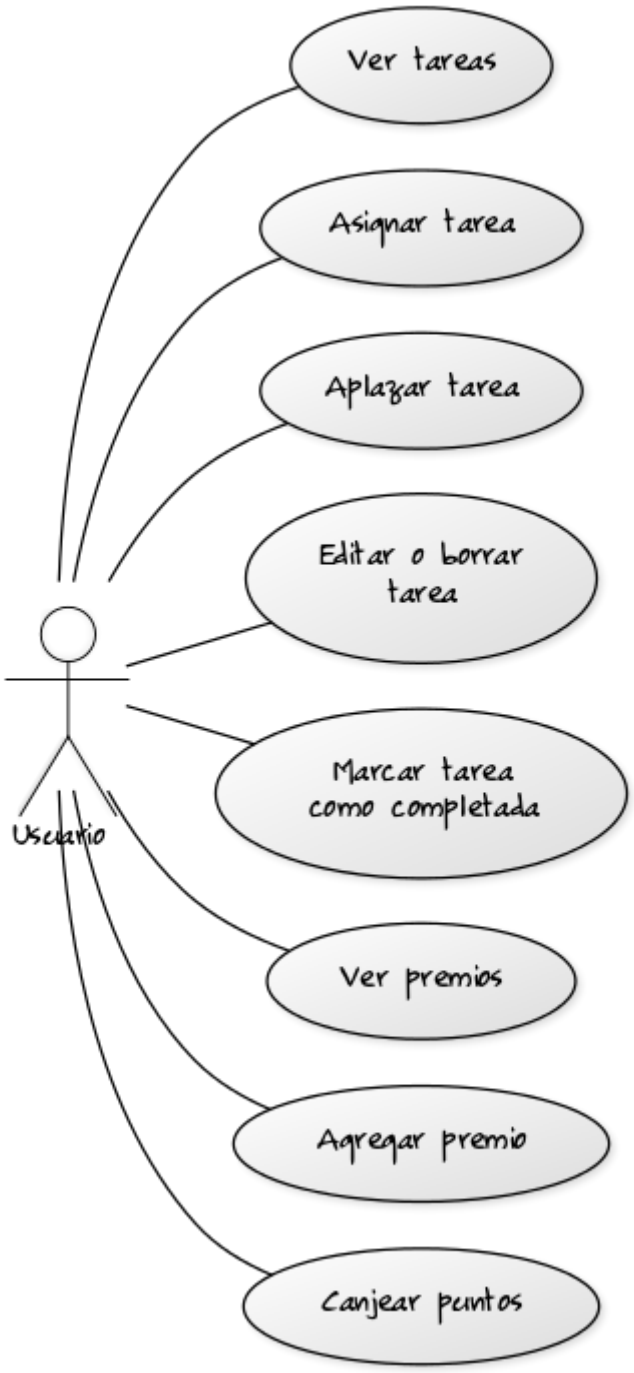


ILUSTRACIÓN 5. DIAGRAMA CASOS DE USO DE TAREAS Y PREMIOS

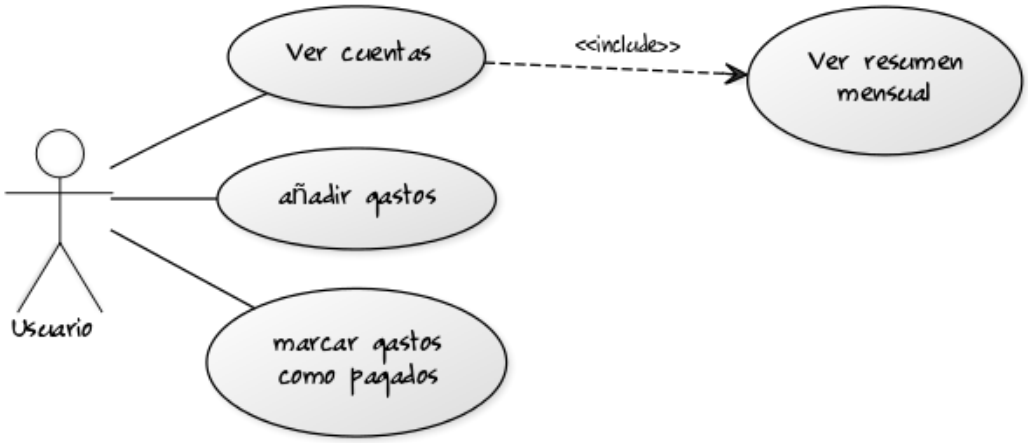


ILUSTRACIÓN 6. DIAGRAMA DE CASOS DE USO DE GASTOS

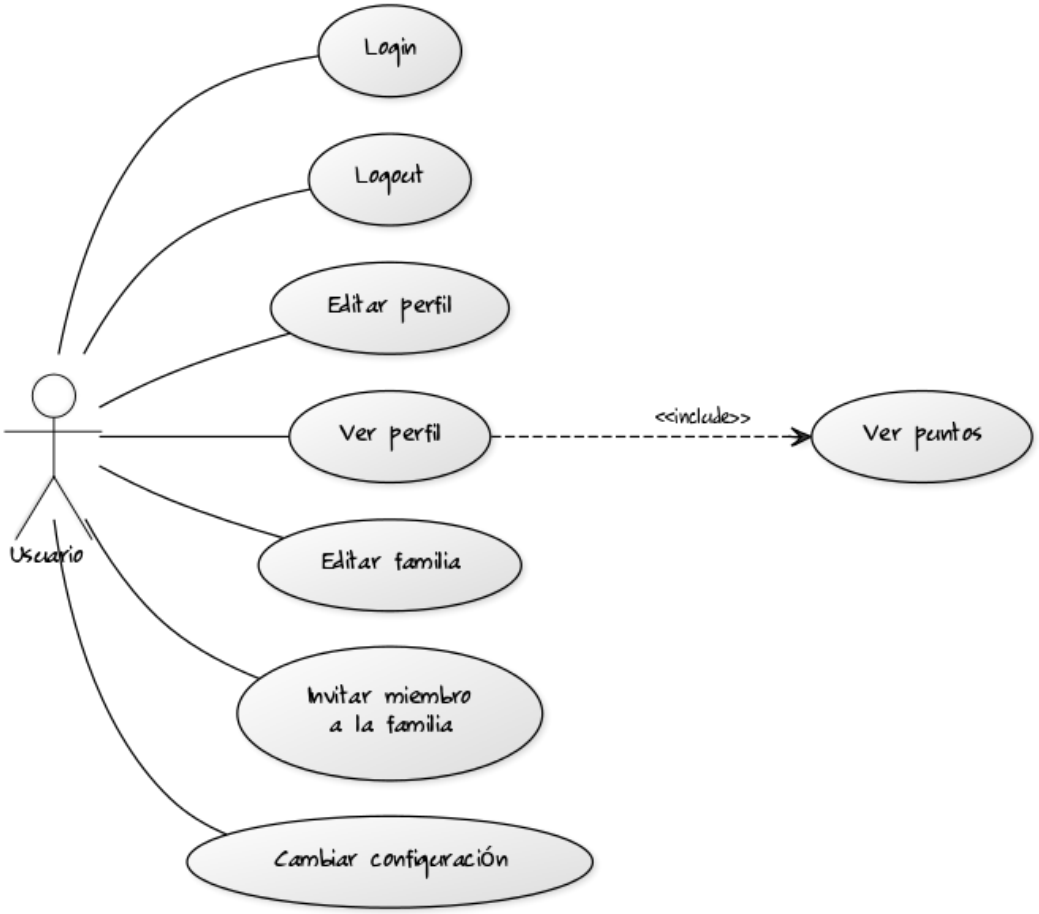


ILUSTRACIÓN 7. DIAGRAMA DE CASOS DE USO GENERALES

2.2 ELECCIÓN DEL LOGO.

Para el logo, se ha decidido utilizar un diseño redondeado con un icono central en fondo blanco, acorde a los diseños de *Material Design*.

ILUSTRACIÓN 8. LOGO ESCOGIDO

2.3 GAMA DE COLORES.

La gama de colores empleada puede visualizarse en la siguiente imagen

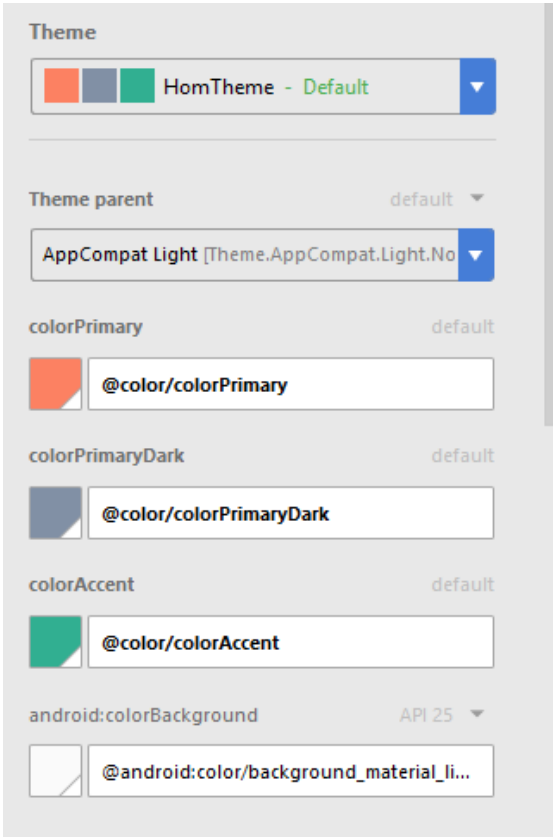


ILUSTRACIÓN 9. COLORES DE LA APP

Cabe destacar que nuestro color primario, el #fc8162 *Deep Orange*, tiene una serie de variaciones que se muestran a continuación y se usan en la app.

ILUSTRACIÓN 10. DEEP ORANGE Y DERIVADOS

3.4 TIPOGRAFÍA.

Para la tipografía se ha mantenido la fuente predeterminada en Android, a excepción del título del logo, que se ha adjuntado siempre como imagen para garantizar su correcta lectura.

Homies

ILUSTRACIÓN 11. TIPOGRAFÍA DEL NOMBRE DE LA APP

3.4 SKETCH.

Para el diseño de la app hemos realizado unos bocetos a mano alzada de las primeras pantallas. La intención es representar de forma simplificada un primer diseño de la aplicación.

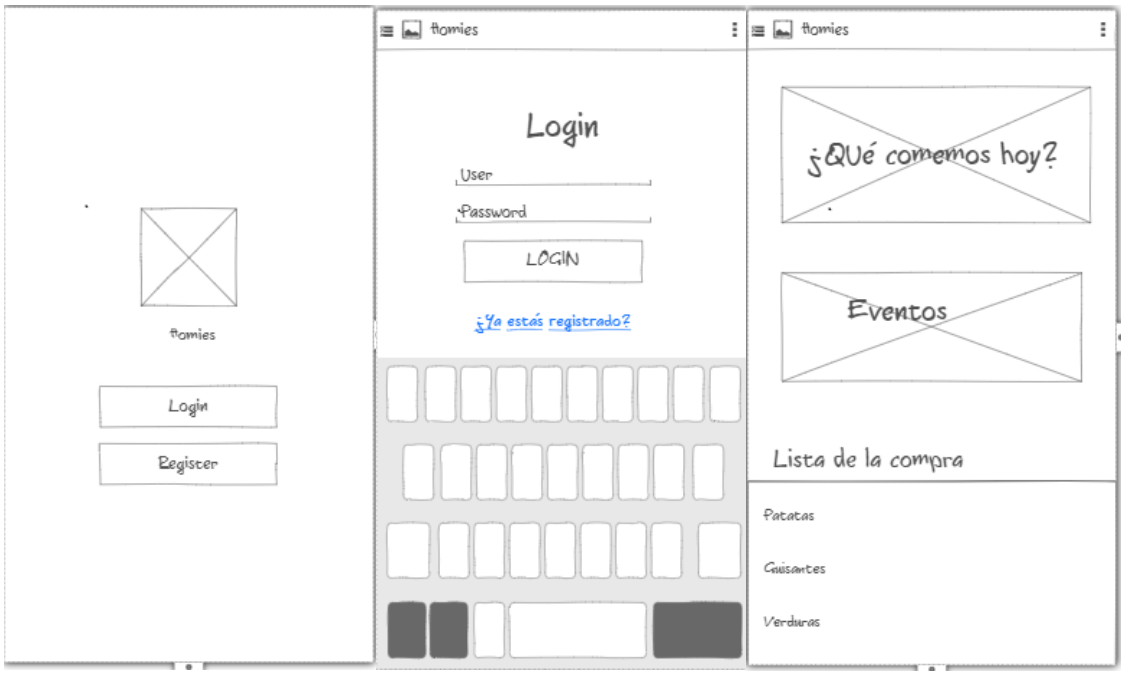


ILUSTRACIÓN 12. ESBOZOS DE: HOME PUBLICA, LOGIN Y HOME PRIVADA.

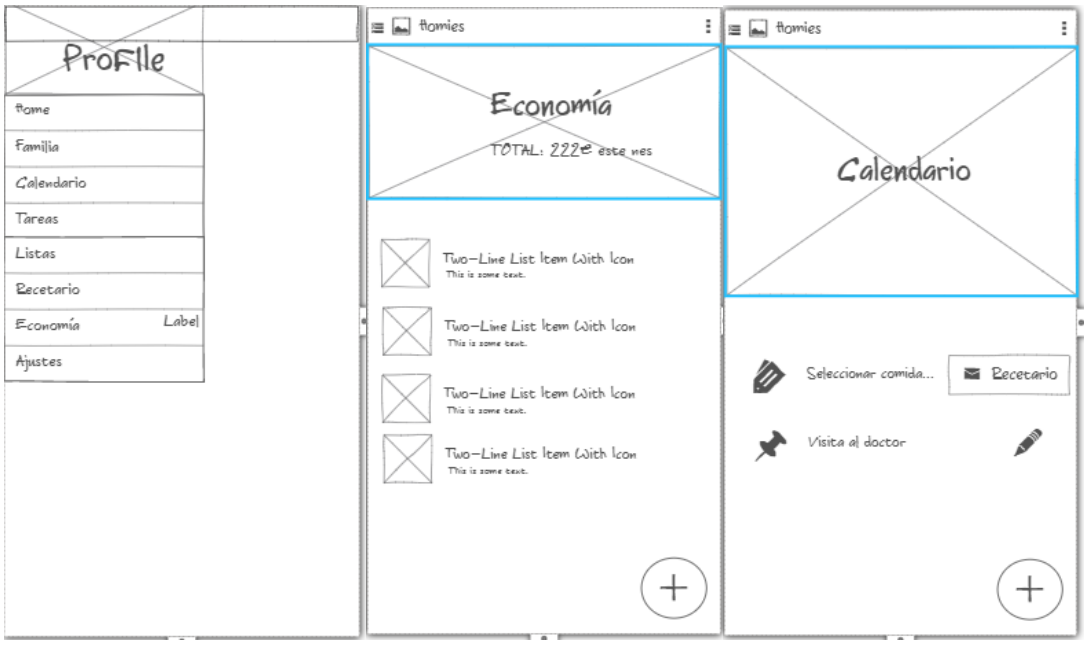


ILUSTRACIÓN 13. ESBOZOS DE PANTALLAS: MENÚ, ECONOMÍA Y CALENDARIO

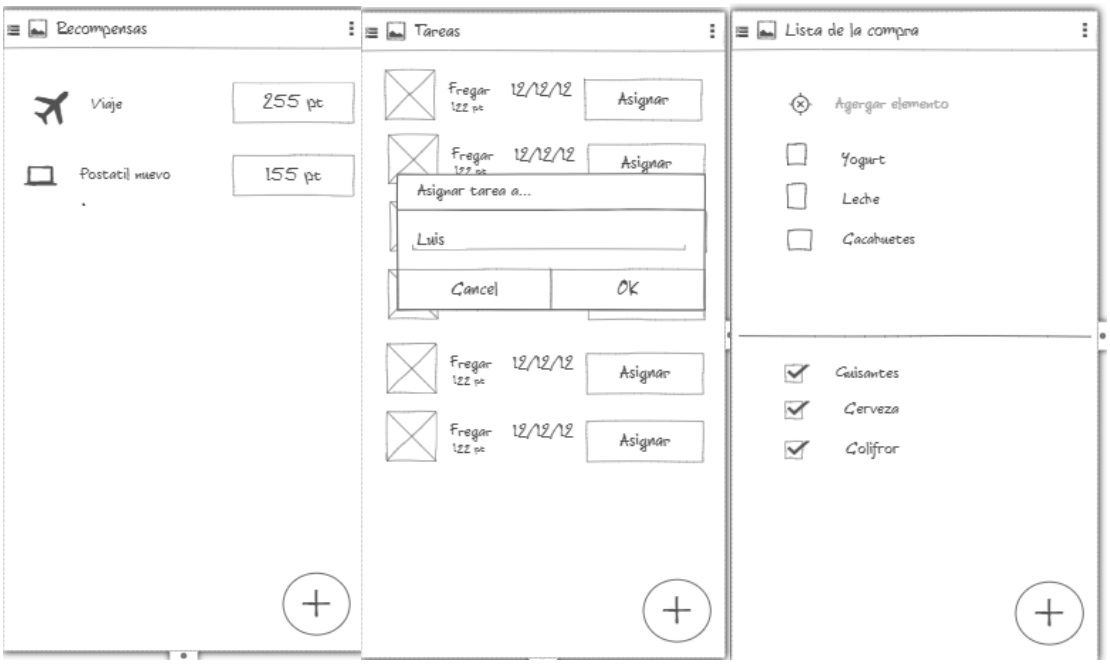


ILUSTRACIÓN 14. ESBOZOS DE PANTALLAS: RECOMPENSAS, TAREAS Y DETALLE LISTA.

3.5 PROTOTIPO HORIZONTAL.

Para la realización del prototipo, hemos trabajado directamente con las herramientas proporcionadas por el software *Marvel app*. A continuación indicamos un enlace donde puede verse y descargarse el prototipo horizontal junto con los diferentes flujos de acción: https://marvelapp.com/jj78ba6

3.6 EVALUACIÓN DEL DISEÑO

Tras haber realizado el prototipo vendría la fase de evaluación. El objetivo sería realizar este proceso de forma iterativa para ir perfeccionando el diseño. Para ello, lo idílico sería trabajar con un conjunto heterogéneo de usuarios que pudieran probar el producto y proporcionar información sobre el mismo.

En las sesiones con los usuarios necesitaríamos obtener información sobre las cuestiones de diseño y las de funcionalidad. Para ello, deberíamos disponer de un conjunto de preguntas que abarcaran todos los campos. Se muestra, a continuación, un listado de ejemplo:

- ¿El diseño de la app parecía apropiado con la temática?

- ¿Hay algún texto en la app que haya resultado difícil de leer?
- ¿Ha resultado comprensible todo lo que puede hacer con la app? ¿Hay algún aspecto que no haya quedado claro?
- Al entrar en la pantalla principal, ¿Se comprende la funcionalidad de la aplicación?
- ¿Le ha resultado fácil invitar a los usuarios de su familia?
- ¿Ha encontrado algún defecto o error?
- ¿Echa en falta alguna funcionalidad?
- Comentarios, críticas y sugerencias.

Las tareas que los usuarios deben realizar, así como el conjunto de preguntas específico de cada sección serían los siguientes:

- Tarea: Agregar una receta.
  - ¿Ha encontrado alguna dificultad al crear la receta?
  - ¿Le ha resultado incómodo incluir los datos?
- Tarea: Planificar una comida.
  - ¿Ha sabido dónde acudir para planificar las comidas?
- Tarea: Visualizar evento.
  - ¿Le ha resultado intuitivo el calendario?
  - ¿Ha echado de menos algún campo en el detalle del evento?
  - ¿Se ha dado cuenta de que había diferentes iconos para los eventos? ¿Ha comprendido cada tipo de evento?
- Tarea: Crear tarea y asignársela a uno mismo.
  - ¿Ha comprendido todos los campos de creación de una tarea?
  - ¿Ha echado de menos algún campo?
  - ¿Le ha resultado fácil asignar la tarea?
  - ¿Ha comprendido cómo funcionan los puntos?
- Tarea: Crear una recompensa
  - ¿Ha entendido todos los campos de la creación de una recompensa?
  - ¿Le ha resultado intuitivo el concepto de recompensa?
  - ¿Comprende su funcionamiento?
  - ¿Cree que esta sección es necesaria?

La idea general sería repetir este proceso hasta que la mayor parte de los usuarios se sientan satisfechos con el resultado y se sientan cómodos con la app.

3.7 PLAN DE CONTINGENCIAS

3.7.2 PLANIFICACIÓN (PRODUCTO MÍNIMO VIABLE)

Antes de llevar a cabo el desarrollo, hemos decidido realizar un plan de contingencias, que nos permita actuar en caso de que se produzca algún imprevisto. En concreto, tras analizar la app, vemos que su mayor problema pueda estar ligado a la funcionalidad, ya que hemos delimitado una aplicación muy grande, lo que aumenta su complejidad.

Por ello, decidimos que se establecerá un PMV diferenciando aquellas funcionalidades que deben formar parte del producto mínimo asociado a la primera entrega, y aquellas que, si bien es mejor que estén, son susceptibles de ser aplazadas en caso de que se produzcan contingencias en el proyecto.

Una vez definido el PMV, podemos afrontar los dos tipos de contingencia que podemos encontrar. Serían los siguientes:

- ***Contingencia temporal - ¿Problemas con la temporización?*** Será la contingencia más probable, y se produce en caso de que veamos que no estamos cumpliendo con la planificación o que hemos encontrado problemas de cara afrontarla. En este caso, la solución pasará por revisar la planificación, y quitar aquellas tareas planificadas que no formen parte del PMV o aplazarlas.
- ***¿Tareas demasiado complejas?*** Es normal a la hora de realizar un desarrollo unipersonal encontrar una tarea demasiado compleja o encontrarse atascado. En ese caso, comprobaremos si la tarea puede realizarse y el bloqueo es temporal o puede resolverse con apoyo externo (profesorado, formación puntual, etc.). Esto nos llevará a dos situaciones:
    - La tarea puede realizarse, pero requiere más tiempo. En este caso, supone una contingencia temporal, por lo que deberemos revisar la planificación y si es necesario, quitar funcionalidad de entre aquellas tareas que no formen parte del PMV.

- La tarea no puede realizarse. Si encontramos que la tarea, sencillamente es demasiado compleja o el tiempo que ocuparía es desproporcionado, intentaremos simplificarla, de forma que podamos proporcionar alguna funcionalidad similar, aunque sea de forma más sencilla. En ese caso, simplificaremos la tarea, y luego volveremos a revisar la planificación de cara a la modificación planteada.

Este proceso queda resumido en el siguiente diagrama:

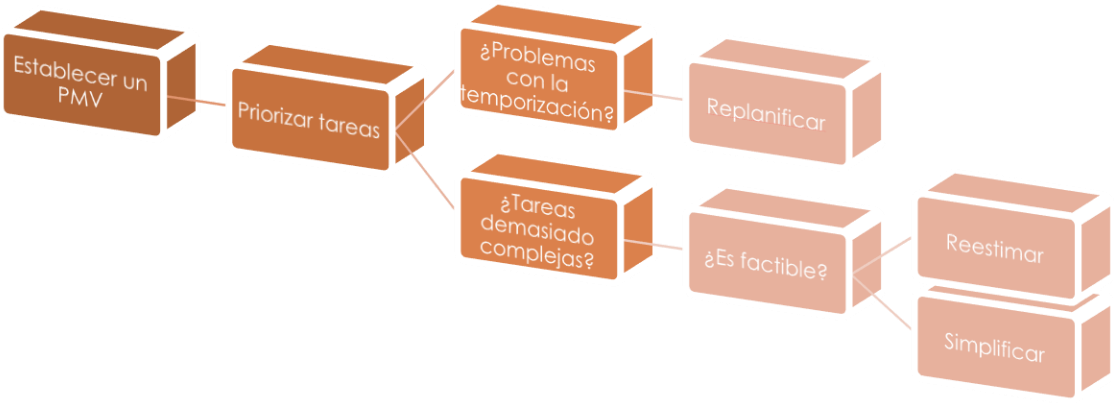


ILUSTRACIÓN 15. PLAN DE CONTINGENCIAS

3.7.1 PMV (PRODUCTO MÍNIMO VIABLE)

Para definir un PMV, nos centramos en los objetivos iniciales (Producto Mínimo Viable) y los casos de uso. La idea general es identificar las funciones mínimas que debe tener la aplicación. Esto no significa que no vayamos a realizar el resto de requisitos, pero nos permite identificar que tareas son susceptibles de aplazarse en caso de que la estimación temporal no encaje.

Las áreas generales que podemos identificar en la aplicación y sus objetivos mínimos serían los que muestra los siguientes:

- **Acceso**: Que incluiría el Login, Registro y Creación de familia.
  - Decidimos que, aunque interesante: no es importante el envío de e-mails, ni para invitar, ni para crear un usuario.

    - La subida de fotos para los usuarios o para la familia queda para post-PMV.
    - El formato de “etiquetas” para invitar usuarios, es interesante, pero no obligatorio.
- **Listas**: La app deberá permitir tener listas disponibles para toda la familia, y permitirá su creación y edición. Las listas deben tener una categoría. Deben poder eliminarse y marcar sus elementos como “*hechos*” o “*terminados*”. Dado que esta historia de usuario fue identificada como importante, todos sus requisitos deben pertenecer al producto mínimo.
- **Tareas del hogar**: la app deber permitir a un usuario crear tareas, asignar tareas, y marcar tareas como finalizadas.
    - Asignar tareas desde la propia lista sería mucho más cómodo, así como asignar soporte gestual para poder marcar tareas como hechas, por ejemplo, deslizando un elemento a la derecha o a la izquierda.
    - Tras el producto mínimo, sería positivo poder repetir las tareas de forma periódica, así como obtener estadísticas de las mismas.
- **Calendario**: la app debe sincronizarse con Google Calendar y permitir visualizar los eventos del calendario, así como guardar los eventos en el calendario de Google. Por tanto, al entrar en el calendario deberá solicitar al usuario permisos para acceder. En el calendario tendrán que mostrarse las tareas y las comidas asignadas.
    - La vista semanal y la asignación de recetas podrían aplazarse, ya que requieren de pruebas con el usuario.
- **Recetario**: Debe permitir crear recetas, asignándoles descripción y pasos a realizar.
    - Las recetas podrían marcarse como favoritas, para saber cuáles te gustan o te gustaron más.
    - Las recetas podrían tener una foto. Queda fuera del producto mínimo ya que debe analizarse el peso que tendrá en nuestra base de datos el almacenamiento de imágenes.

- La asignación de una receta a un día puede quedar fuera del producto mínimo, ya que los usuarios pueden frustrarse al intentar poner varias recetas y la conexión entre receta-calendario resultaba preocupante para los usuarios en el análisis.

- **Economía**: Debe permitir repartir los gastos entre los usuarios y visualizar todos los gastos hasta el momento.
  - Si bien será necesario un apartado de estadísticas mínimo, muchas estadísticas tales como los gastos por categoría, o las predicciones de gastos quedarían fuera del producto mínimo.

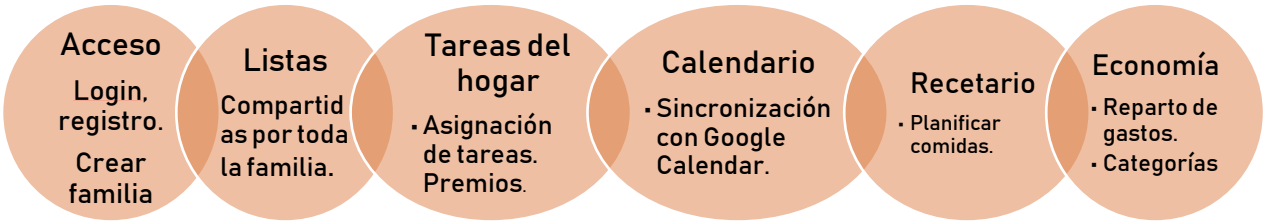


ILUSTRACIÓN 16. FUNCIONALIDAD PMV

4. ARQUITECTURA

4.1 TECNOLOGÍA EMPLEADA.

4.1.1 IDE Y GUÍAS DE ESTILO.

Para la realización del trabajo se ha optado por emplear la versión de Android Studio 2.3.1 y la versión de Java 1.7.

Para facilitar la realización del código se han introducido guías de estilo en el código e inspectores de código, que buscan mantener el código limpio, así como seguir una serie de buenas prácticas. Para la realización de esta guía de estilo, que puede descargarse del mismo proyecto, ha sido de utilidad el libro *Clean Code*, de Robert Cecil Martin, así como las guías de estilo de código de Android Studio y de raywenderlich, recuperables en https://source.android.com/setup/code-style y en https://github.com/raywenderlich/java-style-guide, respectivamente.

Las reglas más importantes de estos estilos de código son las siguientes:

- Orden en las clases:
    - Optimizar los *import* y ordenarlos.
    - Variables siempre al principio.
    - Los métodos privados al final, los métodos públicos al principio, preferentemente comentados, con los estándares de JavaDoc.
- Las variables, clases y métodos deben tener nombres en inglés y semánticos, teniendo en cuenta que:
    - No debe haber variables con menos de tres letras.
    - Las clases empezarán con mayúsculas, y las variables con minúscula.
    - Las variables finales irán con mayúscula y las estáticas comenzarán con s.
    - Las variables booleanas deberán comenzar con is o has, por ejemplo, *isChecked*, o *hasPhone*.
    - Los nombres de variables usados comunmente en elementos comunes, como iteradores, deberían mantenerse solo como tal. Por ejemplo, una variable *i* siempre debe usarse en un *for*, como un *int*.
    - Los métodos deben procurar nombre usando verbos. Por ejemplo: *calculateAverage()* sería preferible a *average()*.
- El código puede separarse en regiones para facilitar su lectura.
- Utilizar las anotaciones @Override siempre que sea necesario.
- Evitar tener variables globales innecesarias.
- Es mejor emplear varios métodos que un único método muy largo, primar la facilidad en la lectura.
- Las excepciones no deben ignorarse:

MAL

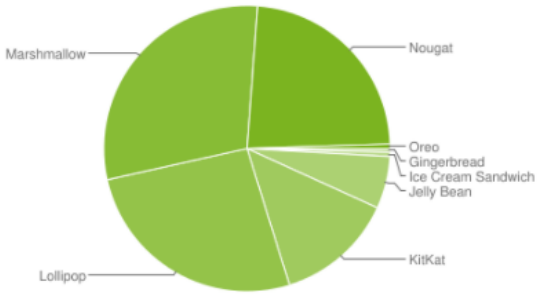
```
void setServerPort(String value) {
    try {
        serverPort = Integer.parseInt(value);
    } catch (NumberFormatException e) { }
}
```

En general, la máxima es intentar mantener un código limpio y ordenado, fácil de leer y consistente. Dado que es un conjunto de normas complejo, importaremos esta guía de estilo dentro del IDE, e intentaremos emplear la filosofía: "Deja el código mejor que como lo encontraste". De este modo, superaremos la dificultad de estar pendientes de cada norma y creamos un sistema iterativo, en el que cada modificación y lectura del código, intentará dejarlo más claro y más limpio.

4.1.2 Versión mínima de Android.

El SDK mínimo necesario para utilizar la app es una decisión que siempre debe tomarse a la hora de realizar un desarrollo en Android. El gráfico muestra el porcentaje de uso para las diferentes versiones, a fecha 12 de diciembre del 2017

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%



Datos recopilados durante un período de 7 días hasta 11/12/2017.
No se muestran versiones con una distribución inferior al 0,1%.

Tras analizar el gráfico, y comprendiendo que las versiones más actuales son menos propensas a fallos y facilitan la realización del código, así como mejoras en el diseño, hemos decidido emplear un API mínimo de 21. Esto, como muestra el gráfico nos hace perder un 20% de los dispositivos, pero nos permite emplear muchas de las nuevas funcionalidades de Android, y sacarle el máximo partido a las nuevas versiones

4.1.3 GIT

A la hora de desarrollar el proyecto, se ha hecho evidente la necesidad de integrar un control de versiones que garantiza que los archivos estuvieran seguros y que permitiera realizar un seguimiento de los cambios.

Para ello, se ha decidido emplear Git, con un repositorio online. En este caso, hemos empleado BitBucket, por ser una herramienta robusta, gratuita, que permite trabajar en privado y con la garantía de disponer de una gran cantidad de usuarios satisfechos.

La URL para acceder al repositorio online es la siguiente: https://sophiecmusicalplus@bitbucket.org/sophiecmusicalplus/homies.git

4.1.4 LIBRERÍAS EMPLEADAS.

Para la realización del proyecto se han empleado diferentes librerías de terceros, que se indican a continuación:

- Librerías de soporte y de diseño de Android: proporcionan las herramientas para el desarrollo de layout. Entre otras, se han empleado las librerías de CardViews y RecyclerViews.

- Vistas de calendario: Con el fin de mostrar un calendario intuitivo y sencillo para el usuario, se ha optado por emplear la librería material-calendar-view, que realiza una implementación limpia de un calendario mensual, siguiendo las guías de estilo de Material Design.

- Librerías de Google: Se han incluido las librerías de Google Play Services, necesarias para la aplicación, así como las librerías de Google que permiten importar los *endpoints* para acceder a la API de Google Calendar.

- **CircleImageView**: Que facilita el recorte de las imágenes en formato circular a través de una vista propia.

- **Firebase**: Empleado a nivel de autorización (OAuth), para el login y registro en la app. Además, se utiliza Firebase Realtime Database a nivel de base de datos.

4.1.5 BASE DE DATOS EMPLEADA

Como gestor de base de datos, se ha empelado Firebase Realtime Database, que proporciona acceso a una base de datos NoSQL almacenada en la nube a través de su librería Database. La creación y mantenimiento de la base de datos son gratuitos, aunque permite un máximo de 100 conexiones simultáneas.

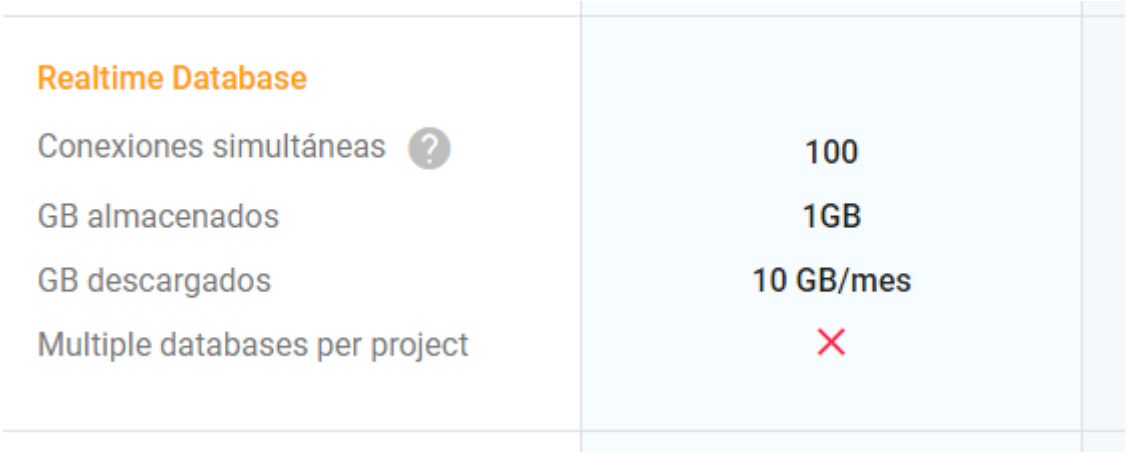


ILUSTRACIÓN 17. SERVICIOS GRATUITOS DE FIREBASE DATABASE.

Para conectar con la base de datos es necesario haber realizado login previamente en Firebase, y tener una sesión activa mediante los token y los objetos OAuth proporcionados por la librería. La gestión de Firebase se realiza a través de identificadores únicos (UIDs) que permiten identificar al usuario. A continuación, se muestra el panel de contol de Firebase con una serie de usuarios de prueba registrados:

ILUSTRACIÓN 18. USUARIOS LOGEADOS EN FIREBASE

En la siguiente imagen, puede apreciarse un ejemplo de la consola de Firebase, en el apartado de base de datos. La captura muestra un conjunto de nodos de prueba en forma de árbol:

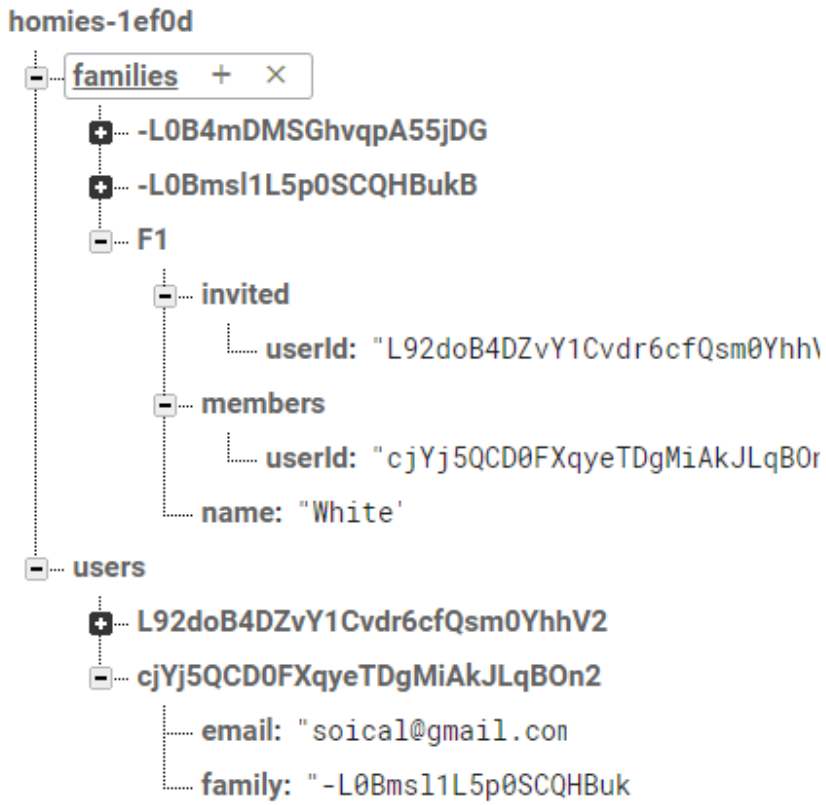


ILUSTRACIÓN 19. BASE DE DATOS NOSQL DE FIREBASE CON NODOS DE EJEMPLO.

4.1.5.1 ESTRUCTURA DE LA BASE DE DATOS

Los nodos *padre* de nuestra base de datos serían los siguientes:

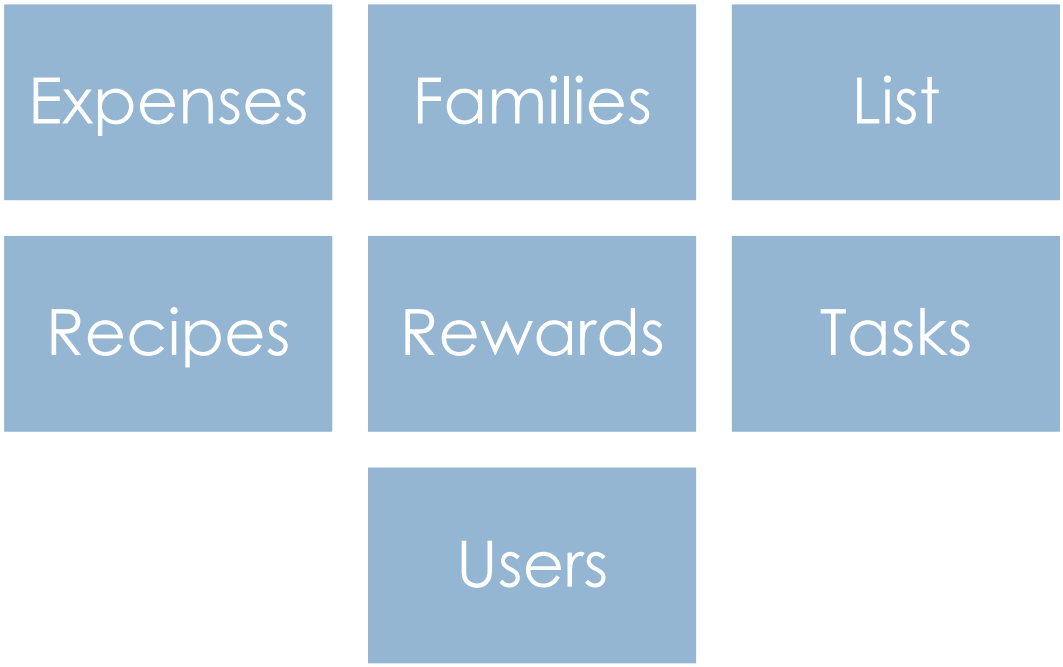


ILUSTRACIÓN 20. NODOS PADRE DE LA BBDD

4.2 DECISIONES RELATIVAS A ARQUITECTURA.

4.2.1 MVC

Para la realización del proyecto se ha optado por escoger una **arquitectura MVC,** de modo que todos los objetos de nuestra aplicación forman parte del modelo, o de la vista, o son controladores. Por tanto, tendríamos, por un lado, los modelos como *User*, *List* o *Family*, que se encuentran en la carpeta model, las clases que controlan la vista, tales como *Adapters* o elementos de los propios layouts como *CircleImageView*, y, por último, controladores que en este caso quedan representados, normalmente, por las propias activities, que ejercen las veces de lógica de negocio.

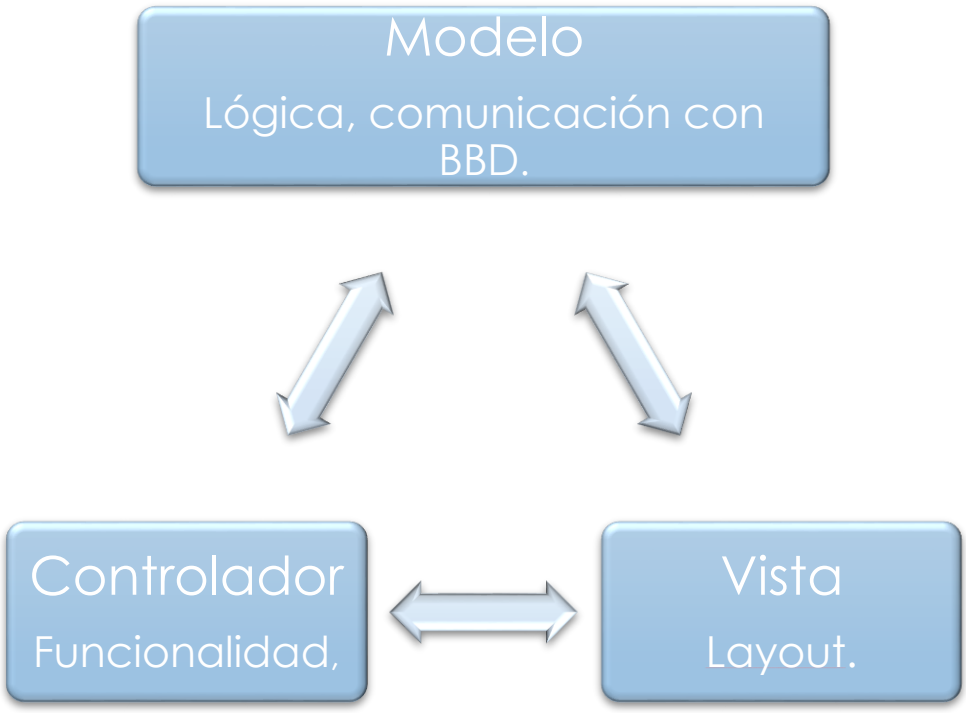


ILUSTRACIÓN 21. PATRÓN MVC

4.2.2 SOLID

Ligado a esto, hemos intentado aplicar, en la medida de lo posible, los **principios SOLID**:

- S-Responsabilidad simple: Manteniendo la máxima de que una clase solo debe tener una responsabilidad, se ha intentado que las clases sean sencillas, con objetivos atómicos, y simplificados. Para las *activities* en concreto, todo el código que pudiera reutilizarse, o que distara de su responsabilidad principal se ha extraído a otras clases que ejercen como parte de la vista en el modelo MVC en algunos casos, helpers que facilitan la interacción entre clases o componentes propios.
- **O-Abierto/Cerrado:** Hemos procurado crear clases genéricas, así como BaseAdapter y BaseActivity con el fin de facilitar la extensión de las mismas. De este modo, vamos desde lo general a lo específico.
- **L-Sustitucion Liskov**: Se ha intentado evitar la dependencia, usando las clases padre cuando fuera posible, para garantizar que el padre es funcional de manera independiente.

- **I-Segregacion del interface**: Las interfaces creadas, si bien son pocas, se han usado fundamentalmente para identificar errores y éxitos en las llamadas al servicio, y se han reducido a la máxima simplicidad.
- **D-Inversión de dependencias**: Se ha intentado en la medida de lo posible, reducir la dependencia, fundamentalmente mediante la segregación de clases y aplicando los otros principios.

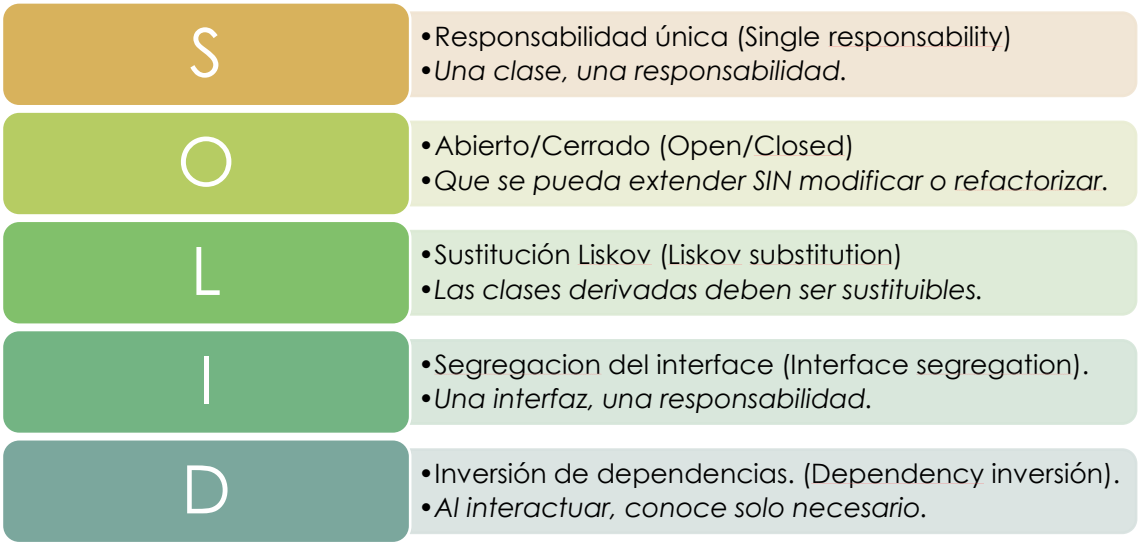


ILUSTRACIÓN 22. PRINCIPIOS SOLID

4.2.3 PATRONES EMPLEADOS

Con el fin de facilitar las labores del desarrollo, hemos utilizando algunos patrones de diseño. En concreto:

- **Patrón Singleton:** Lo hemos empleado en aquellas clases en las que necesitábamos que hubiera una única instancia. Su uso en la app se da exclusivamente en aquellas clases que gestionan repositorios o conexiones.

- Patrón repositorio: Nuestra implementación de este patrón se ha realizado de una manera simplificada, ya que nuestro acceso a datos se ha desacoplado de una API, al hacerse directamente con Firebase. Para su realización, hemos realizado una serie de clases que funcionan como pequeños repositorios cuya responsabilidad es la obtención de una serie de datos, mediante consultas, y su almacenamiento. De esta manera,

evitamos realizar numerosas peticiones a la base de datos, evitando transferencias y consumo de datos innecesarios.

4.1.5 DISTRIBUCIÓN DE CARPETAS

Para facilitar la lectura del código Java, éste se ha dividido de la siguiente forma:

- **Adapters:** Incluye todos los adapters y viewholders del proyecto Su responsabilidad es con la vista e indica, normalmente, como se visualizarán un conjunto de elementos.
- **Components**: Se refiere a vistas propias creadas exclusivamente para este proyecto.
- **Helpers:** Clases que facilitan algún aspecto de la app, por ejemplo, EndlessScrollListener es un listener concreto que facilita la realización de un scroll infinito.
- **Models:** Abarca todos los modelos de la app, desde los modelos de la base de datos, hasta los modelos de error.
- **Services:** En esta carpeta se incluyen todos los archivos repository, que son todas las clases separadas por funcionalidad. Por ejemplo, UserRepository.java es una clase que se encarga de obtener y guardar todos los datos relativos al usuario. De este modo, si en algún lugar de la app necesitamos saber el nombre de usuario, llamaremos a getUser(), lo que nos devolverá el Usuario, realizando la petición solo si hace falta.
- **UI:** Se refiere a aquellos aspectos específicos de la interfaz de usuario, para ello, hemos separado las carpetas relativas a cada pantalla, como si fuesen historias de usuario, y en ella se incluyen los archivos relativos a la interfaz tales como activities o fragments.

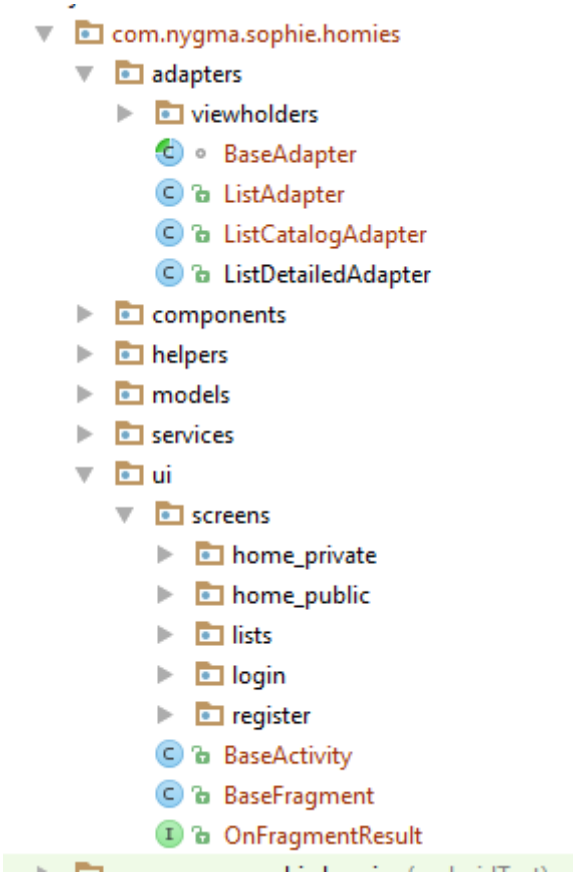


ILUSTRACIÓN 23. ORGANIZACIÓN DE CARPETAS

5. CONCLUSIONES

5.1 CONTINGENCIAS ENCONTRADAS EN EL PROYECTO.

5.1.1 ESTABLIDAD DE ANDROID STUDIO Y CAMBIOS EN GRADLE.

Si bien Android Studio y Gradle son herramientas reconocidas a la hora de desarrollar proyectos en Android, también son herramientas en constante evolución y sus actualizaciones han generado problemas en la compilación, en ocasiones difíciles de resolver. Estos problemas, que en ocasiones han derivado en pérdidas de información, han destacado la necesidad de usar un control de versiones. Del mismo modo, se hace indispensable disponer de un hardware potente para garantizar las pruebas en dispositivos virtuales, o bien disponer de teléfonos diferentes que permitan visualizar el desarrollo.

5.1.2 Realización del back

La parte concerniente a la base de datos, la creación y mantenimiento de la API han resultado ser complejas y más largas de lo esperado. Por ello, se ha decidido emplear una API más sencilla, como la de Firebase que satisfaga los requisitos de seguridad y accesibilidad de la información.

5.1.3 Cambios en la planificación.

Debido a las dificultades encontradas, a la complejidad del proyecto y a complicaciones personales, no ha sido posible seguir la planificación, por lo que a la hora de la implementación se ha optado por simplificar algunos aspectos de la app, en vez de modificar los plazos. En concreto se ha optado por:

- Eliminar la planificación del menú semanal. La vista semanal, además de compleja resultaba confusa a los usuarios que han visualizado el prototipo por lo que ha optado por eliminarse. Como alternativa desde el calendario pueden planificarse las comidas y se ha creado una sección de "recetas" donde incluir el recetario.
- En lo relativo a las estadísticas, que no estaban del todo planificadas se ha optado por dedicar un apartado sencillo a la visualización de lo que debe cada miembro de la familia y un gráfico con lo gastado en los últimos meses. De este modo, se simplifica un poco esta sección que es menos importante.
- Se ha eliminado la subida de fotos a la app, ya que la haría más pesada.
- Se ha eliminado la visibilidad de los eventos y recetas, que inicialmente se planteó la posibilidad de permitir que fueran privados o para toda la familia.
- De momento no se envían e-mails de invitación para incluirte a una familia.
- Se han eliminado las notificaciones push y los recordatorios.

Por último, dado que estas modificaciones se realizan a lo largo de la duración del proyecto y suponen, por tanto, trabajar sin diseño en papel y teniendo que re-evaluar la funcionalidad de la aplicación, se ha decidido solapar la fase de *testing* con estos desarrollos. De esta forma, del 17 de mayo al 13 de junio se concluirá el apartado de estadísticas y recetas.

5.1.3 POSIBILIDADES DE AMPLIACIÓN.

Por un lado, están aquellas funcionalidades que quedaron identificadas dentro de las peticiones de los usuarios y representadas de cara al prototipo, pero que hemos excluido del producto final, identificado como el producto mínimo viable. Sin embargo, además de aquellos aspectos indicados en el apartado anterior hemos considerado algunas otras funcionalidades que podrían mejorar el sistema:

- Guardar menú en PDF: Dado que muchos usuarios lo imprimen y lo pegan en la nevera, quizá fuera interesante disponer de un botón que exportara todas las comidas planificadas en un mes a un archivo PDF.
- Algunas soluciones con funciones de recetario permiten a través de una URL obtener todos los datos de una receta y guardarla en el sistema. Esta sería una funcionalidad costosa, aunque muy útil.
- Integración con apps de conteo de calorías para facilitar la planificación del menú semanal. Un ejemplo de este tipo de aplicaciones es *MyFitnessPal.*
- Retos por puntuación entre usuarios. Con el fin de incentivar la realización de tareas del hogar la app podría permitir retar a otro usuario a conseguir más puntos en un plazo de tiempo. O incluso proporcionar premios a los usuarios con más puntos obtenidos en un mes. Este tipo de "campañas de puntos" buscan fomentar la motivación.

Por último, hay un aspecto que ha quedado de lado a lo largo del proyecto y es la problemática legal de la aplicación. En este sentido, en el transcurso del tiempo de desarrollo ha entrado en vigor a ley de protección de datos que dota de una serie de obligaciones al desarrollador. En este caso, se vuelve prioritaria la búsqueda de asesoramiento legal para proceder a la aplicación de la nueva normativa.

6. BIBLIOGRAFÍA

- MARTIN, ROBERT C and FEATHERS, MICHAEL C, 2009, Clean code. Upper Saddle River, N.J. : Prentice Hall.
- Diseño de Interfaces » Diseño Centrado en el Usuario (DCU), 2018. Multimedia.uoc.edu [online],
- Marvel – Making design simple for everyone, 2018. Marvelapp.com [online].
- 2018. Powtoon.com [online],.
- Trello, 2018. Trello.com [online].
- Curso de Programación Android, 2018. sgoliver.net [online].
- Patrones de diseño: Repository, 2018. developerro [online],
- Cómo funciona Scrum, 2018. Proyectos Ágiles [online],
- Documentation | Firebase, 2018. Firebase [online].
- Google Maps Platform - Geo-location APIs | Google Maps Platform | Google Cloud, 2018. Google Cloud [online].
- Design, 2018. Material Design [online].
- Vectors graphics designed by Freepik.