Menu Generator Android App

Nom Estudiant: Aina Cuxart Caballeria
Màster Universitari en Desenvolupament d'Aplicacions per a Dispositius Mòbils

Nom Consultor/a: Pau Dominkovics Coll
Professor/a responsable de l'assignatura: Carles Garrigues Olivella

6 de Juny de 2018

FITXA DEL TREBALL FINAL

Títol del treball:	*Menu Generator Android App*
Nom de l'autor:	*Aina Cuxart Caballeria*
Nom del consultor/a:	*Pau Dominkovics Coll*
Nom del PRA:	*Carles Garrigues Olivella*
Data de lliurament (mm/aaaa):	*06/2018*
Titulació o programa:	*Màster Universitari en Desenvolupament d'Aplicacions per a Dispositius Mòbils*
Idioma del treball:	*Català*
Paraules clau	*App, Kotlin, Menú*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

En els últims anys ha crescut la preocupació per dur un estil de vida saludable i tenir una alimentació equilibrada. Alhora l'ús de les tecnologies mòbils s'ha disparat entre la població i cada vegada ofereixen més prestacions als usuaris.

Dins d'aquest context, intentarem crear una eina que ajudi als usuaris a planificar una dieta equilibrada de manera còmoda i eficient aprofitant els avantatges de les tecnologies mòbils.

Per fer-ho, desenvoluparem una aplicació nativa per a dispositius Android que tingui com a funcionalitat principal mostrar un menú setmanal a l'usuari.

Inclourem una fase d'anàlisi en la qual estudiarem els usuaris potencials de l'aplicació i quines altres necessitats complementàries tenen. D'aquesta manera buscarem altres funcionalitats secundàries que completin l'aplicació. També farem una definició de requeriments acompanyada d'una fase de disseny, en les quals donarem forma al contingut i a la imatge de l'aplicació.

En la implementació buscarem tecnologies punteres que ens puguin aportar valor i les utilitzarem en el desenvolupament. D'aquesta manera podrem explorar nous camps i aconseguirem fer l'aplicació atractiva tècnicament parlant.

Finalment explorarem altres opcions per fer créixer l'aplicació indicant les línies de treball futur.

Abstract (in English, 250 words or less):

Nowadays, people worry about having a healthy life and a good diet. In addition, the use of mobile phones has increased a lot and offers new features and functions every day.

In this context, we are going to create a new tool that helps users plan a healthy diet easily and effectively by taking advantage of mobile devices.

To do this, we will develop a native application for Android devices whose main functionality will be to display a weekly menu for the user.

There will be a previous stage of analysis where we will study the potential users of the application and what other related needs they have. By doing this, we will look for secondary features that complete the application. We will also define the requirements and design the content and appearance of the application.

For the developement, we will do research to find new technologies that can contribute to create a better app. With these we can explore new fields and make a more technically attractive app.
Finally, other options to grow the app will be explored showing the following lines of developement.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

La nostra societat cada cop es preocupa més per dur un vida saludable i vigila més de què s'alimenta per intentar tenir una bona qualitat de vida. Alhora, busca tenir més temps lliure per gaudir de les aficions personals i s'intenta no ocupar el temps amb coses cootidianes, com anar a comprar aliments o cuinar. La tendència és reduir el temps de dedicació a les obligacions casolanes així com minimitzar les preocupacions relacionades amb la llar.

Moltes de les funcionalitats que ens ofereixen els dispositius mòbils ens serveixen per planificar activitats o per tenir un registre de tot tipus de coses que permeten a l'usuari despreocupar-se de pensar-les i recordar-les.

Dins d'aquest context, ens agradaria crear una aplicació que ajudés a planificar un menú equilibrat per tal d'aportar una nova eina que encaixi amb el l'estil de vida de la societat actual i ajudi a dur una alimentació saludable.

Fent una cerca al Play Store, trobem aplicacions que contenen dietes i/o receptes i d'altres que et permeten organitzar-te els teus propis menús o introduir les teves pròpies receptes. Algunes d'aquestes aplicacions són:

- Nootric: ofereix dietes setmanals amb objectius variats creades per nutricionistes. Entre els objectius trobem aprimar-se x quilos en un cert temps, definir i tonificar el cos, dieta vegana, per celíacs, etc. Un cop triada la dieta et donen el detall de tots els àpats i et permet fer un seguiment de l'evolució del teu pes. (Veure [1])
- Lifesum: ofereix plans per a fer dieta i perdre pes. T'ofereix receptes i et permet fer un seguiment de les calories consumides i de l'evolució del teu pes. (Veure [2])
- Runtasty: hi trobem receptes saludables, vídeo receptes i alguns consells de cuina. No ofereix menú ni opció de crear-los. (Veure [3])
- Big Oven: conté una gran quantitat de receptes i permet a l'usuari crear-se el seu propi menú setmanal i organitzar la llista de la compra. (Veure [4])
- Nestlé cocina: ofereix receptes sanes, vídeos i tutorials. Conté un apartat de "Menú planner" que ofereix un menú amb tots els àpats per a una setmana. És una aplicació de la marca Nestlé i per tant, les receptes que ofereix estan restringides a les que contenen algun dels seus productes. (Veure [5])

Després d'analitzar les diferents aplicacions que hem trobat al Play Store, hem vist que totes tenen alguna mancança. A més, la majoria estan enfocades a perdre pes. Ens agradaria crear una aplicació que oferís la majoria de les funcionalitats observades a les aplicacions que trobem al mercat així com d'altres per tal de millorar les actuals i que ofereixin un menú saludable i equilibrat. Les funcionalitats que creiem que hauria de tenir l'aplicació són:

- Crear un menú setmanal.
- Fer que el menú sigui equilibrat.
- Oferir una base de dades de receptes.
- Mostrar el detall de les receptes.
- Generar una llista de la compra editable a partir del menú planificat.
- Permetre a l'usuari destacar les seves receptes favorites.
- Que l'aplicació sigui multidispositiu.
- Permetre a l'usuari crear un entorn compartit per visualitzar la mateixa informació que els seus companys o familiars.

Degut al poc temps disponible per a realitzar aquest projecte no es poden incloure totes aquestes funcionalitats en el desenvolupament del treball. En les properes seccions s'estudiarà i justificarà l'abast del projecte. Al final de la secció 1.4, després de la planificació temporal, s'indicarà quines d'aquestes funcionalitats es desenvoluparan al treball.

1.2 Objectius del Treball

La idea principal de l'aplicació que es vol desenvolupar és donar a l'usuari un menú equilibrat per a tota la setmana, informant de les receptes de cada àpat i donant el detall de cada recepta. Per fer que el menú sigui equilibrat caldria desenvolupar un algoritme que el calculés.
El disseny i programació de l'algoritme que calculi el menú equilibrat se surt de la matèria del master, per això hem considerat que és una funcionalitat que queda fora de l'abast del projecte.

Els objectius principals que es volen aconseguir amb aquest treball són:

- Crear una aplicació per a dispositius Android, programada en Kotlin, que ofereixi un menú per a una setmana a l'usuari. L'aplicació contindrà les pantalles i funcionalitats necessàries per indicar quines receptes s'han de cuinar per a cada àpat durant una setmana.
- Fer que l'aplicació sigui multidispositiu.
- Realitzar el disseny de l'aplicació seguint les guies d'estil i criteris d'usabilitat dels terminals Android.
- Definir l'abast del projecte a partir de les necessitats detectades i el temps disponible així com una bona planificació. També establir quins passos caldria seguir per desenvolupar una aplicació completa que millori les que trobem actualment al mercat.

A partir d'aquests objectius sorgeixen els següents requeriments funcionals per a l'aplicació a desenvolupar:

1. Es desenvoluparà una aplicació que oferirà a l'usuari un menú equilibrat per a tota una setmana.
2. L'aplicació tindrà les pantalles: "Menú setmanal", "Llista de receptes", "Detall d'una recepta" i "Pantalla inicial".
3. La navegació dins l'aplicació es durà a terme amb un menú lateral. Es podrà accedir al detall de cada recepta des de les pantalles "Menú setmanal" i "Llista de receptes".
4. A la pantalla "Menú setmanal" es mostrarà a l'usuari el nom de la recepta que ha de cuinar a cada àpat. Seleccionant una recepta accedirà al seu detall.
5. L'aplicació gestionarà les dates per actualitzar el menú quan comenci una nova setmana.
6. La "Llista de receptes" mostrarà les receptes disponibles ordenades alfabèticament i permetrà a l'usuari accedir al detall. Inclourà un buscador.
7. El "Detall de recepta" mostrarà la informació necessària per a la seva preparació: ingredients, temps, passos de preparació, imatge…
8. L'usuari podrà marcar una recepta com a favorita i filtrar la llista de receptes per visualitzar només aquestes.

I finalment, els requeriments no funcionals següents:

1. L'aplicació serà per a dispositius Android.
2. L'aplicació es desenvoluparà en Kotlin seguint una arquitectura basada en Model-View-Presenter.
3. Tindrà un disseny adaptat a totes les mides de pantalla.
4. Es requerirà una versió igual o superior a Android 5.0 per poder executar l'aplicació.
5. Per accedir a l'aplicació l'usuari s'haurà d'identificar mitjançant un login amb Google.
6. Les dades de les receptes així com el menú setmanal i les receptes favorites de cada usuari s'emmagatzemaran a Firebase.
7. Les dades s'emmagatzemaran en local al dispositiu, mitjançant una base de Dades Realm, i s'actualitzaran amb una determinada freqüència.
8. Es requerirà connexió a internet per a fer login i per descarregar les dades. Després del primer ús, si no hi ha connexió, es mostrarà informació desactualitzada.

1.3 Enfocament i mètode seguit

Per donar una solució tecnològica al problema plantejat s'hauria pogut optar per fer una web o una aplicació. Ens hem decantat per una aplicació, ja que crearem una eina que l'usuari utilitzarà diàriament, és a dir, tindrà un ús freqüent de manera que li resultarà còmode estar identificat i poder accedir a les seves dades de manera ràpida. A més, pretén ser una eina de consulta que s'utilitzarà en espais diversos, com ara la cuina de casa o el supermercat, entorns on no es disposa d'un ordinador però si d'un smartphone.

Un cop decidit que farem una aplicació per a mòbils, hem optat per fer-la nativa perquè volem fer una aplicació que sigui atractiva fàcil d'usar per l'usuari. "Les aplicacions natives ofereixen un millor rendiment i són més eficients" [6], a més de ser més fàcils de mantenir amb les actualitzacions del sistema operatiu. Un últim motiu és que "les aplicacions natives són més segures" [7], fet rellevant al tractar les dades de l'usuari.

Vull remarcar també un motiu acadèmic. Personalment em suposa un repte crear una aplicació per Android i he considerat que el treball de final de màster és una bona manera d'aprofundir els meus coneixements en aquesta plataforma.

Actualment existeixen dos llenguatges de programació per desenvolupar aplicacions per Android, Java i Kotlin.

La batalla legal entre Oracle i Google va provocar la cerca per part de Google d'una alternativa al seu llenguatge estrella. La va trobar amb Kotlin, el nou llenguatge creat per Jetbrains, el qual és un llenguatge madur, ràpid i estable [8]. El Maig de 2017, Google va anunciar al seu I/O Event que Kotlin passava a ser el llenguatge oficial de desenvolupament d'aplicacions Android [9].

Si buquem documentació i formació per aprendre el llenguatge, trobem el llibre "Kotlin for Android developers" d'Antonio Leiva, que és la guia recomanada per Google per aprendre Kotlin. El seu autor ens dóna 12 arguments sobre la importància d'aquest llenguatge al seu blog personal [10] :

D'aquests, jo en destaco els següents:

- "2. Facilita mucho el desarrollo en Android"
  Kotlin és un llenguatge simple amb molt potencial, que ajuda a fer un codi més sòlid i comprimit que Java, per exemple.
- "4. Su evolución está bien cubierta"
  Darrere de Kotlin hi ha l'empresa Jetbrains, l'empresa creadora de l'Android Studio, l'IDE més potent per desenvolupar amb Android, i juntament amb el suport oficial de Google, fa que es pugui considerar una aposta segura.

- "6. Es mucho más seguro que Java"
  Kotlin utilitza menys codi que Java, fent que sigui més difícil que es donin errors i és més fàcil d'entendre.

Tots aquests motius ens fan indicar que Kotlin és un llenguatge interessant d'aprendre, innovador i modern, i que serà una clara tendència al mercat, motius pels quals m'he decantat per desenvolupar l'aplicació en ell.

1.4 Planificació del Treball

Per al desenvolupament del projecte necessitarem un ordinador amb Android Studio 3.0 i l'Android SDK instal·lat.

La dedicació al projecte serà de 20 hores setmanals, 6 entre tots els dies laborales i 14 els caps de setmana.

Abans de començar el desenvolupament hem de dur a terme les fases de **disseny i anàlisi**, en les quals estudiarem quins són els potencials usuaris i quines necessitats tenen. A partir d'aquest estudi definirem les funcionalitats de l'aplicació i la seva estructura. El resultat d'aquestes fases s'inclourà a la memòria i s'entregarà el dia 4 d'Abril. A continuació detallem les tasques que s'han planificat per aquestes.

Un cop prototipada l'aplicació i dissenyada la seva estructura, començarem amb el la fase **d'implementació**. En aquesta fase prepararem l'entorn per treballar amb les dades i desenvoluparem l'aplicació. A la memòria inclourem un recull de les decisions que s'hagin pres durant el desenvolupament, així com d'aspectes a destacar del codi i el conjunt de proves de l'aplicació. El resultat de la fase d'implementació s'entregarà el 16 de Maig.

Les primeres tasques d'aquesta fase són:
- Preparació de l'entorn per les crides a l'API.
- Gestió de la lectura i escriptura de les dades.
- Crear i configurar una base de dades interna.

Amb les estructures per comunicar amb l'API creades, començarem el desenvolupament de les pantalles. A continuació detallem les tasques que s'han planificat per a realitzar a cada una.

- Pantalla de login
    - Maquetació de la pantalla.
    - Gestió del login amb Google.
- Menú lateral
    - Maquetació del menú lateral, afegint totes les entrades necessàries i aplicant el disseny corresponent.
    - Lògica de navegació del menú cap a les diferents pantalles de l'aplicació.
- Pantalla del menú setmanal
    - Maquetació de la pantalla que mostrarà les receptes indicades per a cada àpat d'un dia i permetrà a l'usuari canviar el dia per tal de consultar les de tota la setmana.
    - Lògica de la llista del menú setmanal, la qual ha de mostrar les dades que s'han descarregat de l'API.
    - Gestió dels resultats segons la data. S'haurà de demanar nova informació un cop hagi acabat la setmana.
- Pantalla detall de recepta
    - Maquetació de la pantalla de detall, la qual mostrarà la informació corresponent a cada recepta: imatge, nom, ingredients, etc.
    - Càrrega de les dades obtingudes des de l'API
- Llista de receptes
    - Maquetació de la pantalla que contingui la llista de totes les receptes ordenades alfabèticament.
    - Gestió de les dades de les receptes emmagatzemades a la base de dades.
    - Càrrega del contingut a la llista.

També es definiran les proves que s'han de realitzar per confirmar el correcte funcionament de l'aplicació i es comprovarà que l'app les superi satisfactòriament.

Després d'aquesta entrega, comencem la **fase final** en la qual acabarem la memòria, crearem el manual d'usuari i prepararem l'entrega del treball, indicant els passos de compilació i enregistrant la presentació virtual. La data de finalització d'aquesta fase, i del treball, és el 6 de Juny.

S'hauran de realitzar les tasques següents:

- Redacció de les conclusions del treball.
- Retocs a la memòria per unificar l'estil i enllaçar seccions.
- Definir instruccions per compilar l'aplicació indicant el programari requerit.
- Preparar el manual d'usuari de l'aplicació.
- Preparar i enregistrar la presentació del treball.

Sintetizem la planificació exposada en un diagrama de Gantt (Veure figura 1).

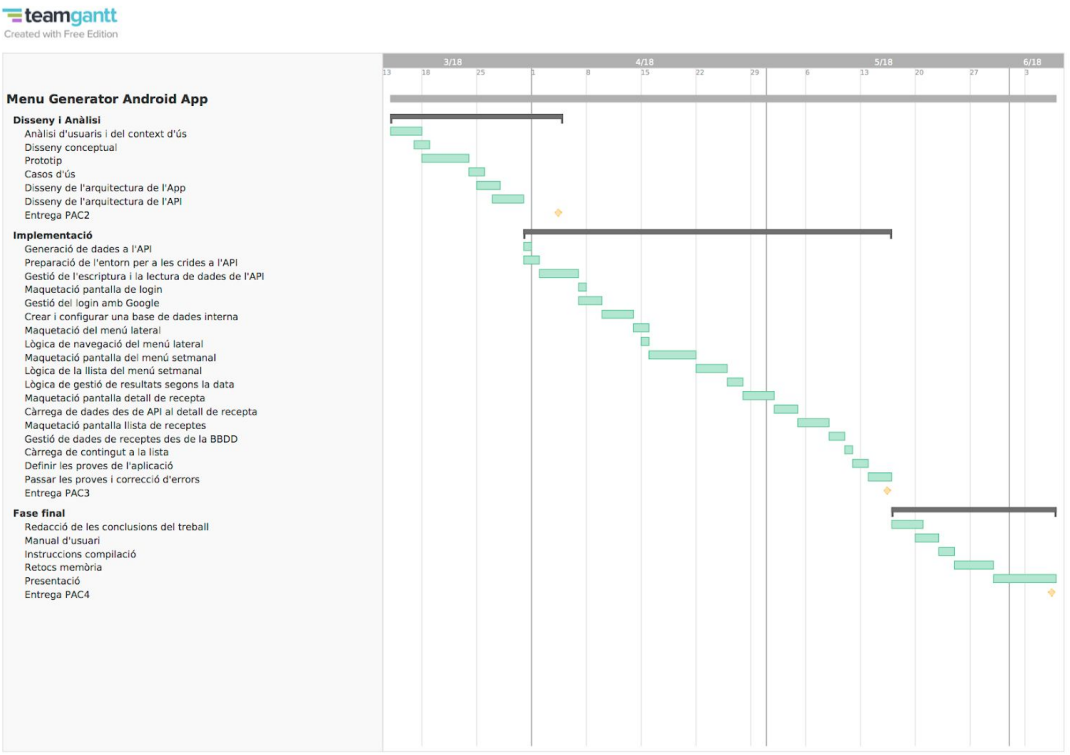


Figura 1: Diagrama de Gantt del projecte

A continuació resumim quines de les funcionalitats exposades a la secció 1.1 entren dins l'abast del projecte. Hem decidit incloure aquestes ja que considerem que són les bàsiques per al funcionament.

Funcionalitat	TFM	Futur
Crear un menú setmanal	✅	
Fer que el menú sigui equilibrat		✅
Oferir una base de dades de receptes		✅
Oferir una base de dades de receptes de prova	✅	
Mostrar el detall de les receptes	✅	
Generar una llista de la compra editable a partir del menú planificat		✅
Permetre a l'usuari destacar les seves receptes favorites *	✅	
Que l'aplicació sigui multidispositiu	✅	
Permetre a l'usuari crear un entorn compartit per visualitzar la mateixa informació que els seus companys o familiars		✅

* Si hi ha imprevistos durant el desenvolupament de l'aplicació, aquesta funcionalitat quedarà fora de l'abast del TFM.

Taula 1: Abast de les funcionalitats del projecte

1.5 Breu sumari de productes obtinguts

Un cop finalitzat el treball de final de master obtindrem una aplicació per Android que oferirà un menú setmanal a l'usuari. L'aplicació anirà acompanyada d'un manual d'usuari.

També s'entregarà el codi font de l'aplicació i una guia indicant els paquets necessaris per a la seva compilació.

Es realitzarà un vídeo de presentació virtual on s'exposaran breument els continguts del treball i es farà una demostració de l'aplicació desenvolupada.

1.6 Breu descripció dels altres capítols de la memòria

El capítol 2 estarà dedicat a l'anàlisi previ de l'aplicació incloent el context i els casos d'ús. També contindrà el disseny de l'aplicació, i s'especificaran les decisions preses en el prototipat.

El capítol 3 contindrà l'estructura i l'arquitectura, tant de l'aplicació com de l'API.

A la memòria no es discutirà ni comentarà tot el codi desenvolupat, però el capítol 4 contindrà els aspectes que es considerin rellevants d'aquest tema.

Al capítol 5 trobarem les proves que haurà de superar l'aplicació.

Finalment al capítol 6, el capítol de conclusions, s'analitzarà l'estat final del projecte i es valorarà si s'han assolit els objectius. Anirà seguit de la bibliografia i el glossari.

2. Disseny

En aquest capítol treballarem el disseny conceptual de l'aplicació, mostrant els resultats obtinguts i els motius de les decisions preses.

2.1. Disseny centrat en l'usuari

En aquesta secció seguirem un patró de disseny centrat en l'usuari, el qual es divideix en 4 fases:

1. Usuaris i context d'ús
2. Disseny conceptual
3. Prototipat
4. Avaluació

Donat que no disposem d'usuaris reals que puguin validar el disseny, en aquest treball no realitzarem les proves de la fase d'avaluació.

2.1.1. Usuaris i context d'ús

L'objectiu d'aquesta primera fase és conèixer les característiques dels potencials usuaris de l'aplicació, així com les seves necessitats, objectius i el context d'ús.

El tipus d'usuari estàndard en què hem pensat per l'aplicació és el d'una dona treballadora de 40 anys, que dins l'entorn familiar és qui s'encarrega de les tasques de casa.

Pensar què ha de cuinar per cada àpat per dur una dieta equilibrada li suposa molt de temps en el seu dia a dia, i sovint la falta de planificació es tradueix en una mala alimentació.

Volem donar solució a aquest problema creant una eina que l'ajudi a organitzar-se, generant un menú equilibrat d'àpats per tota la setmana. D'aquesta manera, estalviarà temps i preocupacions, a més de dur una dieta sana i variada.

Fitxes de Persones

A continuació creem tres fitxes de persona que exemplifiquen quins són els arquetips d'usuari de l'aplicació. Les dues primeres representen usuaris de tipus focal i l' última de tipus secundari.

FITXA 1	
Nom: Gemma	**Edat**: 42 anys
Professió: Administrativa	
Descripció de la persona: La Gemma està casada i té dues filles, viuen a Barcelona i treballa en una consultoria informàtica. Fa ioga i li agrada molt jugar a Pàdel. És una dona molt activa i intenta transmetre a les seves filles les ganes de fer coses. Aprofiten els caps de setmana per fer excursions en família i allunyar-se una mica de la ciutat. Tot i no estar molt a l'última en tecnologia, fa servir molt el mòbil a la feina, i és una usuària molt activa a les xarxes socials.	
Context d'ús: Cada vespre la Gemma prepara el sopar i el dinar de l'endemà per tota la família (carmanyoles). Se n'encarrega ella perquè normalment el seu marit arriba més tard a casa, així que és ella qui ha de decidir que cuinar cada dia, alhora que es preocupa per la salut de tots. Molts dies està cansada i li és molt difícil saber què preparar. Sense Menu Generator faria el primer que trobés a la nevera, deixant de banda la dieta equilibrada, però gràcies a l'aplicació no ha de pensar-ho.	

FITXA 2	
Nom: Rosa	**Edat**: 50 anys
Professió: Metgessa	
Descripció de la persona: La Rosa viu amb el seu fill adolescent a Calella. Li agrada molt la pasta italiana, i li agrada cuinar-la de diferents formes. És molt apassionada dels animals i col·labora sovint amb protectores fent tasques de difusió i ajudant a cuidar els animals. A casa tenen un gat. Li agrada molt viure a prop del mar i li encanta passejar per la platja. Tres cops per setmana surt a córrer pel passeig marítim.	
Context d'ús: No té gaire temps entre setmana, així que aprofita el dissabte per fer anar al supermercat. Això fa que hagi de pensar molt bé què compra i en quina quantitat per fer els àpats que menjaran ella i el seu fill. Fins ara, acabava comprant sempre el mateix i sovint li sobraven o li faltaven aliments al final de la setmana. Ara, gràcies a l'aplicació Menu Generator, pot organitzar-se millor i adequa la compra al que necessita cada setmana.	

FITXA 3	
Nom: David	**Edat**: 30 anys
Professió: Físic	
Descripció de la persona: El David s'ha independitzat fa poc més d'un any i viu sol a un pis a Sabadell. Treballa d'investigador a la universitat autònoma des que va acabar el doctorat. Aprofita molts caps de setmana per fer *room scapes* i sortir amb els seus amics. Segueix moltes sèries a Netflix i aprofita qualsevol estona lliure per veure algun capítol. És molt endreçat i li agrada tenir cura de casa seva. Està molt al dia en aspectes tecnològics.	
Context d'ús: Com que ha de cuinar per ell sol, molts dies acaba comprant menjar precuinat i no té una dieta sana. Tot i haver après a cuinar a casa dels seus pares, moltes vegades no se li acut què fer i escull l'opció més fàcil i ràpida. Des que té l'aplicació Menu Generator, intenta seguir el que li aconsella per menjar millor.	

A partir de l'anàlisi dels usuaris i del seu context d'ús, podem extreure que les funcionalitats de l'aplicació han de ser:

- Fer un menú setmanal, indicant els plats per a cada àpat. (Veure context fitxa 1)
- El menú ha de ser equilibrat i saludable. (Veure context fitxa 1)
- Accedir a una llista de receptes. (Veure context fitxa 3)
- Mostrar el detall de les receptes, amb els ingredients i els passos de preparació. (Veure context fitxa 2)
- S'han de poder marcar receptes com a favorites. (Veure context fitxa 3)
- Crear una llista de la compra amb els productes necessaris per a preparar els plats indicats al menú. L'usuari podrà editar-la, afegint o traient productes en funció de les seves necessitats. (Veure context fitxa 2)

2.1.2. Disseny conceptual

L'objectiu d'aquesta fase és elaborar escenaris d'ús de l'aplicació a partir dels usuaris i contextos presentats anteriorment. Aquests escenaris ens permetran detectar les necessitats dels usuaris que el disseny ha de cobrir.

A continuació es mostren els escenaris d'ús creats per a cada una de les persones creades a l'apartat anterior *2.1.1. Usuaris i context d'ús*.

FITXA 1
Nom: Gemma
Escenari d'ús: Són quarts de vuit del vespre de dimarts i les filles de la Gemma ja han acabat els deures de l'escola, ella les ha ajudat a fer-los. Mentre les nenes es dutxen, la Gemma va cap a la cuina per preparar el sopar i el dinar. Entra a l'aplicació per consultar quins plats corresponen al sopar de dimarts i al dinar de dimecres. Fa una lectura ràpida dels ingredients i els passos de preparació per no oblidar-se res i es posa a cuinar.

FITXA 2
Nom: Rosa
Escenari d'ús: És dissabte al matí i la Rosa va amb el seu fill a comprar al supermercat. Abans de sortir de casa entra a l'aplicació Menu Generator i consulta els ingredients que necessitarà aquesta setmana per cuinar tots els plats indicats. Amb la llista a la mà, comprova el rebost de casa per veure què hi té i modifica la llista, esborra algun aliment que ja té i afegeix altres coses com ara productes de neteja i coses per l'esmorzar. Quan arriben al supermercat, va marcant de la llista els productes a mesura que els va agafant.

FITXA 3
Nom: David
Escenari d'ús: És dissabte al migdia i el David porta tot el matí fent feina a casa. La nit anterior va estar sopant amb uns amics i no ha dormit gaire, està cansat. S'apropa l'hora de dinar i nota que té gana però està cansat i no té gaires ganes de cuinar. El primer pensament que té quan es planteja que dinarà, és fer-se uns fideus instantanis, però sap que no són gaire sans i decideix entrar a l'aplicació Menu Generator per veure quin plat li suggereix. El David té una llista força gran de receptes que ha marcat com a favorites les quals considera que són prou sanes i fàcils de fer. La recepta que li toca avui és una d'elles, així que decideix fer-li cas i es posa a cuinar. Està content perquè s'ha tret la mandra de sobre i farà un dinar sa i equilibrat.

A partir dels escenaris exposats, tenim que les pantalles en les quals s'estructurarà l'aplicació són:

- Pantalla de menú setmanal
- Llista de receptes
- Detall d'una recepta
- Llista de la compra

A més, afegirem una pantalla de login per que els usuaris puguin accedir a les seves dades des de més d'un dispositiu així com compartir dispositiu conservant la seva identitat, i un menú lateral com a eina de navegació entre les diferents pantalles de l'aplicació.

A continuació exposem la relació entre aquestes pantalles: La pantalla per defecte serà la del Menú setmanal i la navegació entre pantalles es farà mitjançant un menú lateral. Des del menú setmanal i des de la llista de receptes es podrà accedir al detall d'aquestes.
Veiem aquesta estructura en el següent diagrama. (Veure figura 2)

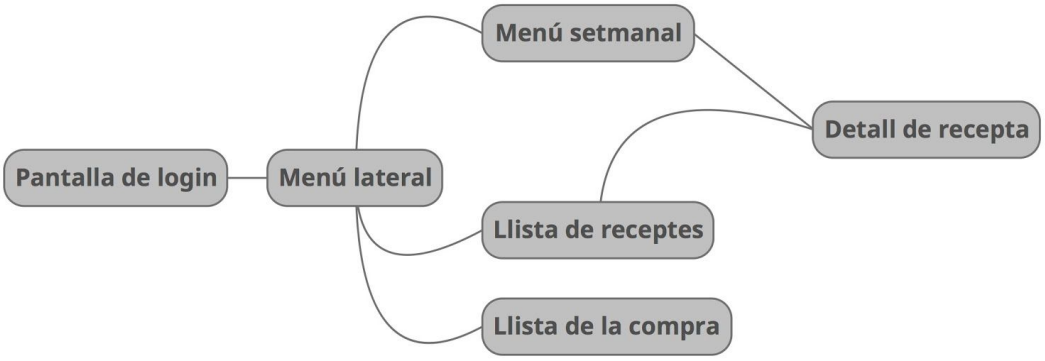


Figura 2: Diagrama de flux de l'aplicació.

2.1.3. Prototipat

A partir del diagrama de flux de l'aplicació s'han prototipat les diverses pantalles presents a l'aplicació.

La pantalla inicial serà la de login, la qual permetrà a l'usuari identificar-se i només apareixerà quan l'usuari hagi tancat sessió. (Veure figura 3)

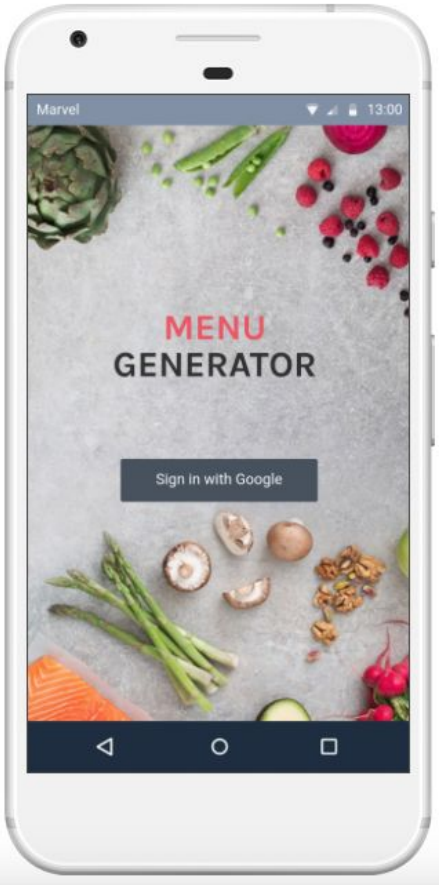


Figura 3: Pantalla de login

La pantalla per defecte serà la del Menú Setmanal, ja que és la que l'usuari utilitzarà amb més freqüència. A la part superior l'usuari podrà canviar de dia de la setmana mitjançant uns botons. A continuació es mostrarà de manera destacada el dia seleccionat i, a sota, una llista amb les receptes indicades per a cada àpat d'aquest dia. Al clicar sobre una recepta, navegarem al seu detall. (Veure figura 4)

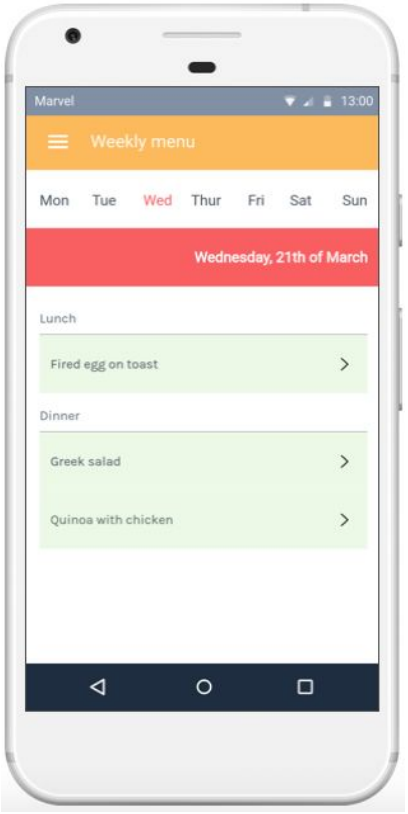


Figura 4: Pantalla del menú setmanal

Per a la navegació entre pantalles, utilitzarem un menú lateral (*Navigation drawer)* per que permet crear enllaços a moltes pantalles i, tot i que en aquest treball ens servirà per navegar entre dues, permet que l'aplicació creixi sense una redefinició d'estructura base. [12] (Veure figura 5)

Hem descartat la utilització d'un menú inferior (*Bottom Navigation)* per que els botons que permeten canviar de dia a la pantalla Menú Setmanal, recorden als *tabs* d'Android i les guies no aconsellen barrejar la navegació inferior amb els *tabs*:

> "Be cautious when combining bottom navigation with tabs,
> as the combination may cause confusion when navigating an app." [13]

Figura 5: Menú lateral de navegació

A la llista de receptes es mostraran totes les receptes disponibles. Com que hi haurà un volum gran de receptes, el format llista permetrà a l'usuari veure-les totes fent *scroll* a la pantalla. Per que l'usuari pugui identificar-les es mostrarà el nom de cada recepta i una miniatura de la imatge que farà més visual i agradable l'experiència de l'usuari. (Veure figura 6)

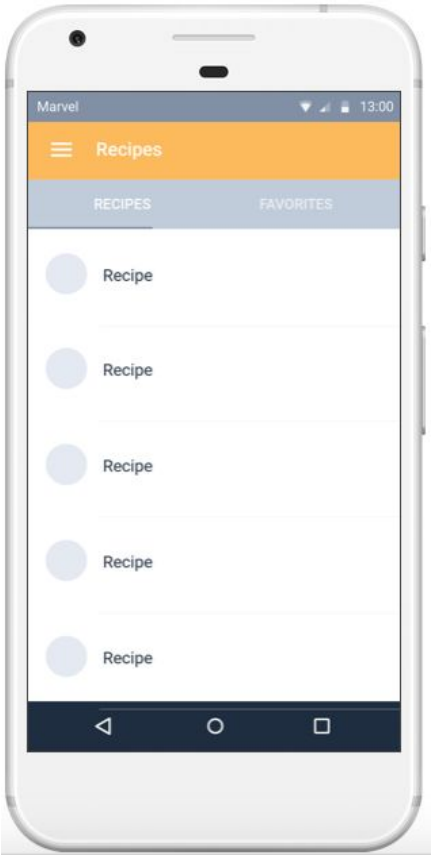


Figura 6: Pantalla llista de receptes

A la pantalla de detall de recepta, es mostrarà una imatge de la recepta a la capçalera per tal de que l'usuari tingui una referència gràfica sobre el que es descriu. A més, hi trobarà la informació necessaria per a preparar la recepta, és a dir,els ingredients i passos de preparació. Per a la gestió dels favorits, hem optat per afegir un *Floating button* per permetre a l'usuari marcar o desmarcar la recepta com a favorita, ja que és una funcionalitat que volem promoure i destacar. [14] (Veure figura 7)

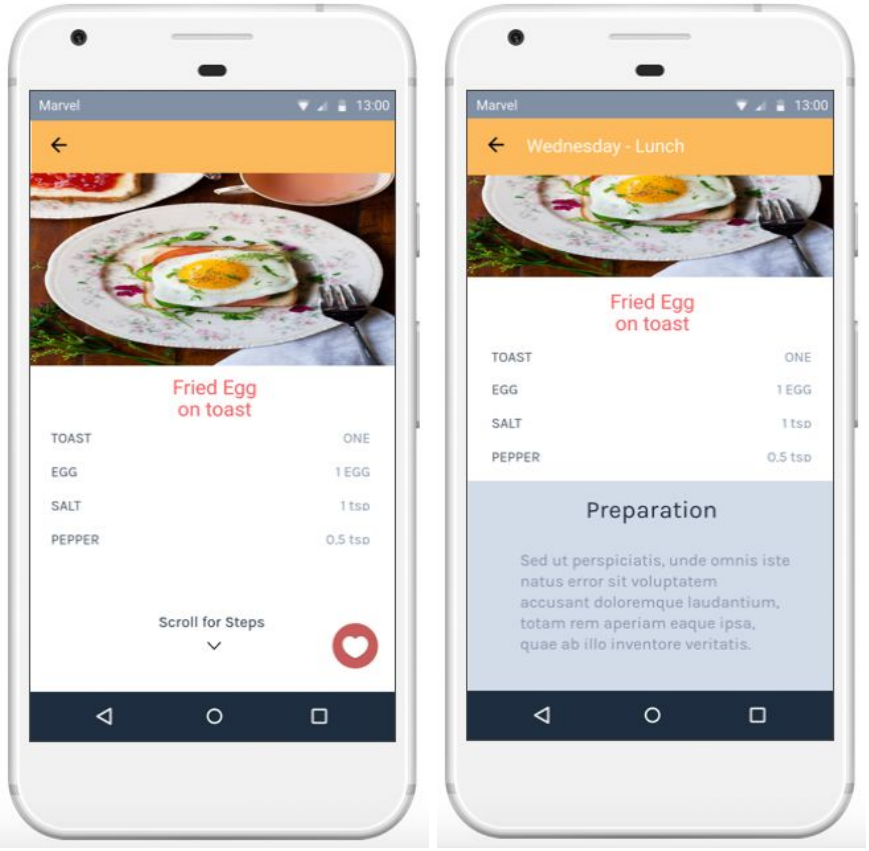


Figura 7: Pantalla de detall de recepta

A partir d'aquestes decisions, hem obtingut el prototip representat a la figura 8:

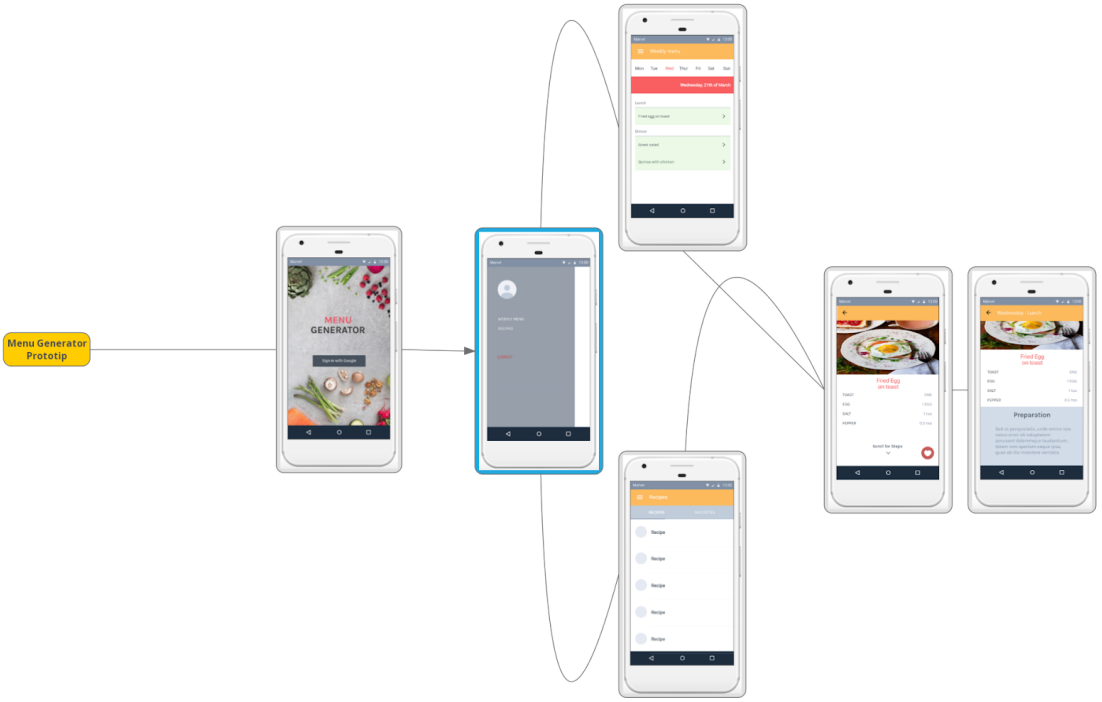


Figura 8: Prototip de l'aplicació

Podem trobar un prototip interactiu de l'aplicació al següent enllaç: https://marvelapp.com/c7ah0ae

2.2. Aclariments funcionals

Repassem alguns aspectes funcionals que queden confusos amb el disseny conceptual, els casos d'ús i el prototipat:

- El menú setmanal es generarà automàticament per a cada setmana. Hi haurà una API encarregada del procés des de la qual l'aplicació es descarregarà les dades.
- Les receptes favorites es marcaran amb un distintiu visual en la llista de receptes i es podran consultar específicament en el seu tab.
- La identificació de l'usuari mitjançant la pantalla de login permet que que l'aplicació sigui multidispositiu, ja que les dades s'emmagatzemaran a l'API i es descarregaran en iniciar sessió.
- També utilitzarem la identificació de l'usuari per a crear l'entorn compartir i fer possible que pugui visualitzar el mateix menú que els seus familiars o companys. (Aquesta funcionalitat no entra a l'abast d'aquest projecte, veure taula 1)

3. Disseny tècnic

En aquest capítol analitzarem el disseny tècnic de l'aplicació, mitjançant casos d'ús i diferents diagrames que definiran l'estructura de l'aplicació i de la base de dades.

3.1. Definició dels casos d'ús

A continuació llistem els diferents casos d'ús de l'aplicació, que ens permetran determinar tot el flux i les funcionalitats de l'aplicació. Per a cada un indicarem: la descripció, els actors, les precondicions i postcondicions, el flux principal i, si s'escau, l'alternatiu.

Nom: Iniciar sessió
Descripció: Permet a l'usuari iniciar sessió i accedir a l'aplicació
Actors: Usuari
Precondició: Tenir l'aplicació instal·lada al dispositiu
Postcondició: Estar identificat a l'aplicació i poder accedir-hi
Flux principal: 1. L'usuari prem el botó d'iniciar sessió amb Google. 2. L'aplicació mostra la pantalla d'inici de sessió de Google. 3. L'usuari introdueix les seves credencials de Google. 4. L'aplicació captura si el login s'ha realitzat correctament i navega a la pantalla del menú setmanal.
Flux alternatiu: 4b. Si les credencials no són correctes, l'aplicació es queda a la pantalla d'introducció de credencials.

Nom: Consulta menú setmanal
Descripció: L'usuari entra l'aplicació per consultar el menú setmanal
Actors: Usuari
Precondició: Estar identificat a l'aplicació amb credencials vàlides
Postcondició: Es mostra el menú setmanal
Flux principal: 1. L'usuari entra a l'aplicació. 2. L'aplicació mostra la pantalla *Menú setmanal.* 3. L'aplicació comprova que té el menú setmanal guardat a la memòria local. 4. L'usuari pot navegar entre els dies de la setmana mitjançant els botons superiors. 5. L'aplicació carrega les dades corresponents al dia seleccionat i les mostra.
Flux alternatiu: 3b. Si no té el menú setmanal emmagatzemat en local, realitza una crida a Firebase per descarregar-lo i el guarda.

Nom: Consulta del detall d'una recepta del menú
Descripció: L'usuari consulta el detall d'una recepta que se li ha indicat al menú
Actors: Usuari
Precondició: L'usuari es troba a la pantalla de *Menú setmanal*
Postcondició: L'usuari està a la pantalla de *Detall de recepta*
Flux principal: 1. Dins la pantalla menú setmanal, l'usuari navega al dia on hi ha indicada la recepta. 2. L'usuari prem sobre el nom de la recepta que vol consultar. 3. L'aplicació navega a la pantalla de *Detall de recepta* on mostra la informació sobre la recepta seleccionada.
Flux alternatiu: 4. L'usuari pot tornar a la pantalla anterior prement el botó "Back" de la pantalla o del dispositiu.

Nom: Consulta del detall d'una recepta de la llista
Descripció: L'usuari entra l'aplicació per consultar el detall d'una recepta
Actors: Usuari
Precondició: Estar identificat a l'aplicació amb credencials vàlides
Postcondició: Es mostra el detall d'una recepta
Flux principal: 1. L'usuari entra a l'aplicació. 2. L'aplicació mostra la pantalla *Menú setmanal.* 3. L'usuari desplega el menú lateral. 4. L'aplicació mostra el menú lateral. 5. L'usuari selecciona la pantalla *Llista de receptes.* 6. L'aplicació navega a la pantalla *Llista de receptes.* 7. L'usuari fa scroll per la pantalla buscant la recepta que vol consultar i la selecciona quan la troba. 8. L'aplicació navega a la pantalla de *Detall de recepta* on mostra la informació sobre la recepta seleccionada.
Flux alternatiu:

Nom: Log out
Descripció: L'usuari tanca la sessió i surt de l'aplicació
Actors: Usuari
Precondició: Estar identificat a l'aplicació amb credencials vàlides
Postcondició: No estar identificat a l'aplicació
Flux principal: 1. L'usuari desplega el menú lateral. 2. L'aplicació mostra el menú lateral. 3. L'usuari selecciona *Logout.* 4. L'aplicació envia la petició de tancar sessió a Firebase. 5. L'aplicació esborra les dades emmagatzemades en local. 6. L'aplicació navega a la pantalla inicial de login.
Flux alternatiu:

3.2. Disseny de l’arquitectura de l’aplicació

A continuació definim el disseny de l’arquitectura de l’aplicació, identificant les entitats, classes i objectes de la Base de Dades i de l’aplicació. També explicarem el patró de disseny que utilitzarem en la implementació de l’aplicació.

3.2.1. Base de dades

La base de dades es dividirà en quatre entitats principals, les quals són:

- Usuaris (User)
- Menú Setmanal (Week Menu)
- Menú Diari (Day Menu)
- Receptes (Recipes)

El Diagrama de la Base de dades descriu com es construirà i les relacions que hi haurà entre les diferents entitats: (Veure figura 9)

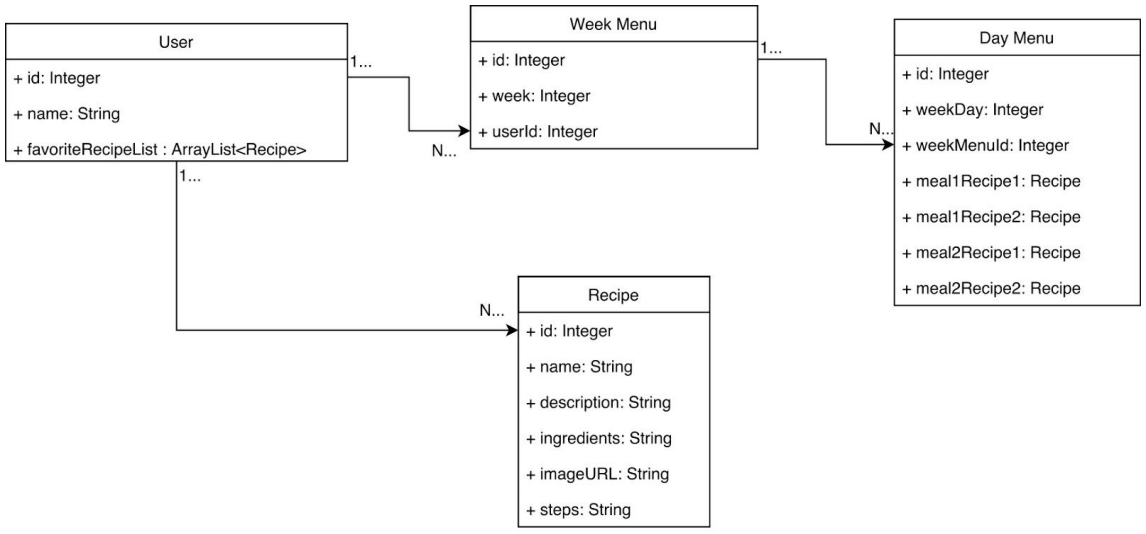


Figura 9: Diagrama de la Base de dades

La Base de dades es construirà amb Realm Database [15] , una base de dades d’objectes dissenyada específicament per dispositius mòbils.
Hem escollit Realm pels següents motius:

- *“Realm is an Object-Centric, Present-Day Database for Mobile Applications”*
    - Realm no utilitza la tecnologia SQL ja existent, sinó que treballa directament amb objectes.
- *“The Realm API is Optimized for Performance and Low Memory Use”*
    - Al ser dissenyada per dispositius mòbils, està preparada per ser eficient dins d’aquest ecosistema.

- *“Realm Offers a Multi-Platform Database - Across the Apple Ecosystem and Android”*
    - El Core de Realm està fet amb C++, fent que el seu funcionament sigui pràcticament idèntic entre iOS i Android, amb una sintaxi adaptada a cada llenguatge.
- *“The Realm SDK allows for clean and easy separation of concerns”*
    - La flexibilitat de l’SDK ajuda a generar una arquitectura sòlida, separant el codi per capes.
- *“Live Objects and Fine-Grained Notifications: Realm Updates”*
    - Realm és una “Live Object Database”, vol dir que té eines natives per notificar als objectes quan hi ha hagut un canvi, i mantenir la informació sempre fresca encara que es canviï de classe.

(Referències extretes de [16])

3.2.2. Classes

El Diagrama de classes classes defineix:

- Les Classes utilitzades a l’aplicació.
- Les propietats i els mètodes de cada classe.
- El flux de navegació entre les classes, i els punts de decisió que influeixen en aquesta navegació.

Veiem el diagrama a la figura 10:

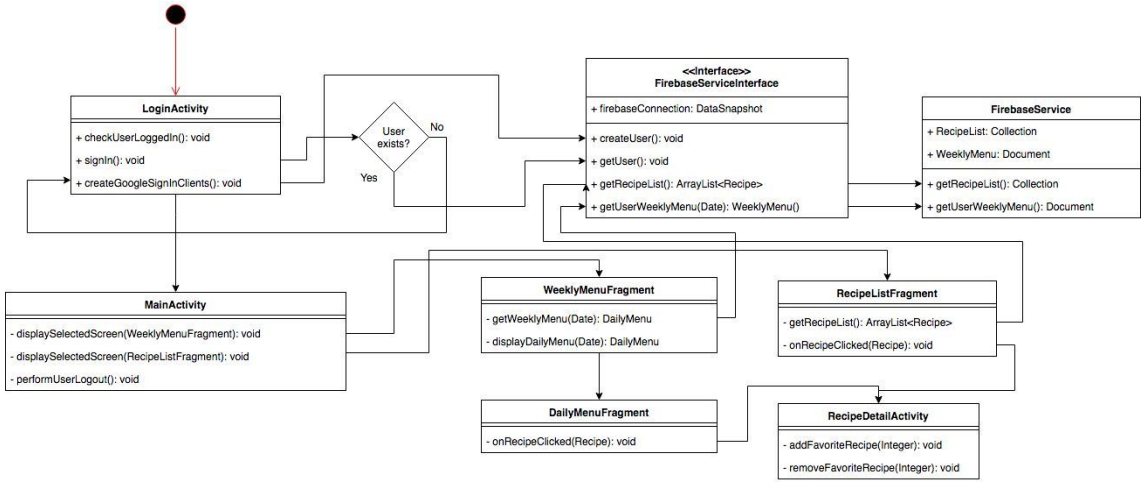


Figura 10: Diagrama de classes

3.2.3. Arquitectura del sistema

Per a implementar l'aplicació, seguirem un patró de disseny MVP (Model-View-Presenter) [17] com a disseny d'arquitectura.

Aquest model facilita les proves automatitzades i millora la lògica de presentació. Els seus components són:

- **Model**: És la interfície responsable de gestionar la informació. En el nostre cas, la informació obtinguda tant de base de dades com del servidor referent a l'Usuari, els Menús i les Receptes.
- **Presenter**: És el pont entre el Model i la Vista, qui s'encarrega d'accedir a la informació del Model i actualitzar la Vista. També és qui conté la lògica de l'aplicació. Per exemple, és qui accedeix a la base de dades mitjançant codi, o qui connecta amb Firebase per obtenir-ne la informació.
- **View**: És l'encarregada de mostrar la informació que li envia el Presenter, qui actualitza els TextViews o qui s'encarrega de donar la informació a una RecyclerView.

4. Fase d'implementació

En aquest capítol analitzarem els aspectes rellevants del codi desenvolupat així com els problemes trobats i les decisions preses.

4.1. Base de dades

Hem començat el desenvolupament de la base de dades amb l'emmagatzematge local, incloent les dependències de Realm al projecte i aplicant la configuració tal com s'indica a la documentació oficial [18].
Hem creat les 4 entitats del model de dades, Recepta, Usuari, Menú Setmanal i Menú diari, com a *open class* les quals extenen de `RealmObject`. Dins d'aquestes, per crear les relacions entre elles hem utilitzat l'anotació `LinkingObject`.

Un cop creada l'estructura de dades local hem començat amb la inclusió de Firebase al projecte per a emmagatzemar les dades en un servidor remot. Per afegir-lo hem seguit els passos que indiquen a la documentació oficial [19]: crear el projecte a la consola de Firebase, introduir el *package name* de l'aplicació i descarregar el `google-services.json`.

A l'hora de crear la *Database* dins el projecte de Firebase, vam haver de triar quin tipus de base de dades volíem: *Cloud Firestore* o *Realtime Database* (Veure figura 11).

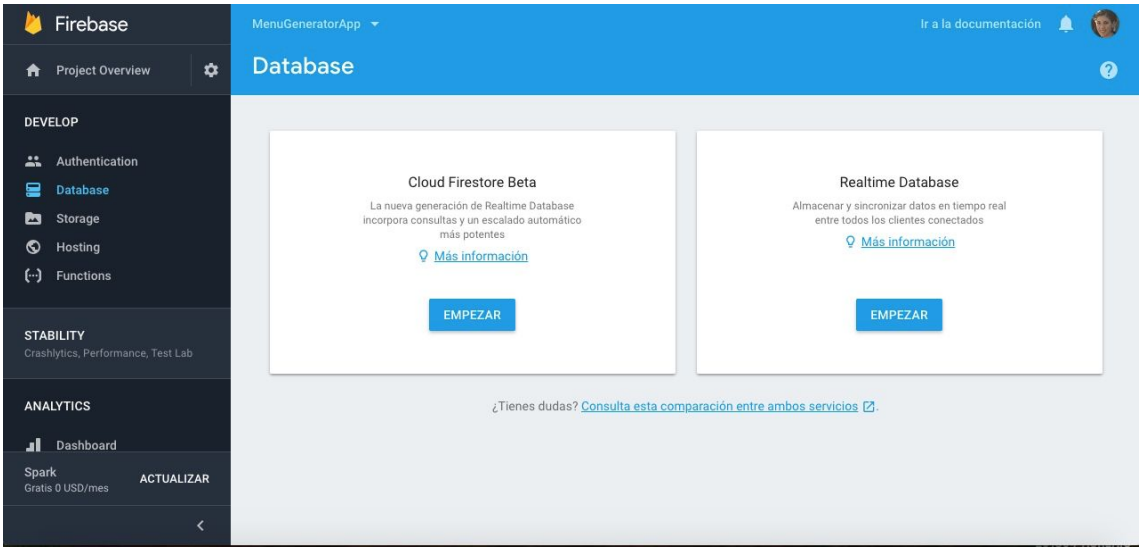


Figura 11: Tria de base de dades al inicialitzar el projecte a Firebase

La mateixa plataforma proporciona una guia amb els pros i contres de cadascuna [20], a partir de la qual hem pres la decisió d'utilitzar la primera. Els motius pels quals l'hem triat, extrets de la guia mencionada [20], són:

1. *Cloud Firestore*, tot i està en fase Beta, és una nova versió que millora alguns aspectes claus respecte a la *Realtime Database*. Millora el model de dades fent-lo més intuïtiu i permet escalar millor la quantitat de dades realitzant *queries* més complexes i de forma més ràpida.
2. Les dues permeten tenir les dades actualitzades en tot moment, és a dir, són *realtime.*
3. Les *queries* a *Cloud Firestore* tenen un temps de resposta proporcional al conjunt de dades que retornen. En canvi, a *Realtime Database,* aquest temps és proporcional al conjunt total de les dades.
4. Finalment, la plataforma Firebase recomana utilitzar Cloud Firestore per a projectes nous, si no es té inconvenient en utilitzar un producte en fase Beta. Al ser un projecte acadèmic hem considerat que és una bona oportunitat per explorar aquest nou producte i descobrir les millores respecte a la *Realtime Database.* A més, de cara al futur, quan el producte deixi d'estar en fase Beta serà més potent i fiable que la *Realtime Database*.

Dins la base de dades de tipus *Cloud Firestore* les dades s'organitzen en col·leccions i documents. Hem adaptat el nostre model de dades a aquesta estructura i ens han quedat 3 col·leccions principals que contenen els documents detallats a continuació:

- **Recipes**: Cada document té com a *key* l'ID de la recepta i conté tota la informació referent a aquesta.
- **Users**: Cada document té com a clau l'ID d'usuari, creat aleatòriament per Firebase al fer el login amb Google.
- **Weekly menu**: Col·lecció on cada document s'identifica per l'ID de l'usuari al qual pertany el menú i que conté el menú per a l'última setmana.

Per a realitzar les crides a la base de dades des de l'app hem creat l'objecte `FirebaseService`, el qual implementa els mètodes per llegir les dades, definits a la interfície `FirebaseServiceInterface`. Hem decidit que `FirebaseService` sigui un `object` [21] perquè volem que tingui una única instància, és a dir, que sigui un *Singleton* [22]. Volem remarcar que els `object` a Kotlin són *thread-safe*, és a dir, que encara que es cridin des de més d'un fil d'execució al mateix temps, sempre es farà referència a la mateixa instància [23]. Una altre avantatge és la senzillesa del codi a l'hora de cridar a una funció, ho fem directament amb el nom de l'objecte, p.ex: `FirebaseService.getRecipeListQuery()`

Un cop implementats els mètodes del `FirebaseService`, ens vam adonar que no era necessari utilitzar Realm al projecte, ja que firebase emmagatzema les dades en local i les manté actualitzades quan té connexió a internet. Utilitza un sistema de *cachés* que permet llegir, a partir de referències a la base de dades, les dades que l'aplicació ha consultat l'últim cop que ha tingut connexió a internet [24].

Per no crear inconsistències amb les dades emmagatzemades vam decidir eliminar Realm del projecte. Per aquest motiu, les classes del model de dades van passar a ser del tipus `data class`.

4.2. Estructura MVP

Com vam dir a l'apartat 3.2.3 d'aquest document, hem seguit un patró Model-View-Presenter.

La part del model, donat que tant les dades locals com les remotes provenen de Firebase, l'hem implementat a l'objecte `FirebaseService` mencionat a l'apartat 4.1.
Si el projecte fos més gran i utilitzés altres tipus d'emmagatzematge de dades, com ara les Shared Preferences, hauríem de crear els mètodes al DataManager, el qual s'hauria d'encarregar de cridar a la plataforma corresponent en cada cas. No ho hem implementat en el projecte actual per falta de temps.

Al patró MVP les *Activities* i els *Fragments* actuen com a vista d'aquest. Implementar el codi seguint aquest patró ens ha portat més temps del que havíem estimat a la valoració inicial, degut a problemes amb les dependències de Kotlin i ja que la feina prèvia d'anàlisi i d'entendre el model ha suposat més temps del previst.
Això ha fet que ens endarreríssim en la planificació i ha calgut ajustar-la. Hem decidit implementar el patró de *View-Presenter* només a les activities, per als fragments no.

Per fer-ho, hem creat la classe `BaseActivity` la qual implementa la interfície `BaseMVPView`, i totes les activities de l'aplicació tenen aquesta classe com a `super class`. Dins aquestes s'injecta el seu corresponent presenter al qual li passem la vista en el mètode `onCreate(savedInstanceState: Bundle?)` fent: `presenter.onAttach(this)`. I els desvinculem al `onDestroy()`.

Finalment tenim el presenter, del qual també hem creat una classe base, el `BasePresenter`, que implementa la interfície `BaseMVPPresenter`. En aquesta classe hem implementat les funcions relatives a la relació del *Presenter* amb la Vista, és a dir, el `onAttach(view: V?)`, el `getView(): V?` i el `onDetach()`.
Quan implementem els presenters de l'aplicació, que són subclasses del `BasePresenter`, indiquem al constructor que s'ha d'injectar a la vista fent: `@Inject internal constructor(private val mContext: Context)`.

4.3. Dagger 2

A la secció anterior hem esmentat que el *presenter* s'injecta a la vista. Per fer-ho utilitzem l'injector de dependències *Dagger 2* [25].
Hem decidit utilitzar un injector de dependències pels beneficis que presenta:

- Facilita l'accés als *Singletons*, només cal anotar-les amb `@Inject`.
- Permet fer *refactor* fàcilment, ja que és el framework qui s'encarrega de proveir les dependències.
- Facilita el *testing* de l'aplicació, ja que és molt simple substituir les dependències reals per dades *mock*.
- Simplifica l'ús de dependències complexes.

[26], [27]

Per fer funcionar les injeccions, hem hagut d'afegir un constructor anotat amb `@Inject` a totes les classes que volem que s'injectin.
Com que volem injectar objectes *custom*, hem d'indicar a Dagger com fer-ho. Hem creat mètodes anotats amb @Provides dins d'una classe amb @Module. Per exemple, en el cas del Login hem creat la classe `LoginActivityModule`:

```
@Module
class LoginActivityModule {

   @Provides
   internal fun provideLoginPresenter(presenter:
LoginPresenter<LoginMVPView>)
           : LoginMVPPresenter<LoginMVPView> = presenter
}
```

Aquestes són les accions que hem hagut de fer per a cada classe en particular que hem volgut injectar.

En termes més generals, hem creat les classes `ActivityBuilder` i `AppModule` i la interfície `AppComponent`.

En aquesta última hem afegit l'anotació `@Component` per indicar tots els mòduls que hem creat i que volem que *Dagger* tingui en compte.

A la classe abstracta `ActivityBuilder`, la qual hem anotat amb `@Module`, indiquem quins moduls contribueixen a la injecció d'Android. Per exemple, pel login tenim:

```
@ContributesAndroidInjector(modules = [(LoginActivityModule::class)])
abstract fun bindLoginActivity(): LoginActivity
```

Finalment, a la classe AppModule, tenim una funció que proporciona el Context, el qual volem que sigui un Singleton. Fem:

```
@Provides
@Singleton
internal fun provideContext(application: Application): Context =
application
```

4.4. Objectes *Parcelable*

Per fer la navegació entre la llista de receptes i el detall hem volgut afegir l'objecte *Recipe* seleccionat a l'intent per enviar la informació a la nova activity, per fer-ho, hem fet que l'objecte *Recipe* sigui *Parcelable* [28]. D'aquesta manera podem passar tot l'objecte, sense haver de fragmentar l'enviament en els diferents camps.

Per fer-ho, hem indicat que la classe Recipe implementi la `interface Parcelable,` i dins la classe hem afegit els mètodes:

- `private constructor(p: Parcel) : this()`
- `override fun writeToParcel(dest: Parcel, flags: Int)`
- `override fun describeContents()`

També hem afegit un `companion object` que conté la variable `CREATOR` la qual té els mètodes:

- `override fun createFromParcel(parcel: Parcel): Recipe`
- `override fun newArray(size: Int): Array<Recipe?>`

Hem implementat aquests mètodes seguint una guia extreta de 29.

A la llista de receptes, abans de cridar a l'activity `RecipeDetailActivity`, afegim la recepta a l'*intent* fent: `intent.putExtra(RECIPE, recipe)`.
Després, al detall, obtenim les dades de *l'intent* fent:
`recipe = intent.getParcelableExtra(RECIPE) as? Recipe`.

Seguim el mateix procediment quan cliquem a una recepta al menú setmanal per a navegar al seu detall.
També hem fet que la classe `DailyMenu` sigui *parcelable* per poder enviar les dades des de la `WeeklyMenuFragment` al `DailyMenuFragment`.

4.5. Dades *mock*

Per poder treballar amb l'aplicació sense tenir les dades reals ni el generador automàtic de menús, hem creat dades falses (*mock*).

Per fer-ho hem creat les funcions `addRecipes` i `createWeeklyMenu` al `FirebaseService`. Hem forçat la seva crida durant el desenvolupament per poblar la base de dades.
A més, la funció de crear el menú es crida quan es detecta que el menú creat no correspon a la setmana actual.

Al final del desenvolupament, rer tal que el resultat visual de l'aplicació fos més semblant al desitjat un cop es calculin els menús equilibrats, hem afegit manualment un conjunt de receptes a Firebase. Aquestes, es mostren a l'app i s'utilitzen per a generar menús aleatoris. Hem extret les receptes de la pàgina web *All Recipes* [30].

4.6. Novetats de Kotlin

Kotlin ofereix algunes novetats com a llenguatge de programació, les quals, en concret, es poden aplicar al desenvolupament d'aplicacions Android. Anem a fer un repàs d'aquestes i mostrarem fragments de codi de la nostra aplicació on les hem utilitzat.

4.6.1. Inferència de Tipus

Kotlin és capaç d'inferir els tipus d'objecte de forma automàtica. En alguns casos es poden declarar explícitament per millorar la legibilitat, per inicialitzar-los a `null` però indicant el tipus o per tenir variables amb `lateinit`, entre d'altres.

`LoginPresenter`
```
val currentUser = mAuth?.currentUser
```

Infereix el tipus `FirebaseUser` a la variable `currentUser.`

4.6.2. Inicialització de paràmetres en línia

Es permet crear valors dins els arguments d'una funció, no cal crear primer una variable per després enviar-la.
També permet indicar arguments per defecte els quals s'utilitzen en cas que no s'indiquin en alguna crida.

WeeklyMenuFragment
```
displayDailyMenu(date = Calendar.getInstance())
```

Envia un valor a `date` directament a la crida de la funció.

4.6.3. L'expressió "When"

Es canvia l'expressió "*switch*" per "*when*", molt més llegible i flexible. La sintaxi dins el *when* també ha canviat, s'utilitza el símbol "->" per separar la condició del codi que s'ha d'executar en cada cas.

MainActivity
```
when {
    reload -> Log.d(TAG, "Reloading")
    fragment != null -> loadFragment(fragment!!)
    else -> {
        finish()
        val intent = Intent(this, LoginActivity::class.java)
        startActivity(intent)
    }
}
```

Es donen tres opcions un cop entra al when: valida que el valor de reload sigui *`true`*, valida si el fragment és *`null`*, i una opció per si no es compleixen cap de les dues anteriors.

4.6.4. La Classe Data

Han introduït el tipus de classe Data, la qual és de tipus POJO (Plain Old Java Object [31]) amb els mètodes bàsics, com ara `toString()`, `equals()` o `copy()`, creats per defecte.

User
```
data class User(var id : String = "",
 var name : String = "" )
```

La classe `User` és del tipus `data class`.

4.6.5. Funcions Exteses

Són funcions associades a determinats objectes i que es poden utilitzar des de tota l'aplicació.

AppExtensions

```
fun Activity.logd(message: String){
    if (BuildConfig.DEBUG) Log.d(this::class.java.simpleName, message)
}
logd("onRecipeClicked" + recipe.toString())

fun Activity.toast(message: String){
    Toast.makeText(this, message , Toast.LENGTH_SHORT).show()
}
toast("Error: " + task.exception)
```

Són dues funcions exteses per l'Activity, una per mostrar un Log enviant només un missatge, i l'altre per enviar un missatge de Toast.

4.6.6. Null Safety

Kotlin permet distingir entre objectes nullables i els que no ho són. Per indicar que un objecte pot prendre el valor *null* afegim "?" darrere el tipus.
A l'hora d'utilitzar l'objecte, si necessitem que sigui diferent de null haurem d'utilitzar el símbol "!!"per forçar-ho. Només es poden utilitzar les exclamacions quan sabem que el valor no serà nul. D'altra banda, es considera una bona pràctica afegir un condicional de diferent de *null* abans d'utilitzar-ho, ja que d'aquesta manera evitem els *Null Pointer Exception*.

WeeklyMenuFragment

```
private var weeklyMenu : WeeklyMenu? = null
```

El "?" indica que la variable `weeklyMenu` és de tipus `WeeklyMenu` opcional i, per tant, que pot ser *null*.

4.6.7. Lambdas

Una lambda és una forma simplificada de representar una funció. Existeixen en múltiples llenguatges de programació, però especialment en Kotlin ofereixen moltes possibilitats [32].

FirebaseServiceInterface
```
fun getRecipeList(recipeListGotten: (recList: ArrayList<Recipe>?) ->
Unit)
```

Funció lambda que obté la llista de receptes.

4.6.8. Kotlin Android Extensions

Són un plugin de Kotlin que permet recuperar vistes sense haver de fer el "`findViewById`", generant codi de forma automàtica que permet accedir a les vistes de l'xml com si fossin propietats.

DailyMenuFragment
```
menu_title_1.setBackgroundResource(R.drawable.layout_white_bottom_bord
er)
```

fragment_daily_menu
```
<include
    android:id="@+id/menu_title_1"
    layout="@layout/menu_title_row"
    ...
```

El Fragment accedeix a l'ítem `menu_title_1` directament, sense findViewById, i li canvia el fons.

Alguns d'aquests punts els hem extret de [33].

4.7. Càrrega d'imatges

Per a la càrrega d'imatges des d'una URL hem utilitzat la llibreria Picasso [34], la qual permet descarregar, mostrar i guardar en *caché* les imatges.

Utilitzar aquesta llibrera permet reduir el consum de dades de l'aplicació, ja que no s'ha de descarregar la imatge cada vegada que s'ha de mostrar.

4.8. Reajust de tasques

En el desenvolupament ens hem trobat algun entrebanc que no havíem previst com ara la inconsistència entre les diferents BBDD explicada a la secció 4.1. i algunes tasques ens han costat més temps de l'estimat, com per exemple, la maquetació del menú setmanal. Això ens ha obligat a reajustar les tasques i ha fet que canviés l'abast del projecte. S'ha vist afectada la funcionalitat d'indicar i emmagatzemar receptes favorites, la qual hem descartat.

4.9. Control de versions

Durant tot el procés d'implementació de l'aplicació, hem utilitzat l'eina de control de versions Git [35] i el repositori Bitbucket [36]. Gràcies a aquesta hem pogut desfer algun canvi que havia sigut erroni i hem pogut veure el detall dels canvis que afegíem dia a dia.

Per a realitzar les accions habituals a l'hora de treballar amb control de versions, pujar o baixar els canvis al repositori, ajuntar dues branques, etc., podem fer-ho o bé per línia de comandes o bé mitjançant algun programa d'escriptori. Nosaltres hem utilitzat el programa Source Tree [37] el qual presenta una interfície molt intuïtiva i facilita les tasques de control (veure figura 12).

És una bona pràctica utilitzar aquest tipus d'eines, ja que permeten consultar l'estat del codi en diferents etapes del desenvolupament. A més, a l'hora de treballar en equip permet paral·lelitzar tasques donant l'opció d'ajuntar la feina després.

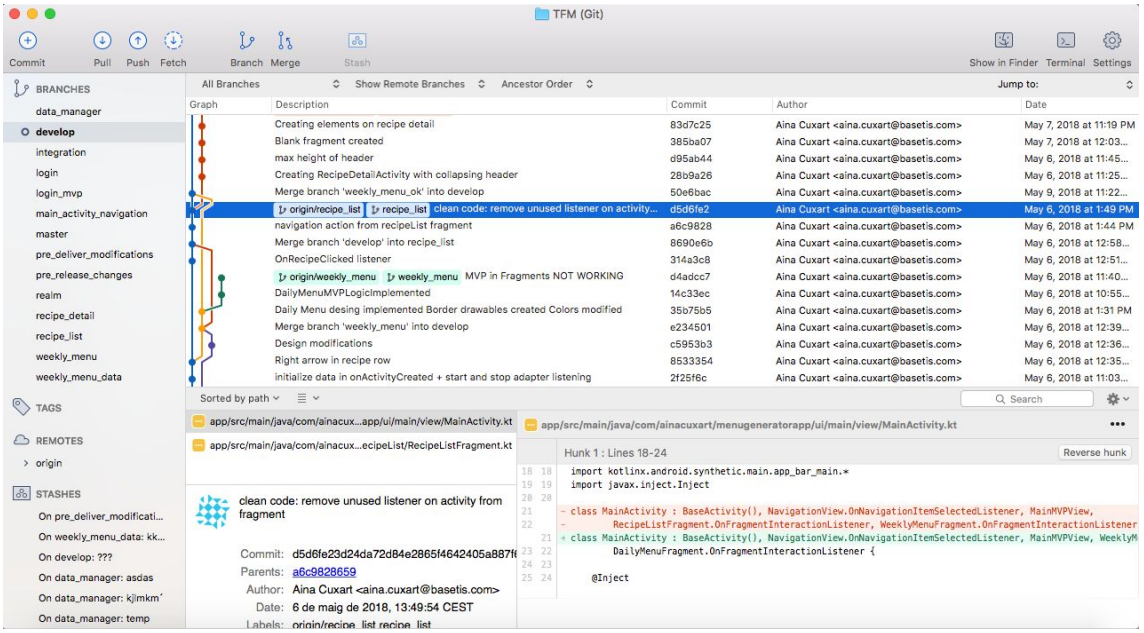


Figura 12: Captura de pantalla del Source Tree

5. Joc de proves

En aquest capítol definirem el conjunt de proves que ha de superar l'aplicació per a tenir un bon funcionament.

Pantalla	Comportament esperat
Login	Si és el primer cop que l'usuari entra a l'aplicació, li apareix la pantalla de login.
Login	Si l'usuari ja s'havia identificat, a l'entrar a l'app navega directament a la pantalla del menú setmanal.
Login	En clicar al botó "Sign In" s'obre l'intent de Google per triar el compte d'usuari.
Login	En tornar de l'intent de Google si l'usuari és correcte, es navega automàticament a la pantalla del menú setmanal.
Menú setmanal	Si és el primer cop que l'usuari accedeix a l'app en la setmana actual, es crea un menú, s'envia a Firebase i es mostra.
Menú setmanal	Si no és el primer cop que l'usuari accedeix a l'app en la setmana actual, es descarrega el menú de Firebase i es mostra.
Menú setmanal	En seleccionar un dia, es mostren les receptes per aquell dia.
Menú setmanal	En clicar sobre una de les receptes es navega al seu detall.
Menú setmanal	En clicar a la icona "d'hamburguesa" de la part superior esquerra es desplega el menú lateral.
Menú lateral	En seleccionar "Weekly Menu" o "Recipes" es navega a la pantalla corresponent.
Menú lateral	En seleccionar "Logout" es tanca la sessió i es navega a la pantalla de Login.
Receptes	Es mostren les receptes i es pot fer scroll per veure-les totes.
Receptes	En clicar sobre una recepta es navega al seu detall.
Detall de recepta	A l'entrar al detall la imatge de la capçalera es veu en gran.
Detall de recepta	La pantalla permet fer scroll i en fer-ne la imatge es comprimeix deixant més espai pel text.

6. Conclusions

Un cop finalitzada la implementació de l'aplicació, la sensació general és de satisfacció amb la feina feta i amb els coneixements adquirits. A continuació faré un repàs de si s'han assolit els objectius, en quins aspectes he après i si la planificació era bona. Mostraré també algunes imatges de l'aplicació creada i exposaré les línies de treball futur de què ha quedat pendent així com amb idees que no s'han pogut aplicar.

Pel que fa als objectius, podem dir que els he assolit quasi al 100%:

- He creat una aplicació per a dispositius Android, programada en Kotlin que està preparada per oferir un menú a l'usuari. Conté diferents pantalles que indiquen quines receptes s'han de preparar per a cada àpat i el detall de com fer-ho.
- He afegit login a l'aplicació permetent a l'usuari tenir-la en més d'un dispositiu i conservant la informació.
- En descriure el disseny de l'aplicació, he tingut en compte les guies d'estil i usabilitat d'Android, com es reflecteix al capítol 2 d'aquesta memòria.
- He definit l'abast del projecte i planificat les tasques, tal com veiem sintetitzat al Diagrama de Gantt de la figura 1. Tot i això, l'ordre seguit en el desenvolupament no ha sigut l'indicat degut a dependències no previstes entre diferents parts del projecte. Algunes de les tasques plantejades s'han indicat com a treball futur, a causa de la falta de temps. En trobem un resum a la taula 1. En aquest capítol faré una anàlisi més exhaustiu de les possibles tasques de continuació del projecte.

Amb la realització d'aquest projecte he millorat en l'àmbit tècnic aprofundint en els coneixements d'Android i Kotlin, en concret:

- He ampliat el coneixement del lleguantge Kotlin.
- He aprofundit en l'estructura *Model-View-Presenter*.
- He après a utilitzar l'injector de dependències Dagger2, el qual no coneixia.
- He descobert la nova base de dades que ofereix Firebase, Cloud Firestore, la qual he utilitzat per emmagatzemar les dades de l'aplicació al *cloud.*
- He experimentat les incompatibilitats entre Realm i Firestore, alhora que aprofundia en els coneixements de la primera.

També he après altres aspectes més transversals:

- En començar un projecte des de zero, he après a definir i detallar bé els requeriments així com a plantejar l'arquitectura i tots els components necessaris per al desenvolupament.
- Durant la implementació, he après a adaptar la planificació i l'ordre de les tasques segons les dependències que han sorgit i no s'havien tingut en compte.
- He après a justificar les decisions tècniques preses durant el desenvolupament per tal de poder plasmar-les a la memòria.
- He après a treballar de forma autònoma buscant recursos per solucionar els entrebancs.

Durant la implementació he fet petites modificacions a la planificació degut a imprevistos que no havíem tingut en compte i a dependències no previstes.

Vaig començar creant l'entorn de les dades a Firebase, però no vaig introduïr totes les dades ja que em va semblar més eficient implementar l'escriptura de dades des de l'aplicació i automatitzar-ne la creació. En els següents passos vaig seguir la planifcació: crear el projecte d'Android, connectar-lo amb el de Firebase i preparar l'entorn per a les crides, fent algunes proves per veure que funcionava la comunicació.

Abans de realitzar la lectura i escriptura definitiva des de Firebase vaig fer la tasca d'introduïr una BBDD per emmagatzemar les dades en local. Aquesta tasca l'havíem estimat en 16 hores però em va costar una mica més del doble, fet que em va provocar un retràs en la planificació.

Un cop finalitzades les crides de comunicació amb Firebase vaig començar amb la implementació de les diferents pantalles de l'aplicació. Primer menú lateral (veure figura 13) i després la pantalla del menú setmanal. La maquetació d'aquesta també em va costar una mica més de lo estimat, degut a la complexitat dels elements que hi trobem (veure figura 14).

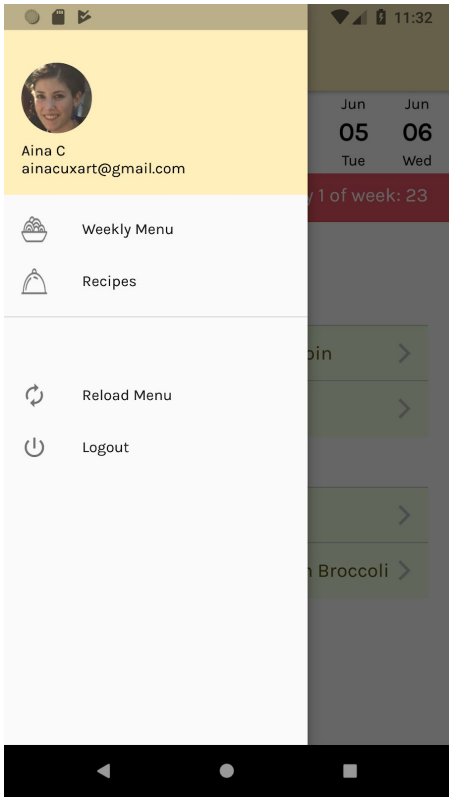


Figura 13: Menú lateral

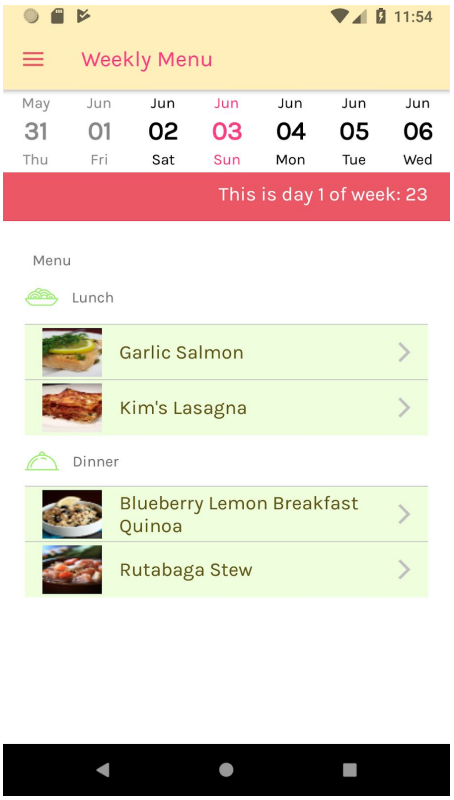
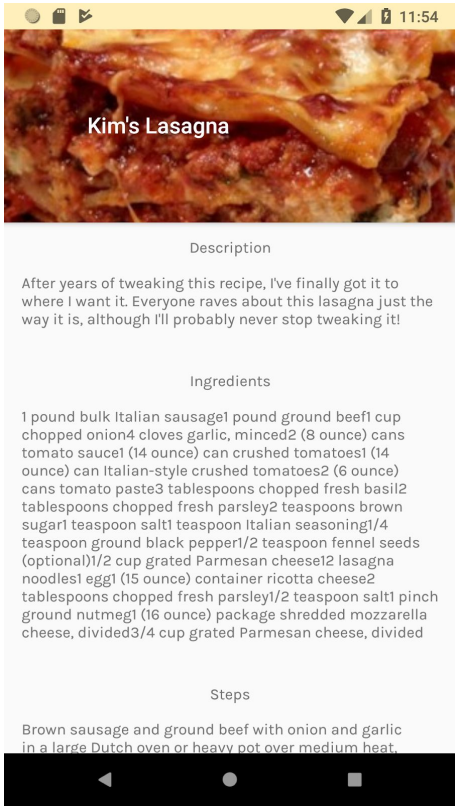


Figura 14: Menú setmanal

Després vaig implementar la pantalla de detall d'una recepta (veure figura 15), la qual havíem planificat bé i finalment la llista de receptes (veure figura 16). Abans de començar-ne el desenvolupament, vaig fer un repàs de la planificació i vaig veure que no arribàvem a tot a causa dels petits endarreriments que havia tingut els quals s'havien anat acumulant. Vaig considerar més important poder definir les proves de l'aplicació que fer la gestió de receptes favorites, per això vaig decidir no incloure aquest desenvolupament.



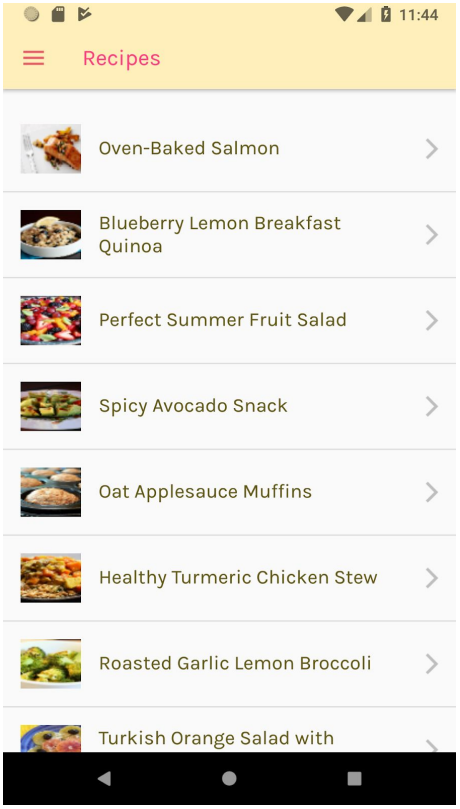


Figura 15: Detall de recepta **Figura 16**: Llista de receptes

Finalment vaig definir i passar el joc de proves, sense detectar cap error.

Les línies de treball futur comencen per incloure la gestió de receptes favorites la qual he hagut de descartar per falta de temps durant la implementació. També s'hi inclouen les tasques que havíem pensat que havia de tenir l'aplicació però que no vam incloure durant la fase de planificació:

- Fer que el menú sigui equilibrat.
- Oferir una base de dades de receptes.
- Generar una llista de la compra editable a partir del menú planificat.
- Permetre a l'usuari crear un entorn compartit per visualitzar la mateixa informació que els seus companys o familiars.

Noteu que algunes d'aquestes tasques no són de l'àmbit de l'aplicació però que són necessàries per al correcte funcionament d'aquesta. En concret, fer que el menú sigui equilibrat és de l'àmbit de l'algorísmia i el big data, i crear un entorn compartit a l'usuari és una tasca de l'api que proporciona les dades a l'app.

A part d'aquestes tasques, tinc altres idees que caldria estudiar si tenen encaix i sentit dins l'aplicació, per exemple:

- Incloure un perfil d'usuari per tal de poder gestionar les dades personals i els entorns compartits.
- Connectar amb un supermercat online per realitzar la compra dels elements de la llista.

Aquest treball ha sigut un repte personal molt gran ja que el món d'Android és molt desconegut per mi, tant a nivell d'usuari com a nivell de programació. Alhora, crec que és molt interessant tenir coneixements d'aquesta plataforma, ja que em faran créixer a nivell professional com a *Mobile Developer*.

Donada la poca experiència que tenia en Android, estic molt satisfeta del resultat obtingut.
Espero poder reprendre aquest projecte en un futur per poder completar l'aplicació i fer que es pugui publicar.

7. Bibliografia

1. Nootric a la Play Store: https://play.google.com/store/apps/details?id=com.arcadiaseed.nootric (13/03/2018)
2. Lifesum a la Play Store: https://play.google.com/store/apps/details?id=com.sillens.shapeupclub (13/03/2018)
3. Runtasty a la Play Store: https://play.google.com/store/apps/details?id=com.runtastic.android.runtasty.lite (13/03/2018)
4. Big Oven a la Play Store: https://play.google.com/store/apps/details?id=com.bigoven.android (13/03/2018)
5. Nestlé cocina a la Play Store: https://play.google.com/store/apps/details?id=com.mubiquo.nestlecocina (13/03/2018)
6. Native vs hybrid apps: https://blog.techmagic.co/native-vs-hybrid-apps/ (12/03/2018)
7. Hybrid vs native apps: https://ymedialabs.com/hybrid-vs-native-mobile-apps-the-answer-is-clear
8. Wikipedia de Kotlin https://en.wikipedia.org/wiki/Kotlin_(programming_language) (12/03/2018)
9. Batalla Oracle (Java) vs Google: https://www.theregister.co.uk/2017/05/25/your_roadmap_to_the_google_vs_oracle_java/ (12/02/2018)
10. 12 Arguments per Utilitzar Kotlin: https://devexperto.com/12-razones-usar-kotlin-android/ (12/02/2018)
11. Marvel App: https://marvelapp.com (14/03/2018)
12. Navigation Pattern Android: https://material.io/guidelines/patterns/navigation.html#navigation-patterns (01/04/2018)
13. Bottom Navigation Android: https://material.io/guidelines/components/bottom-navigation.html#bottom-navigation-usage (01/04/2018)
14. Floating Button Android: https://material.io/guidelines/components/buttons-floating-action-button.html#buttons-floating-action-button-floating-action-button (01/04/2018)
15. Realm Database: https://realm.io/products/realm-database/ (03/04/2018)
16. Learning Path: What Makes Realm Different? https://academy.realm.io/posts/learning-path-what-makes-realm-different/ (03/04/2018)

17. Model-View-Presenter: Android guidelines: https://medium.com/@cervonefrancesco/model-view-presenter-android-guidelines-94970b430ddf (04/04/2018)
18. Realm Database for Java: https://realm.io/docs/java/latest/ (07/04/2018)
19. Firebase, Android Setup: https://firebase.google.com/docs/android/setup (17/04/2018)
20. Cloud Firestore Beta vs Realtime Database: https://firebase.google.com/docs/firestore/rtdb-vs-firestore?authuser=1 (17/04/2018)
21. Object Declarations in Kotlin: https://kotlinlang.org/docs/reference/object-declarations.html (18/04/2018)
22. Singleton Patern: https://en.wikipedia.org/wiki/Singleton_pattern (18/04/2018)
23. Objects in Kotlin, Antonio Leiva: https://antonioleiva.com/objects-kotlin/ (18/04/2018)
24. Cloud Firestore Offline Data: https://firebase.google.com/docs/firestore/manage-data/enable-offline (22/04/2018)
25. Dagger 2: https://google.github.io/dagger/ (30/04/2018)
26. An intro to dependency injection for Android: https://medium.com/rocknnull/the-lost-droid-and-the-magic-dagger-an-intro-to-dependency-injection-for-android-c686f4399117 (30/04/2018)
27. What is Dagger and why we use it: https://stackoverflow.com/questions/41820133/what-is-dagger-and-why-we-use-it (30/04/2018)
28. Parcelable, Android: https://developer.android.com/reference/android/os/Parcelable (07/05/2018)
29. Reducing Parcelable boilerplate code using Kotlin: https://medium.com/@BladeCoder/reducing-parcelable-boilerplate-code-using-kotlin-741c3124a49a (07/05/2018)
30. All recipes, healthy recipes: https://www.allrecipes.com/recipes/84/healthy-recipes/ (29/05/2018)
31. Plain Old Java Object: https://ca.wikipedia.org/wiki/Plain_Old_Java_Object (30/04/2018)
32. Lambdas in Kotlin: https://antonioleiva.com/lambdas-kotlin/ (30/04/2018)
33. Why you should totally switch to Kotlin: https://medium.com/@magnus.chatt/why-you-should-totally-switch-to-kotlin-c7bbde9e10d5 (30/04/2018)
34. Picasso: http://square.github.io/picasso/ (09/05/2018)
35. Git in Wikipedia: https://en.wikipedia.org/wiki/Git (29/05/2018)
36. Bitbucket: https://bitbucket.org/product (29/05/2018)
37. Source Tree: https://www.sourcetreeapp.com (29/05/2018)

8. Glossari

API: Una API és un conjunt de comandes, funcions, protocols i objectes que els programadors poden utilitzar al crear software per interaccionar amb un sistema extern.

Caché: La caché emmagatzema les dades que l'aplicació ha consultat recentment per poder-hi accedir ràpidament més tard.

Dades mock: Diem que un conjunt de dades és mock quan no són dades reals, és a dir, són dades generades automàticament amb la finalitat de poblar la BBDD per a treballar durant el desenvolupament.

IDE: Sigles de "Integrated Developement Environment", entorn de desenvolupament integrat. Ens referim per IDE als editors de codi que estan adaptats al llenguatge que utilitzem i que ofereixen facilitats per fer-ho.

Interfície: Una interfície conté funcions abstractes les quals han de ser implementades per la classe que implementi la interfície.

Multidispositiu: Diem que una aplicació és multidispositiu quan l'usuari pot accedir a l'aplicació veient les mateixes dades des de més d'un dispositiu.

Package name: El package name d'una aplicació Android és el seu identificador únic que la diferenciarà de la resta d'aplicacions del mercat.

Play Store: És la plataforma o mercat oficial de Google on trobar aplicacions per Android així com jocs, música, pel·lícules, etc.

Queries (BBDD): Les queries a una base de dades són la manera de consultar la informació que hi ha emmagatzemada. La paraula query és sinònima de "pregunta".

Scroll: Diem que fem scroll pel contingut de la pantalla quan el desplacem.