Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: APLICACIÓ MÒBIL PER SISTEMA OPERATIU ANDROID: SUMPLAPP

Document: Memòria

Alumne: Marc Planas Quintanas

Tutor: Ignacio Martín Campos
Departament: Informàtica, matemàtica aplicada i estadística
Àrea: Llenguatges i sistemes informàtics

Convocatòria (mes/any): Setembre 2022

Universitat de Girona
Escola Politècnica Superior

APLICACIÓ MÒBIL PER SISTEMA OPERATIU ANDROID: SUMPLAPP

PROJECTE DE FI DE GRAU

Autor:

MARC PLANAS QUINTANAS

Setembre 2022

Grau en enginyeria informàtica

Tutor:

IGNACIO MARTÍN CAMPOS

ÍNDEX

1. INTRODUCCIÓ

1.1 Origen de la idea de SUMPLAPP

Un estiu mentre treballava a l'empresa del meu pare, la qual es basa en el manteniment de descalcificadors de diverses comunitats, vaig detectar que una de les tasques que fan una vegada acaben el manteniment d'aquests, és apuntar-se manualment en un paper el servei que acaben de realitzar, per tal que ell en tingui constància i més endavant pugui fer la corresponent factura del servei proporcionat. Al principi em va semblar que era una clàssica però alhora bona metodologia per utilitzar, i que a més no suposa haver de tenir massa problemes durant el seu ús, per tant, en cap moment la vaig descartar. No obstant això, quan ja feia unes setmanes que estava col·laborant amb la feina de l'empresa familiar vaig començar a donar voltes al tema, fins que finalment em vaig adonar que tot i no ser una mala metodologia, requereix més temps per aplicar-la, a diferència d'altres que a part de ser senzilles, són més eficaces i, per tant, se'n pot treure un millor rendiment en la gestió del temps laboral.

Aquesta reflexió em va fer qüestionar, el següent: "si deixo la finestra del cotxe oberta, no me n'adono i per casualitat fa vent i els papers que hi pugui tenir anotacions respecte als manteniments realitzats surten volant, què faré? Com ho recuperaré?" o "si com és molt probable que a tots ens pugui passar, acabo perdent algun paper perquè no recordo on l'he deixat i ara ja ves a saber on estarà…" Aquestes suposicions tant obvies i moltes més altres que s'hi assemblin, són errors que els humans podem cometre amb facilitat davant circumstàncies de descuit, les quals algunes d'elles potser es podrien evitar canviant la metodologia de treball. La meva reflexió va ser el que llavors vaig compartir amb la meva família, i el que justament ens va donar peu a replantejar-nos què podríem fer per resoldre les qüestions hipotètiques inicials del projecte. Després de valorar pros i contres de diferents opcions que havíem plantejat entre tots, finalment vam optar per la idea de fer una aplicació per mòbil que permetés fer aquest tipus de gestions, la qual llavors implicaria poder oferir al meu pare una metodologia més senzilla i eficaç per fer les anotacions dels manteniments de descalcificadors portats a cap.

Va ser posteriorment a la presa d'aquesta decisió, quan em vaig posar a investigar quin tipus d'aplicació seria la més convenient i que més s'ajusta a la feina que realitza el meu pare, així com tenint clara la metodologia que ell segueix, i amb aquesta idea en ment, he acabat creant una aplicació anomenada **SUMPLAPP** al llarg del meu Treball Final de Grau.

1.2 Motivacions que van impulsar SUMPLAPP

És evident que cada persona segueix les seves motivacions i cadascú valora més uns aspectes que altres. En el meu cas per exemple, la motivació intrínseca que m'ha impulsat a voler dur a terme aquest treball és molt clara, i parteix d'un plantejament molt lògic: què hi ha millor que

poder ajudar contribuint en la millora de la feina d'una persona important de la família, i a més poder-ho fer aplicant gran part de coneixements adquirits al llarg de la carrera durant tots aquests anys? En el fons, no negaré que aquesta era la millor "excusa" per poder posar en pràctica tot el que m'han ensenyat durant tots aquests anys que he estat estudiant el Grau d'Enginyeria Informàtica, i a sobre tenir la satisfacció personal de poder facilitar la feina a la meva família, que són les persones que més m'han estat ajudant i donant suport com ningú al llarg de tot aquest temps. Dit d'una altra forma, podria concloure que les meves motivacions a l'hora de fer aquest treball han estat el fet de voler "matar dos ocells d'un sol tret". A més, des de petit sempre m'ha entusiasmat la informàtica, i és per aquest motiu que m'engrescava la idea de poder fer alguna aplicació per mòbil en la qual qualsevol persona/usuari hi pugues interactuar i treure'n un òptim profit. Sempre havia tingut curiositat per crear una aplicació, i de fet tenir ganes d'aprendre'n. Així doncs, el meu interès per la informàtica des de petit i el fet de voler crear una aplicació, són un dels motius principals que em van portar a endinsar-me en aquesta carrera i no una altra. Per tant, haver fet l'aplicació *SUMPLAPP* en aquests moments per mi ha estat el primer pas per aprendre a fer-ne agafant les bases i a partir d'aquí, més endavant poder-ne fer de més professionals, una vegada hagi fet més pràctica la qual m'hagi permès adquirir més experiència realitzant aquest tipus de projectes.

1.3 Quin propòsit hi ha en aquesta aplicació?

El propòsit que hi ha davant de l'aplicació SUMPLAPP és molt simple: Aconseguir millorar una de les moltes tasques que ha de realitzar un familiar en la seva empresa, i aquesta consisteix en el seguiment de manteniments de descalcificadors realitzats. Per tal d'assolir-lo, s'ha volgut crear una aplicació que pugui utilitzar-se amb el mòbil, la qual garanteixi una notable millora en la seva metodologia de treball, tant en l'àmbit d'assoliment de la tasca com una millora en la gestió del temps molt, i que, per tant, sigui molt més eficient per l'empresari.

Per poder-ho entendre correctament, cal remarcar el principal problema que presenta la metodologia de la feina en qüestió, que consisteix en el següent: cada vegada que duien a terme un servei a una comunitat, havien d'apuntar en un full quin servei s'havia fet per posteriorment, fer-ne la factura. Tot i que és una metodologia totalment vàlida i poc problemàtica, vaig decidir marcar-me el propòsit d'elaborar una aplicació telefònica que permetés facilitar el procés de gestió ja sigui de tasques com la d'apuntar i entrar un nou servei, o bé la de poder consultar algun servei que s'ha realitzat anteriorment, sense haver d'estar buscant enmig de tanta paperassa ni carpetes de factures que arriba a tenir una empresa, ja que a més de ser complicat, tot el temps que s'ha d'invertir a buscar, podria ser invertit a uns altres tipus de tasques que requereixen més dedicació directa del professional.

1.4 Objectius del projecte

Una vegada pensada la idea inicial del projecte i els propòsits que tenia marcats darrere d'aquesta, em va tocar pensar en els objectius que necessitaria assolir per tal de poder dur a terme aquest treball. Primerament, fer un esbós de la manera com podria dividir el projecte, i ho vaig fer en petites tasques mitjançant el mètode de "Divideix i venç", que, tal com el seu nom indica, es basa en fraccionar cada una de les parts a realitzar en diversos fragments més simples i petits per així aconseguir una visió més àmplia i més simplificada d'una sola tasca que, una vegada s'hagin completat totes les seves parts, el resultat obtingut serà global i unitari.

A partir d'aplicar aquest sistema, el primer que vaig fer va ser pensar en quines dades necessitaria recollir mitjançant aquesta aplicació i una vegada pensades i triades, plantejar-me també qui serien els seus "actors" o "protagonistes" que en formarien part. Una vegada resoltes aquestes dues parts més bàsiques, vaig passar a plantejar-me quines accions necessitaria dur a terme l'usuari que utilitzi l'aplicació, per tal de poder dur a terme la tasca en qüestió.

Així doncs, els objectius que havia d'assolir a través de l'aplicació van ser els següents:

-Permetre que qualsevol persona que tingui interès a utilitzar-la, pugui crear un compte i també esborrar-lo sempre que ho consideri convenient.

-Garantir que l'usuari pugui iniciar i tancar sessió.

-Oferir la possibilitat de poder actualitzar el correu o la contrasenya a l'usuari, sigui perquè ha canviat de correu o perquè ha perdut la contrasenya.

-Donar permís a l'usuari perquè pugui crear una comunitat, consultar les dades de qualsevol comunitat, modificar les dades de qualsevol comunitat si s'escau, i també perquè pugui esborrar una comunitat si ja no li realitzen un manteniment del seu descalcificador.

-Facilitar que l'usuari pugui veure a on està situada la comunitat mitjançant una eina de *Google Maps* (geocoding).

-Permetre que l'usuari pugui inserir nous serveis que s'han fet a qualsevol comunitat, així com que pugui consultar les dades d'aquestes o editar-les, i també que pugui esborrar un servei d'una comunitat si així ho desitja.

-Possibilitar l'opció de descàrrega dels serveis d'una comunitat amb les seves dades pertinents.

2. ESTUDI DE VIABILITAT

2.1 Recursos pel desenvolupament de l'aplicació

L'aplicació s'ha pogut dur a terme sense necessitat d'utilitzar eines molt específiques o difícils d'obtenir, ja que m'he basat en fer servir *Android Studio* durant tota la creació i implementació d'aquesta, a més de diversos recursos que he anat extraient d'Internet, com ara components o, fins i tot, eines de *Google*. Així doncs, la viabilitat tecnològica de la qual s'ha disposat ha estat tota aquella que m'ha ofert Internet, juntament amb *Android Studio* i una *API de Google* que s'usa com a bases de dades de l'aplicació.

Referent a la seva viabilitat econòmica, aquesta és una aplicació nativa que està encarada a un sistema operatiu en concret (*Android*), i, per tant, això garanteix que l'aplicació sigui de major qualitat fent que tingui un rendiment i prestacions més potents. Com a resultat, tot això suposa que l'aplicació sigui més costosa, ja que aquest tipus d'aplicacions requereixen una implicació més gran a l'hora de dissenyar-la i desenvolupar-la, pel fet que la seva base i disseny són més sòlids i ampliables per tal que siguin aprofitables de cara al futur. Tenint en compte els diferents rangs de programadors, s'estima que aquesta aplicació pertany a un freelance amb poca experiència adquirida, i és per això mateix que el seu preu base s'ajusta a uns 30 euros l'hora. En total, s'han comptabilitzat unes 110 hores de treball destinat a la programació de l'aplicació, i fins a un total de 20 hores més, destinades a realitzar proves de les diferents funcions que ofereix per tal d'evitar possibles anomalies i mal funcionament de l'aplicació.

Per tant, es preveu un cost total de **3900 €**, distribuïts de la següent manera; **3300 €** a la programació (30 €/h * 110 h) més un total de **600 €** al llarg de totes les proves realitzades (30 €/h* 20 h).

Com que el *backend* i el *frontend* utilitzat no requereixen una autorització de pagament, la seva autorització d'entrada no faria elevar el preu de l'aplicació, a menys que l'aplicació s'arribés a fer servir per a un nombre d'usuaris molt elevat, ja que llavors *Firebase* passaria a demanar una taxa destinada a la realització del manteniment adient. No obstant això, aquesta taxa també variaria segons el nombre d'usuaris o del nombre de descàrregues que tingués l'aplicació.

3. METODOLOGIA

3.1 SCRUM

Per dur a terme aquest treball s'ha fet servir un marc de treball anomenat *SCRUM*, el qual és una metodologia molt adient per treballar en conjunt i en equip, i així acabar obtenint un millor resultat. Tot i que la idea és fer *SCRUM* en equip, en aquest cas s'ha utilitzat perquè en el meu lloc de feina és una metodologia que s'empra de manera abundant i dóna molt de suport en l'àmbit organitzatiu de les tasques de manera fent que quedin més clares i sintetitzades. A més, permet fer una millor planificació prèvia i això facilita part de la feina.

Normalment, quan s'opta per *SCRUM* s'efectuen entregues parcials del projecte. D'aquesta manera, el client pot tenir accés a les diferents entregues i, per tant, pot veure els diferents avenços que s'han anat fent. En el cas de les entregues d'aquest projecte, s'han dividit en 3 fases: la fase inicial, la fase de desenvolupament i la fase final. Aquestes entregues han permès en tot moment, que el client hagi pogut disposar d'un seguiment de l'evolució sobre com d'avançada estava l'aplicació.

3.2 En què es basa i què hi guanyem amb SCRUM?

3.2.1 Bases de SCRUM

Tot i que hi ha altres mètodes com *Waterfall* entre d'altres que es poden utilitzar, l'ús de la metodologia *SCRUM* et permet desenvolupar diferents requisits de l'aplicació en tasques més curtes i més sintetitzades, donant una visió més clara del seu requisit que s'haurà de fer i que caldrà implementar. Tanmateix, els seus requisits segons el seu ordre descendent d'importància, però sempre tenint en compte l'opinió del client en qüestió.

A més, permet mostrar-li en qualsevol moment com d'avançada està l'aplicació una vegada feta una entrega, i conseqüentment, una vegada s'ha fet l'entrega permet que el pugui veure com aquesta ha quedat i, si s'escau, suggerir nous canvis o bé arreglar o afegir tot allò que consideri oportú. D'altra banda, *SCRUM* dóna l'opció de poder fer canvis en alguns dels requisits si s'escau, i que per cada una de les iteracions que s'hi afegeixin es pugui analitzar si l'efecte que provoca sobre aquests, és positiu o negatiu. Aquests factors donen molta adaptabilitat a la metodologia, i donen la possibilitat de poder fer molts més canvis que siguin necessaris i amb menys anomalies que interfereixin.

3.2.2 Beneficis de l'SCRUM

Aquests factors afavoreixen que s'obtinguin resultats anticipats i de major qualitat, a més permet que el programador pugui treballar segons l'opinió i el suggeriment del client partint d'una referència clara de què li demana. També garanteix una major confiança de treball en la relació de client – treballador, i pugui afrontar el projecte d'una manera més còmoda. Tot i això, cal destacar aquesta forma de treballar mitiga notablement els riscos que puguin sorgir al llarg del projecte, sobretot en termes de funcionalitat i requisits que demani el client sobre l'aplicació.

3.3 Com s'aplica SCRUM?

Per tal d'aplicar *SCRUM*, el primer que es va fer va ser parlar amb el client amb la intenció de deixar clares les funcionalitats i quins requisits haurien de constar en l'aplicació. Tot i això, com que era un sol desenvolupador qui desenvolupava aquesta aplicació, es va acordar que les entregues s'anirien fent per fases, i que en cada fase es faria mínim una reunió per poder mostrar al client l'avenç del projecte. Pel que fa a les entregues, es va arribar a l'acord mutu amb el client de què no hi hauria una entrega periòdica cada setmana, sinó que es farien per **blocs de feina** i que estarien dividits en 3, que són: **Perfil, Comunitats i Serveis**. Cal destacar que l'única data que finalment vaser acordada va ser l'entrega final de l'aplicació, el dia **23 de setembre de 2023**. Això no obstant, encara ens mantenim a l'espera de saber amb certesa que es podrà allargar aquesta data final, ja que recentment, hi ha hagut noves demandes d'implementacions que no havien estat estipulades en els requisits inicialment acordats amb el client.

3.4 Gestió de les tasques del projecte

Aquesta gestió s'ha realitzat mentre es feien les reunions amb el client. Durant aquestes, apuntava en un Bloc de notes els requisits que em sol·licitava, juntament amb les dades que ell demanava. D'aquesta manera, si apareixien dubtes sobre alguna de les funcionalitats que s'estava desenvolupant o de qualsevol dada que s'hagués de tenir en compte, se li ensenyava el document posteriorment escrit, per tal que tingués clar la funcionalitat que s'estava implementant en aquell instant:

ACTORS:

- USUARI
- COMUNITATS
- SERVEIS

FUNCIONALITATS:

- REGISTRE
- INICI SESSIÓ (Correu/Contrasenya)
- TANCAR SESSIÓ
- ELIMINAR CONTA USUARI
- INSERTAR NOVA COMUNITAT (IdCom, Nom carrer, Nº, Planta, Codi Postal, Població) -> A MÉS A MÉS, TAMBÉ CALDRIA AGREGAR Dades de la persona encarregada de la comunitat (President de la Comunitat: Nom, Cognom, Telèfon)
- ELIMINAR COMUNITAT
- MODIFICAR DADES COMUNITATS
- SELECCIONAR DATA, SELECCIONAR COMUNITAT I INSERIR SERVEIS (manteniment_mensual, manteniment_anual, averia)
- DESCARREGAR DADES DELS SERVEIS D'UNA COMUNITAT EN UN EXCEL
- VEURE LOCALITZACIÓ COMUNITAT VIA GOOGLE MAPS

4. PLANIFICACIÓ

4.1 Les fases i el diagrama de Gantt

La planificació que seguida durant el projecte, és la que ha estat esmentada en l'apartat anterior. Específicament, la feina ha estat dividida en tres fases: la fase inicial, la fase de desenvolupament i la fase final.

Tal com bé s'ha comentat en l'apartat anterior, aquestes fases no es van planificar segons el temps d'entrega, sinó que directament es va acordar amb el client, una única data, i aquesta és la final d'entrega de l'aplicació, moment en el qual ja estés totalment acabada i en funcionament. Entrant concretament a la fase de desenvolupament, aquesta va ser dividida en3 blocs (*Perfil, Comunitats i Serveis*), les quals es van presentar una vegada funcionaven sense presentar cap mena d'anomalies, encara que no hi va haver un previ acord del seu temps d'entrega.

Per tal mostrar la planificació esmentada de manera més visual, s'ha dissenyat un **diagrama de Gantt**, en el qual hi ha reflectides totes les tasques i activitats dutes a terme durant la creació de l'aplicació:

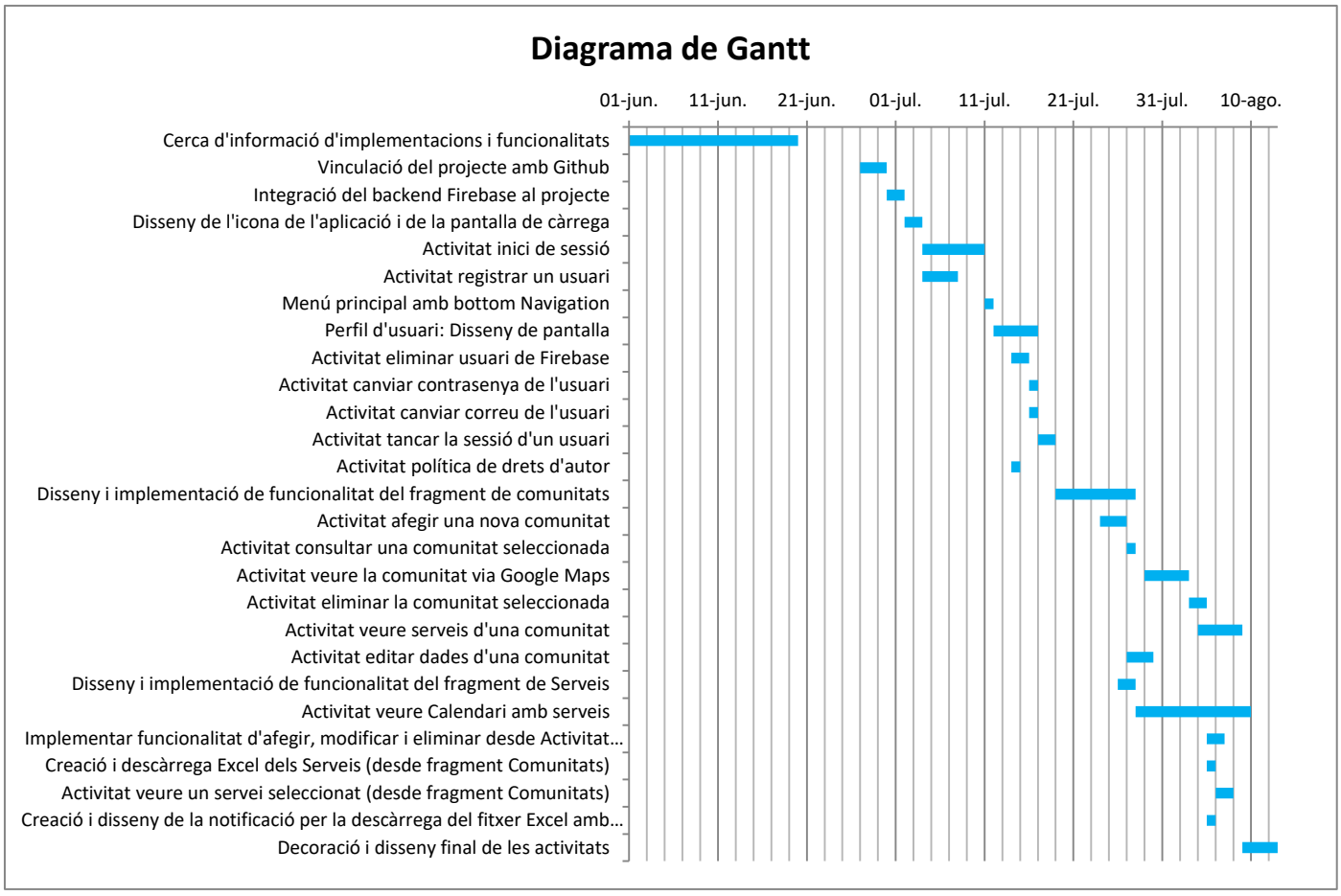


Figura 4.1: Diagrama de Gantt del projecte

4.2 Planificació de la fase inicial

La principal tasca a realitzar durant aquesta fase, va consistir a parlar amb el client i arribar a uns consensos sobre el tipus de dades necessitaria tenir enregistrades a l'aplicació i també sobre les funcionalitats que aquesta li hauria de garantir.

Una vegada aclarides les dades necessàries, es va començar a dissenyar un diagrama de casos sobre l'ús de l'aplicació, però sempre respectant les funcions establertes que hi apareixerien, així com les dades que el client sol·licitava. D'altra banda, es va fer una recerca a través de diversos fòrums i pàgines d'Internet amb la finalitat de trobar algorismes i sistemes que fossin útils i adients amb algunes de les funcions que havia de tenir l'aplicació. Finalment, es va fer un primer disseny de l'aplicació mitjançant una pàgina web anomenada *Uizard*, la qual va permetre començar a donar forma al projecte que s'ha assolit. En el moment de donar per acabada aquesta primera fase, es va fer una reunió amb el tutor per ensenyar-li tot el plantejament més general del projecte per tal de rebre feedback i continuar avançant amb la pròxima fase.

Així doncs, la distribució i planificació d'aquesta fase és la següent:

- **Planificació del projecte:**
  - Reunió amb el client
  - Establiment d'objectius i anàlisi de requeriments de l'aplicació
  - Disseny del diagrama de casos d'ús de l'aplicació (primera versió)
  - Disseny de l'aplicació amb *Uizard*
  - Reunió amb el tutor
  - Temps empleat: 15h

- **Estudis previs:**
  - Llibreries *d'Android Studio*
  - Cerca d'informació relacionada amb l'ús *d'Android Studio i Firebase*
  - Cerca d'informació de futures implementacions a realitzar
  - Temps empleat: 12h

4.3 Planificació de la fase de desenvolupament

La fase de desenvolupament està constituïda per tasques més àrdues. En general, es podria definir aquesta fase com la més complexa, ja que en aquesta fase s'hi crea l'aplicació, tant pel que fa al seu disseny com la practicitat de les funcions lògiques de cada un dels requisits que la constitueixen. Per aquesta fase, he fet les següents tasques:

- **Desenvolupament:**
  - Integració del Github al projecte i del Firebase
  - Dissenys més implementacions
    - Disseny de l'icona de l'aplicació i de la seva pantalla de càrrega
    - Disseny i implementació de la funcionalitat del menú Perfil
    - Disseny i implementació de la funcionalitat del menú Comunitats
    - Disseny i implementació de la funcionalitat del menú Serveis
  - Creació de les diverses activitats de cada menú (iniciar sessió, registrar usuaris, ...)
  - Realització de proves de les diferents funcionalitats que s'han afegit al projecte
  - Diverses reunions amb el client per tal d'aclarir noves sol·licituds i suggeriments
  - Reunió amb el tutor per aclarir dubtes i comentar el progrés del projecte
  - Temps empleat: 110h

4.4 Planificació de la fase final

Aquesta darrera fase del projecte, bàsicament ha constat en aplicar un disseny més actual i elegant a l'aplicació i també s'ha centrat en arreglar els *bugs* i correcció d'errors que encara hi havia en aquesta. A més, com a tasca final del projecte, s'ha realitzat tota la documentació del projecte en general.

- **Desenvolupament:**
  - Canvis en el disseny visual de les pantalles de l'aplicació
  - Correcció de petits errors i *bugs* existents
  - Reunions amb el client per mostrar-li el disseny final de l'aplicació
  - Reunió amb el tutor per ultimar detalls sobre el projecte i de la documentació necessària que s'haurà d'entregar
  - Temps empleat: 15h

- **Documentació:** Memòria del projecte desenvolupat
  - Temps global destinat a l'elaboració de la memòria escrita: 60h

5. MARC DE TREBALL I CONCEPTES PREVIS

En aquest apartat s'introduiran tots els conceptes clau d'aquest projecte per tal d'entendre'l millor, així com s'especificarà quina ha estat l'estructura i les eines que s'han fet servir al llarg d'aquest.

5.1 Intellij IDEA i Android Studio

5.1.1 Intellij IDEA

Intellij IDEA és un entorn de desenvolupament integrat en *JAVA*, el qual permet desenvolupar programari informàtic que estigui escrit en llenguatges c*om JAVA, Kotlin, Groovy,* entre d'altres. Està desplegat per *JetBrains* i està disponible en edició comunitària amb llicència d'Apache2, i també en una edició comercial pròpia.

5.1.2 Android Studio

El concepte *Android Studio* ha estat utilitzat en molta freqüència, però... realment què és i per a què serveix?

Android Studio és un entorn de desenvolupament integrat, sovint conegut com a **IDE** [19] . Aquesta IDE s'utilitza per al desenvolupament d'aplicacions per dispositius que requereixen S.O. Android i està basat en **Intellij IDEA**.

A més de ser una eina molt potent per codificar, també ofereix una gran varietat de funcions que afavoreixen la seva productivitat. Algunes d'aquestes funcions són les següents:

- Un sistema de compilació flexible basat en **Gradle**
- Emulador ràpid i carregat de funcions
- Entorn unificat en el qual desenvolupar aplicacions per tota mena de dispositius *Android*
- Aplicar canvis en algunes parts de codi i altres recursos de l'aplicació sense necessitat de reiniciar-la
- Integració amb **GitHub** i plantilles de codi que ajuden en la compilació
- Varietat de marcs de treball i eines de prova
- Eines per identificar problemes de rendiment, compatibilitat i ús de versions, entre altres
- Compatibilitat amb *C++ i NDK*
- Compatibilitat integrada amb **Google Cloud Platform**, que facilita la integració amb *Google Cloud Messaging i App Engine*

5.2 Estructura del projecte

El projecte elaborat inclou diversos mòduls amb recursos i codis font. Alguns dels mòduls que s'inclouen són els següents:

- Mòduls d'aplicacions per *Android*
- Mòduls de biblioteques
- Mòduls de *Google App Engine*

Android Studio permet veure tots els arxius en la vista del projecte *Android*, de manera que aquests quedin organitzats per **mòduls**. D'aquesta manera, permet l'accés a un arxiu "font" d'un mòdul amb més rapidesa i simplicitat.

A continuació es mostra una vista de l'estructura del projecte que s'ha dissenyat:

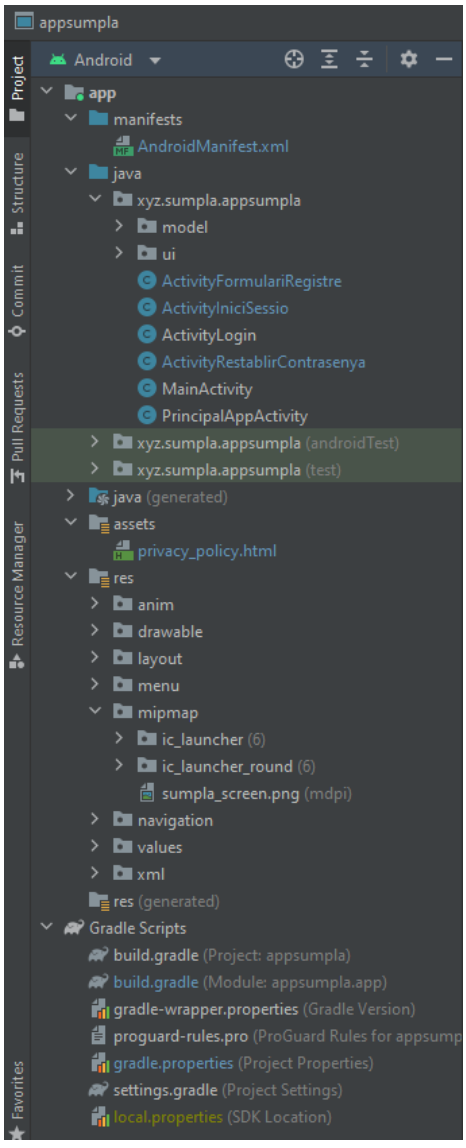


Figura 5.1: Captura de pantalla del Projecte – Vista de l'estructura del projecte dissenyat

5.3 Interfície d'usuari

La *interfície d'Android* consta de diverses àrees lògiques, posteriorment reflectides a partir de la següent imatge:

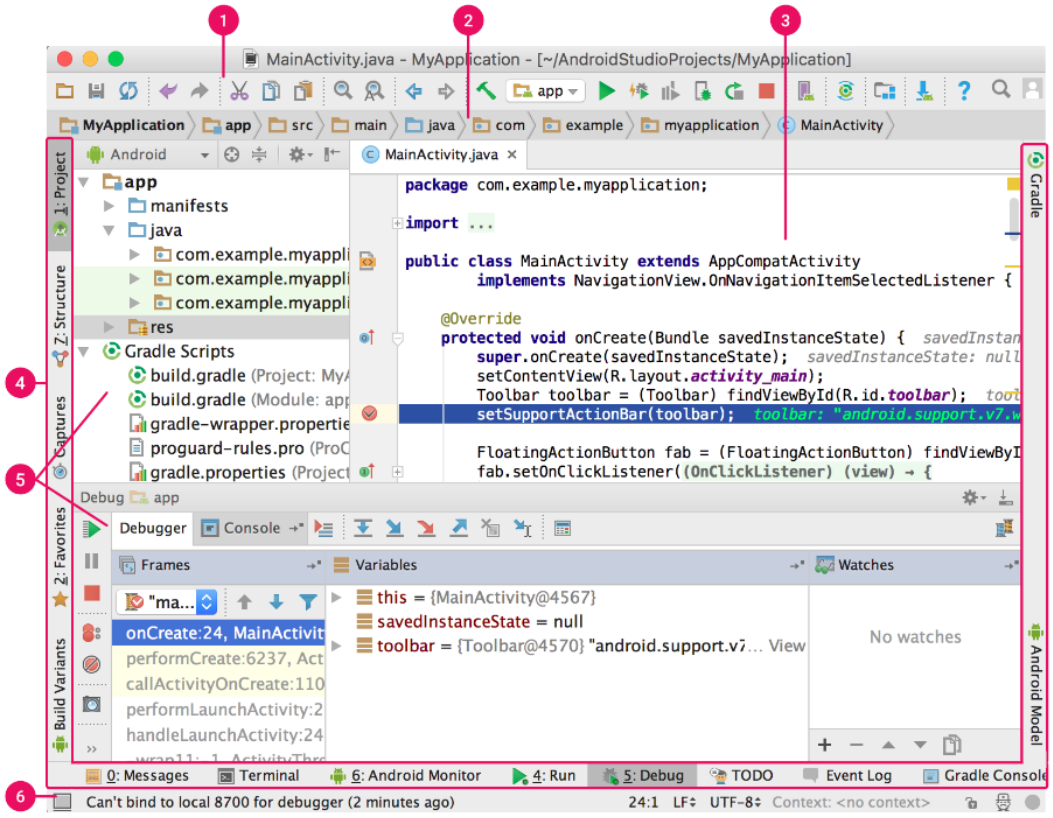


Figura 5.2: Finestra principal d'Android Studio. [DESARROLLADORES DE ANDROID]

1- **Barra d'eines**: Permet una gran varietat d'accions com ara la d'executar l'aplicació
2- **Barra de navegació**: Ajuda en l'exploració del projecte i a obrir arxius per modificar-los. Tanmateix, la barra de navegació dóna una vista més compacta del lloc on està l'arxiu.
3- **Finestra d'edició**: Àrea en la qual es pot crear i modificar codi. En funció de quina activitat es tracta, l'editor canvia. Per exemple si editem un arxiu de disseny (**XML**), l'editor mostra un *Editor de disseny*.
4- **Barra de la finestra d'eines:** Conté els botons que permeten engrandir o contraure les finestres d'eines individuals
5- **Finestres d'eines:** Donen accés a tasques més concretes, com l'administració de projectes, la cerca, el control de versions entre altres.
6- **Barra d'estat:** Mostra l'estat del teu projecte i de l'IDE, a més d'advertències i missatges.

5.4 Conceptes bàsics d'Android Studio

5.4.1 Activitat i Fragment

En alguns moments s'ha posat molt d'èmfasi a la paraula *Activitat* i també a la paraula *Fragment*, però realment què és exactament cada concepte, i per què al final ho he decidit fer d'una manera o altra?

Per començar es definirà què és una *Activitat* i què és un *Fragment*, així com s'especificarà en què es diferencien entre si. Una *Activitat* és una representació d'una sola pantalla que consta d'una interfície d'usuari com si fos una finestra. Dit d'una altra manera, és la creació d'una nova pantalla la qual tindrà una funció que a priori, serà diferent de qualsevol altra [21] .

Pel que fa al Fragment, se'l podria definir com una part de la interfície d'usuari que es pot afegir o eliminar de forma independent a la resta d'elements que apareixen a l'Activitat. O bé, podríem definir el fragment com una vista. Per tant, podríem diferenciar aquests conceptes pensant que una Activitat pot prevaldre per ella mateixa o bé incloure un o més Fragments, mentre que un Fragment necessita ser cridat per una Activitat i així llavors poder-la mostrar [20] .

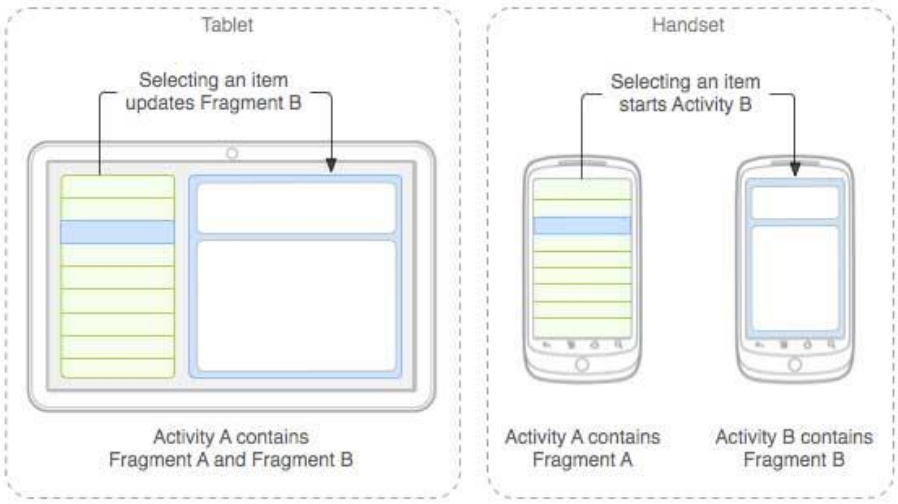


Figura 5.3: Diferents maneres en què una Activitat conté diferents Fragments.
[TUTORIALSPOINT]

D'altra banda, es pot veure que les funcions que es criden en una *Activitat* i en un *Fragment* són similars, però alhora diferents entre si, és a dir, mentre que en una *Activitat* crees tota la interfície que més tard es mostrarà, en un *Fragment* crees la vista que una vegada creada serà mostrada a *l'Activitat*.

5.4.2 Els serveis d'Android

Un altra concepte imprescindible d'aquest projecte són els serveis [23] . Aquests serveis s'utilitzen per resoldre tasques en segon pla dins l'aplicació sempre que no continguin cap interfície de vista per l'usuari. En l'aplicació, aquests serveis s'usen per poder accedir a la base de dades de *Firebase* i obtenir-ne les dades necessàries a cada moment. També s'empren els serveis de *Google Maps* per tal de mostrar la ubicació de la comunitat que s'hagi sol·licitat.

Figura 5.4: Geolocalització amb els Serveis de Google Maps Platform. [GOOGLE DEVELOPERS]

5.4.3 Els Intents

Un altre dels conceptes més importants i utilitzats en aquest projecte és *l'Intent*. Els *Intents* són objectes que permeten enviar missatges. Hi ha diversos tipus *d'Intents*, però els que s'usen en aquest projecte són únicament per fer cridar una *Activitat*. D'aquesta manera, s'aconsegueix navegar per totes les pantalles de l'aplicació, provocant que cada vegada que es va d'una pantalla a una altra, facis un nou *Intent*. L'únic lloc el qual no conté Intents és el menú principal, que ha estat substituït per un *bottom Navigation*.

5.4.4 Bottom Navigation

El *bottom Navigation* és una *Activitat* que utilitza una navegació situada a la part inferior de la pantalla, dividida en tres botons: el *menú de Serveis*, el *menú de Comunitats* i el *menú de Perfil.*

Aquest tipus de navegació és molt típica en les aplicacions que existeixen actualment i també una de les més utilitzades. Aprofitant la seva comoditat i senzilla en la implementació, aquesta activitat s'ha utilitzat únicament en el menú principal de l'aplicació. Aquesta Activitat té tres fragments diferents, els quals s'hi pot accedir a través d'aquesta navegació.

5.4.5 Els avisos

Els avisos, o més comunament anomenats “Toasts”, proporcionen informació sobre una acció a través d’una petita finestra emergent. Habitualment aquesta finestra apareix a la part de sota de la pantalla, tot i que és possible editar-la. Només sol ocupar l'espai necessari per mostrar el missatge que s'hagi de notificar. Aquests avisos són temporals i desapareixen al cap de pocs segons. Dins l'aplicació, aquests avisos s'han utilitzat en bastants activitats, ja que és essencial notificar a l'usuari si l'acció realitzada ha estat realitzada amb èxit o bé si pel que sigui, aquesta ha fallat.

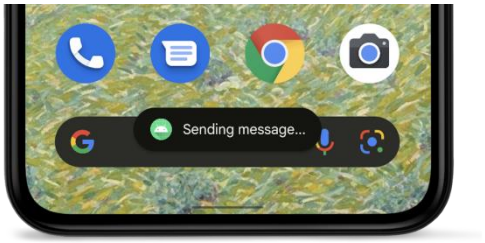


Figura 5.5: Representació d’un avís. [DESARROLLADORES ANDROID]

5.4.6 Els Widgets

Els *widgets* es podrien definir com a petites aplicacions que reben actualitzacions periòdiques. Per exemple, per utilitzar un *widget* relacionat amb el clima al mòbil, no és necessari haver d’accedir a l’aplicació per assabentar-se de quina és la temperatura o clima que fa en un determinat moment. En el cas de *SUMPLAPP* es pot considerar *widget* el calendari personalitzat que disposa, ja que permet saber quins dies hi ha serveis sense haver-ne de seleccionar un en concret quan es vol obtenir aquesta informació.

Els *widgets* que més han estat utilitzats en l’aplicació són:

- **Button**: *Widget* que permet a l’usuari clicar-hi, i així dur a terme una acció o funció en concretes

- **TextView**: Permet mostrar un text a l’usuari. És un text no editable per part seva, motiu pel qual l’usuari només el podrà visualitzar.

- **EditText**: Element que dóna permís a l’usuari perquè hi pugui interactuar i, si s’escau, també editar-lo. Depenent del tipus, podria ser que només permetés l’entrada de números o bé també de caràcters.

Pels widgets *EditText* i *TextView*, existeixen diverses classes:

  - **Numeric**: Només permet l’entrada de text numèric

- **Password**: Permet qualsevol caràcter, però es mostra el text que es va entrant de forma xifrada

- **PhoneNumber**: Permet entrar un text com si es fos el marcatge d'un número de telèfon

- **Text**: Permet entrat qualsevol caràcter

- **FloatingButton**: Element similar al *Button,* però aquest és circular i acostuma a ser l'acció principal de l'aplicació. Aquest botó és molt versàtil i dóna molt de joc en la manera de desenvolupar alguns dels menús que hi ha implementats en l'aplicació. A més a més, és una eina molt recent i utilitzada freqüentment en la gran majoria d'aplicacions.

- **SearchView:** Cercador que permet la cerca de caràcters en determinades parts de l'aplicació. En l'aplicació, aquest cercador s'utilitza dins la llista del *RecyclerView* per poder buscar entre les diferents *Comunitats* que hi apareixen.

  La cerca de comunitats, permet fer-la de quatre maneres diferents:

  - Per nom del carrer

  - Per número de carrer

  - Segons la població

  - Per codi Postal

- **ImageView**: Mostra els recursos d'una imatge, ja sigui un *mapa de bits* o un *Drawable*

5.4.7 Els layouts

Els *layouts* són una de les parts més importants de qualsevol aplicació de mòbil. Un *layout* és la distribució dels diferents elements i formes que apareixen dintre un disseny, o bé també se la podria definir com la representació d'un pla sobre el qual es va dibuixant tota la distribució d'un espai.

5.4.8 Grups de vistes i View Groups

Un *ViewGroup* és un objecte invisible que s'utilitza per incloure objectes de tipus *View* i *ViewGroup* amb l'objectiu d'organitzar i controlar el *layout* d'una pantalla, mentre que una *View* és únicament una d'aquestes vistes. Aquests *ViewGroups* s'usen per a la creació d'una jerarquia d'objectes *View*, de manera que permeti tenir *layouts* més complexes. Seria possible que un disseny seguís un estil com el següent:

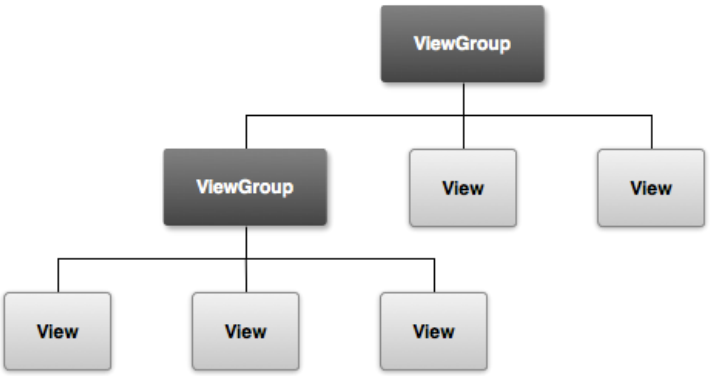


Figura 6: Model d'encapsulament d'un ViewGroup. [CODE ENVATO TUTS]

Cada subclasse de *ViewGroup* pot ser-hi arran d'un propòsit totalment diferent d'un altre. A continuació, hi vegem algunes de les subclasses utilitzades durant el projecte:

- **LinearLayout**: Mostra els elements amb un ordre apilat, sigui horitzontal o vertical.
- **RelativeLayout**: Permet visualitzar elements a la pantalla que estan relacionats entre si, proporcionant més flexibilitat i llibertat sobre la manera de com aparegui el *layout*.
- **ScrollView:** Permet desplaçar la pantalla en horitzontal o vertical, i així poder veure tota *l'Activitat o Fragment* que hi hagi. En el cas del meu projecte, s'utilitza *ScrollView* vertical.
- **RecyclerView**: Facilita la mostra de grans conjunts de dades eficientment. Utilitza un *adaptador*, l qual s'encarrega de buscar a la base de dades les que s'hagin de mostrar, seguidament en crea una llista i finalment l'acaba mostrant.

5.4.9 Les animacions

Les animacions són efectes visuals que s'afegeixen a certs elements, per així informar a l'usuari sobre què està succeint a l'aplicació. Són molt útils per quan es vol fer canviar d'estat la UI. Per exemple quan es carrega un contingut nou o bé quan hi ha noves accions disponibles.

Tot i això, però, cal destacar que les animacions no abunden en excés dins del projecte desenvolupat, però que sí que se'n poden identificar algunes en els botons flotants que formen part de l'aplicació, i això dóna joc a nous menús amb una nova interfície, i noves accions a realitzar.

5.4.10 Gràfics vectorials

Android Studio inclou una eina anomenada *Vector Asset Studio*, que permet afegir icones o importar arxius de gràfics vectorials escalables (també coneguts com a *SVG*) al projecte, com a recursos d'elements de disseny vectorial [14] . L'ús d'aquests elements de disseny vectorial en lloc de fer servir *mapes de bits*, obliga a reduir la mida de la pròpia *APK* i això permet canviar la mida de l'arxiu sense que la imatge perdi qualitat.

Es poden fer servir *Vector Assets* , els quals ja formen part del mateix *Android Studio* i també importar alguna imatge, de la qual seguidament se'n faria un gràfic vectorial.

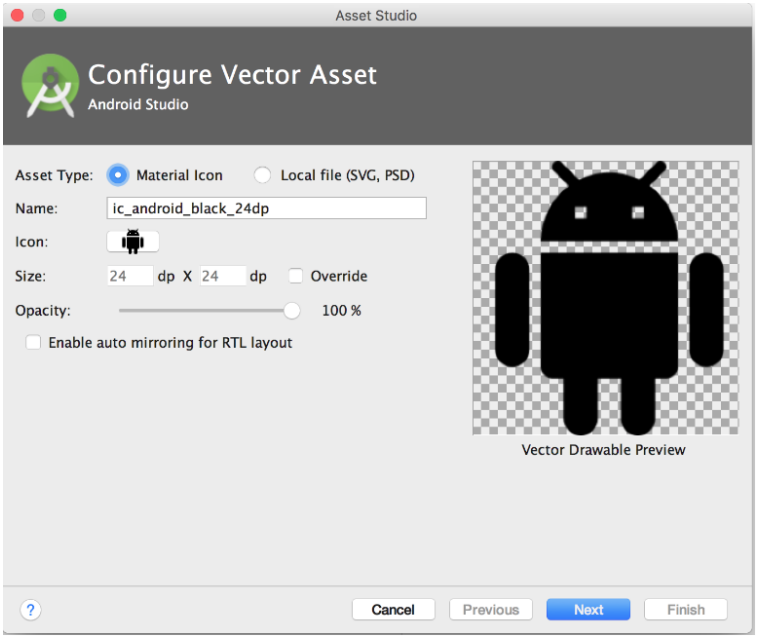


Figura 5.6: Configuració d'un Vector Asset amb un icona predeterminat d'Android Studio.
[DESARROLLADORES DE ANDROID 2]

Tal com s'ha esmentat anteriorment, és possible la importació d'un arxiu local, i a continuació es pot comprovar mitjançant la següent imatge:

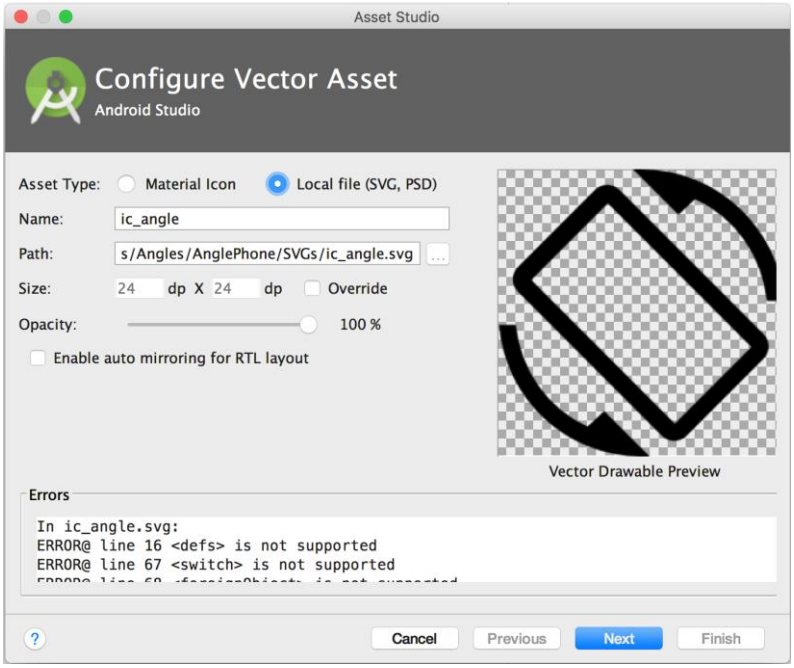


Figura 5.7: Configuració d'un Vector Asset amb un arxiu local importat. [DESARROLLADORES DE ANDROID 3]

6. REQUISITS DEL SISTEMA

6.1 Requisits Funcionals

Per saber quins són els requisits funcionals de l'aplicació, abans de res és necessari identificar els actors que hi participen. Com a actors participants, doncs, n'hi haurà dos de principals: **Usuari registrat i usuari no registrat.**

Un **usuari no registrat** només haurà de complir un sol requisit funcional:

- Podrà crear-se un compte d'usuari a través del qual fer servir l'aplicació amb totes les seves funcionalitats. Una vegada l'usuari estigui creat caldrà que verifiqui el compte, altrament continuaria essent un usuari no registrat.

Un **usuari registrat** tindrà un ampli ventall de funcionalitats per realitzar:

- Es podrà canviar el seu correu electrònic i la seva contrasenya
- Tindrà la possibilitat de veure la política de privacitat sobre l'aplicació
- Li serà possible tancar la sessió de l'usuari actual
- Podrà eliminar el compte permanentment (tot i que per fer-ho s'haurà de re-autenticar).
- L'usuari tindrà l'opció d'afegir una nova comunitat amb les seves respectives dades
- Podrà modificar les dades d'una comunitat o bé directament eliminar-la.
- Serà possible que pugui veure la localització d'una comunitat via *Google Maps*
- L'usuari podrà veure els serveis d'una comunitat
- L'usuari podrà veure les dades d'un servei a partir d'una comunitat en concret
- Se li oferirà la possibilitat d'eliminar un servei a partir d'una comunitat en concret
- L'usuari podrà descarregar-se en format *Excel* tots els serveis d'una comunitat
- L'usuari podrà ordenar els serveis d'una comunitat per "més recent" o per "més antics" segons la data en què s'hagi realitzat el servei
- Tindrà la possibilitat d'accedir al calendari d'una comunitat, així com d'afegir-hi un servei en una data en concret
- Podrà accedir al calendari d'una comunitat i modificar les dades d'un servei d'una data en concret
- L'usuari tindrà accés al calendari d'una comunitat i eliminar un servei d'una data en concret

6.2 Requisits no funcionals

Com que per dur a terme aquesta aplicació s'ha utilitzat *Android 11.0*, aquesta ofereix la possibilitat que qualsevol mòbil amb sistema operatiu *Android* i amb accés a internet la pugui fer servir.

7. ESTUDIS I DECISIONS

7.1 IDE i llenguatge escollit

La realització d'aquest projecte, ha estat basada en l'assignatura "Projecte de Desenvolupament del Software" de la carrera d'Enginyeria Informàtica, la qual es basa a fer servir la *IDE Android Studio* i, en conseqüència, utilitzar-la com a *frontend Android* [16] . El dubte més gran que va aparèixer mentre es pensaven i s'estudiaven les diferents possibilitats que sorgien durant l'estudi del marc de treball, era en relació amb decidir quin *Backend* fer servir. Després d'estudiar-ne varis i valorar els avantatges que comportaria utilitzar-ne un o altre, finalment es va optar per fer servir *Firebase*, ja que, entre molts altres avantatges, aquest és un *backend* molt modern i utilitzat amb freqüència avui en dia que, a més de constar d'una consola on poder fer les diferents integracions i consultes pertinents, et facilita molt la integració d'aquest *backend* amb el projecte *frontend d'Android* en ser un producte de *Google* [2] .

Figura 7.1: Característiques i guies que ofereix Firebase a la seva documentació. [FIREBASE]

Pel que fa al llenguatge, es va dubtar sobre si fer servir *Kotlin* o *Java*, però al final, després de veure diverses implementacions en diverses pàgines web, es va comprovar que el llenguatge més utilitzat era el de *Java*, i que, per tant, aquesta era la millor opció per si més endavant apareixien qüestions en alguna implementació que s'anés fent.

7.2 API de Google

Per tal de poder fer servir el projecte *d'Android* juntament amb les implementacions que oferia *Firebase,* va ser necessari unir *l'API de Google* amb el projecte. Aquesta integració es va dur a terme a partir d'una *Credencial* i d'un codi únic anomenat *SHA-1,* els quals disposa el mateix projecte *d'Android Studio.*

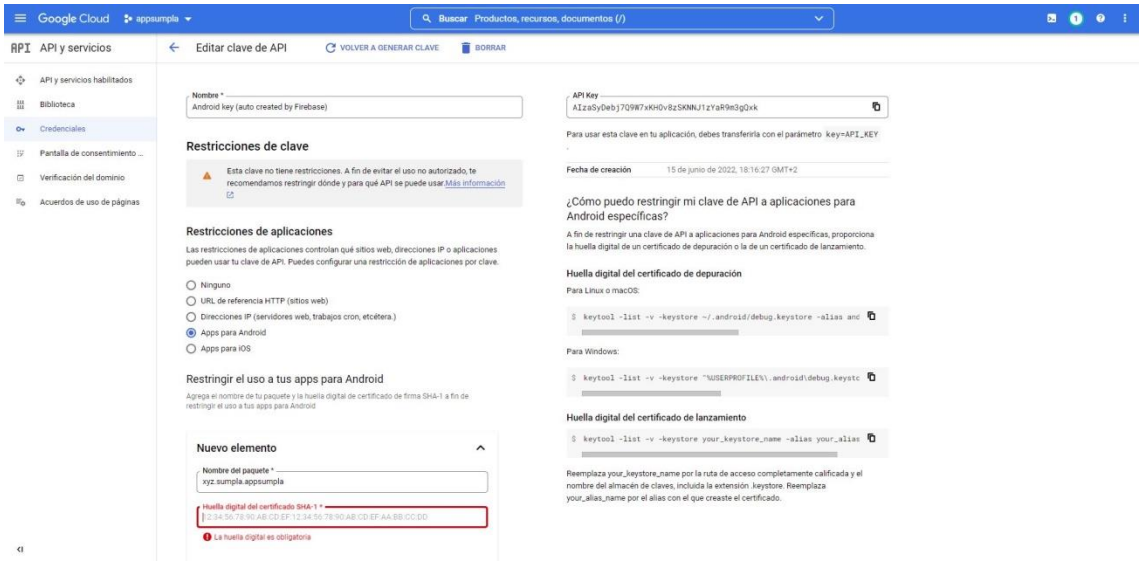


Figura 7.2: Captura extreta de google cloud propi – S'afegeix el codi SHA-1 per crear una credencial pel projecte d'Android

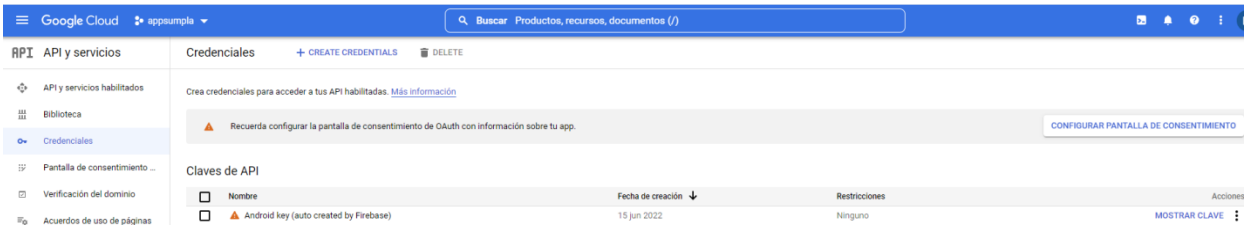


Figura 7.3: Captura extreta de google cloud propi - La clau d'API creada i ja integrada en el projecte anomenat appsumpla

7.3 Llibreries i dependències

El programari que s'ha utilitzat al llarg d'aquest projecte és *Android Studio i Firebase* juntament amb la utilització d'una *API* de *Google,* per poder fer la integració del *Google Maps i* el mateix *Geocoding.* Tanmateix, s'hi ha afegit dues noves llibreries: una d'elles es va afegir amb la finalitat d'obtenir el component anomenat *CompactCalendarView,* mentre l'altra llibreria permet crear fitxers en format *Excel* des de la mateixa aplicació [7, 9, 17].

La primera llibreria o dependència, consisteix en un calendari personalitzat que permet afegir-hi esdeveniments i fer-hi modificacions personals que, el mateix calendari *d'Android Studio* no permet realitzar.

La segona llibreria (tal com s'ha descrit anteriorment), permet la creació d'un fitxer *Excel* dins la pròpia aplicació [13] .

Una altra dependència externa consisteix en la utilització d'un *GIF* per a simular un *Spinner* de càrrega. La resta de dependències són pròpies *d'Android Studio*, així com ho és el mateix *Navigation fragment* o el fet d'utilitzar el *ConstraintLayout*.

A continuació s'esmenten les dependències que s'han afegit al projecte:

- *Component GIF - implementation 'pl.droidsonroids.gif:android-gif-drawable:1.2.19'*
- CompactCalendarView – *implementation 'com.github.sundeepk:compact-calendar-view:3.0.0'*
- *Firebase Authentication - implementation 'com.google.firebase:firebase-auth:19.2.0'*
- *Firebase Realtime Database - implementation 'com.google.firebase:firebase-database:20.0.5'*
- *Google Play Services Maps - implementation 'com.google.android.gms:play-services-maps:17.0.1'*
- *Apache poi (crear fitxer Excel) - implementation 'org.apache.poi:poi:3.17'*

A part d'aquestes dependències, també s'han afegit *propietats* que permeten fer ús de components externs *d'Android Studio* com seria el cas del *CompactCalendarView*. Per aquest cas en concret, es van haver d'afegir les següents propietats [11] :

- *Android.nonTransitiveRClass=true*
- *Android.enableJetifier=true*

7.4 Versions de programa, emulador utilitzat i servidor Firebase

Pel que fa al programari, tal com ja s'ha anat comentant en l'apartat anterior, s'ha utilitzat *Android Studio i Firebase*. Tanmateix, pel *frontend Android Studio*, s'ha fet servir la versió 2021.2.1 la qual ens permetria, si així ho desitgéssim, usar **APIS** *d'Android 13*, tot i que pel cas d'aquest projecte no seria necessari. Tot i això, l'ús d'aquesta versió ve lligada al fet que fa servir *Android Gradle*, que és el sistema de compilació principal. A més, el complement que té *Android* per *Gradle* incorpora diverses funcions específiques per compilar aplicacions ja pensades per aquest sistema operatiu.

Aquesta versió fa servir una mínima *SDK* de 21, la qual té una compilació *SDK* de 32 i una versió *SDK* de destí de *32*. Per poder-les diferenciar entre sí, es pot definir la *minSDK* com la primera versió en què s'executa l'aplicació, mentre que la *targetSDK* (o versió SDK de destí) és la versió destinada a l'ús de l'aplicació. Com bé ja s'ha comentat, si es fes servir *targetSDK* de 33 seria necessari fer servir una *API d'Android 13*, però, com que únicament s'usa fins a 32, es pot fer servir una *API d'Android 11*, sense cap mena de problema. Tanmateix, la versió de compilació *SDK* marca quina versió s'hauria de fer servir en la *IDE*, per tant, en aquest projecte s'hauria d'utilitzar la 32 [22] .

Pel que fa a l'emulador, s'ha anat treballant sempre amb el *Pixel 2 API 30*, una de les més utilitzades en *Android Studio* en ser una versió més pròxima als estàndards actuals de dispositius mòbils existents avui dia, però també perquè és un estil de mòbil molt semblant al que utilitza el client en qüestió, per tant, es considera que aquest és el millor emulador per fer servir, i realitzar les proves pertinents a través d'ell. El mòbil del client consta d'un *Nivell API 30* i d'una *versió Android 11.0*, la qual és la més recent i la que proporciona més privacitat i seguretat queles versions anteriors.

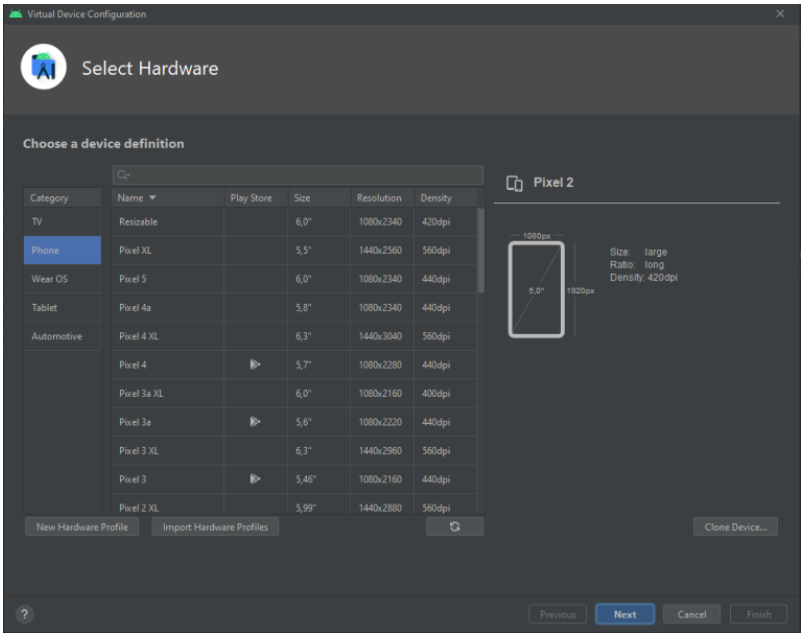


Figura 7.4: Captura extreta del propi projecte Android – Creació de l'Emulador amb dispositiu PIXEL 2

Tanmateix, aquest dispositiu s'utilitza amb la imatge de sistema *R Studio* al ser compatible amb els serveis de *Google Play*.

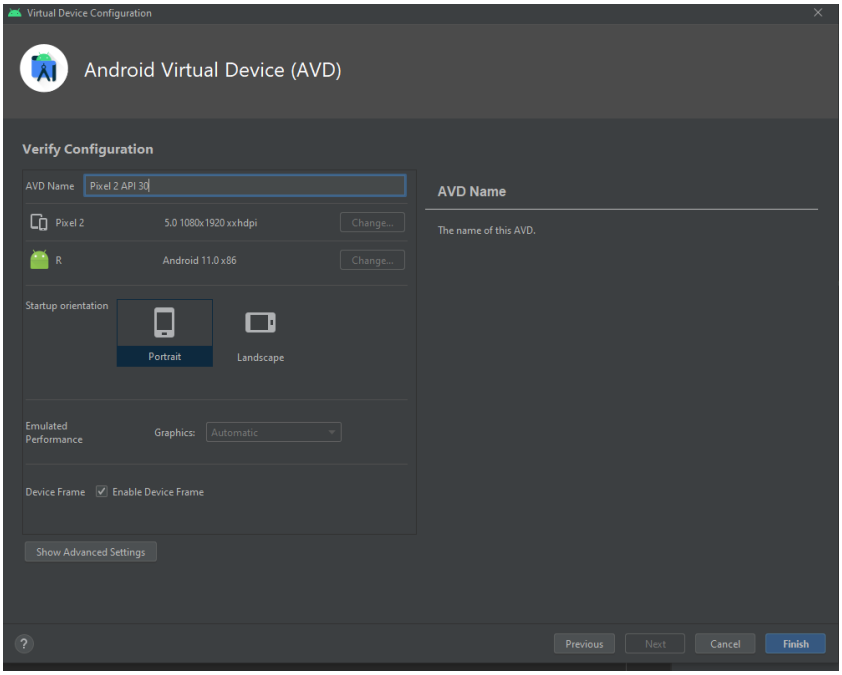


Figura 7.5: Captura extreta del propi projecte Android – Sistema R Studio escollit

Respecte al *Firebase*, una vegada es crea el projecte i es té interès per començar a integrar-hi algunes característiques com *Firebase Authentication* o *Realtime Database* entre altres, cal escollir a si vol ubicar la seva base de dades. Per l'aplicació creada en aquest projecte, el país més proper on poder ubicar aquesta base de dades el qual a priori sembla més factible i que provoca menys interferències era a **Bèlgica**, concretament en la regió *europe-west1*.

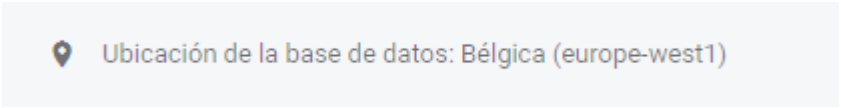
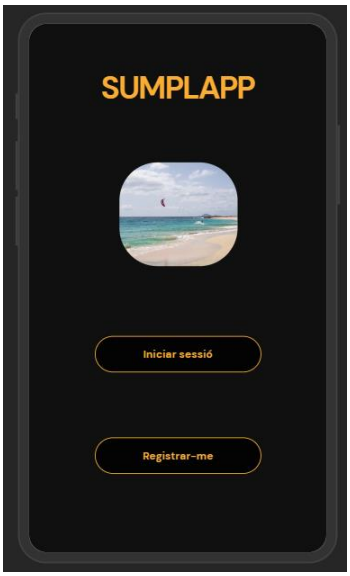


Figura 7.6: Captura extreta del Firebase propi – Ubicació de la base de dades Realtime Database

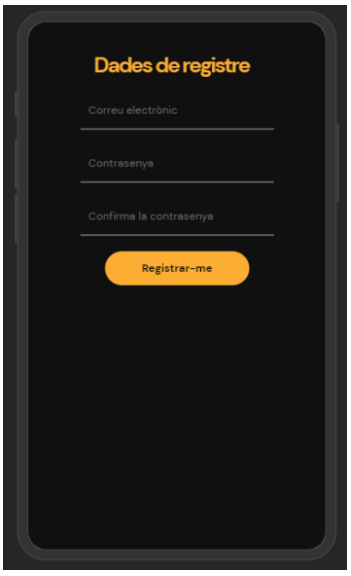
8. ANÀLISI I DISSENY DEL SISTEMA

8.1 Disseny d'interfície de l'aplicació amb UIZARD

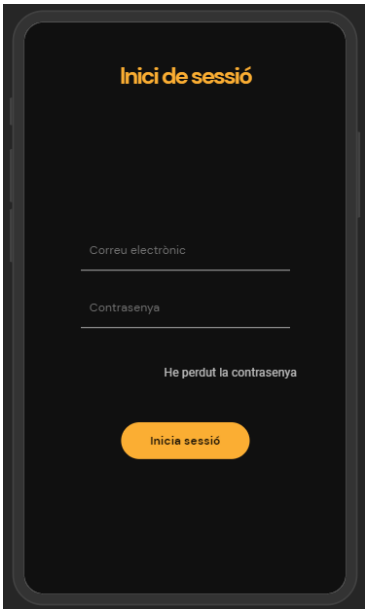
El primer disseny de l'aplicació va ser creat amb una eina anomenada *Uizard* [1] . Aquesta permet la creació i el disseny d'una interfície d'una aplicació via web. Cal remarcar que el que es mostra a continuació, era un primer esbós generat amb les eines que proporcionava *Uizard*, motiu pel qual hi havia limitacions en alguns d'aquests menús. No obstant això, per poder plantejar una primera visió de com s'hauria de veure l'aplicació i les seves respectives funcionalitats, aquesta va ser una eina fonamental per agafar una primera idea i llavors anar-la perfilant més detalladament. A continuació, s'hi poden veure les diferents pantalles que es van plantejar amb *Uizard*:



8.1 Uizard - Pantalla principal



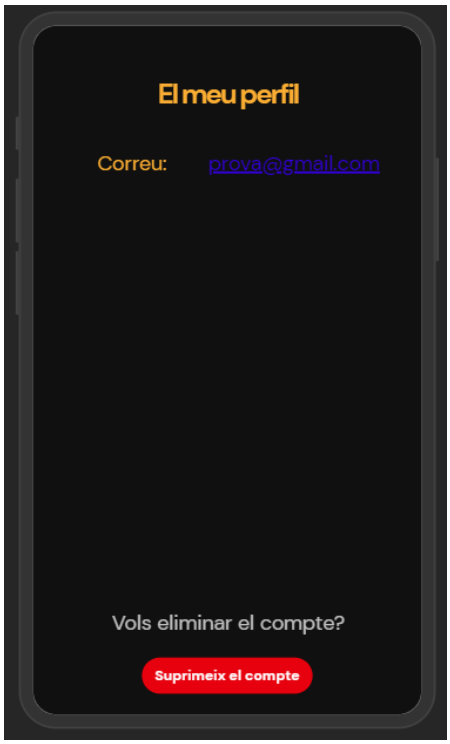
8.2 Uizard - *Registrar nou usuari*



8.3 Uizard - Inici de sessió

8.4 Uizard - Pantalla principal amb els menús

8.5 Uizard - Menú serveis amb el calendari

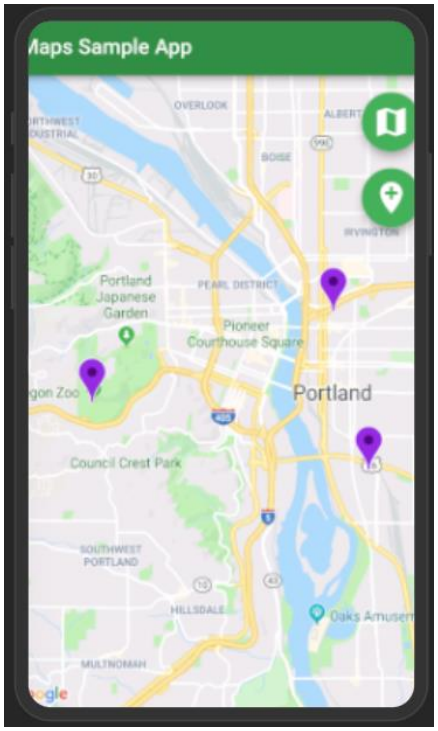


8.6 Uizard - Menú del perfil de l'usuari

8.7 Uizard - Menú Comunitats

8.8 Uizard - Menú inserir una nova comunitat

8.9 Uizard - Menú gestionar una comunitat



8.10 Uizard - Menú veure localització d'una comunitat via Google Maps

8.2 Disseny del sistema

Tot i que en primera instància s'havia fet un diagrama de casos d'ús, més endavant s'ha hagut de modificar per poder-hi afegir alguns requisits que demanava el client. Per ser més concrets, el client demanava una manera de poder accedir als serveis d'una *Comunitat* dins del mateix *menú de Comunitats*, cosa que al principi només s'havia plantejat que es pogués fer des del *menú de Serveis*. Aquesta nova funcionalitat permet, doncs, que l'usuari pugui visualitzar tots els serveis que té una comunitat en concret i, que posteriorment, pugui descarregar tots aquests serveis en format Excel [12] . A més d'això, també s'ha afegit una opció per poder modificar i eliminar un servei concret des d'aquest mateix menú, cosa que abans només estava planejat fer-ho des de *l'apartat de Serveis*.

Finalment, però, s'ha fet un canvi de disseny en el diagrama de casos d'ús per a poder realitzar aquesta nova funcionalitat.

A continuació, es mostra el diagrama de casos d'ús final que s'ha implementat:

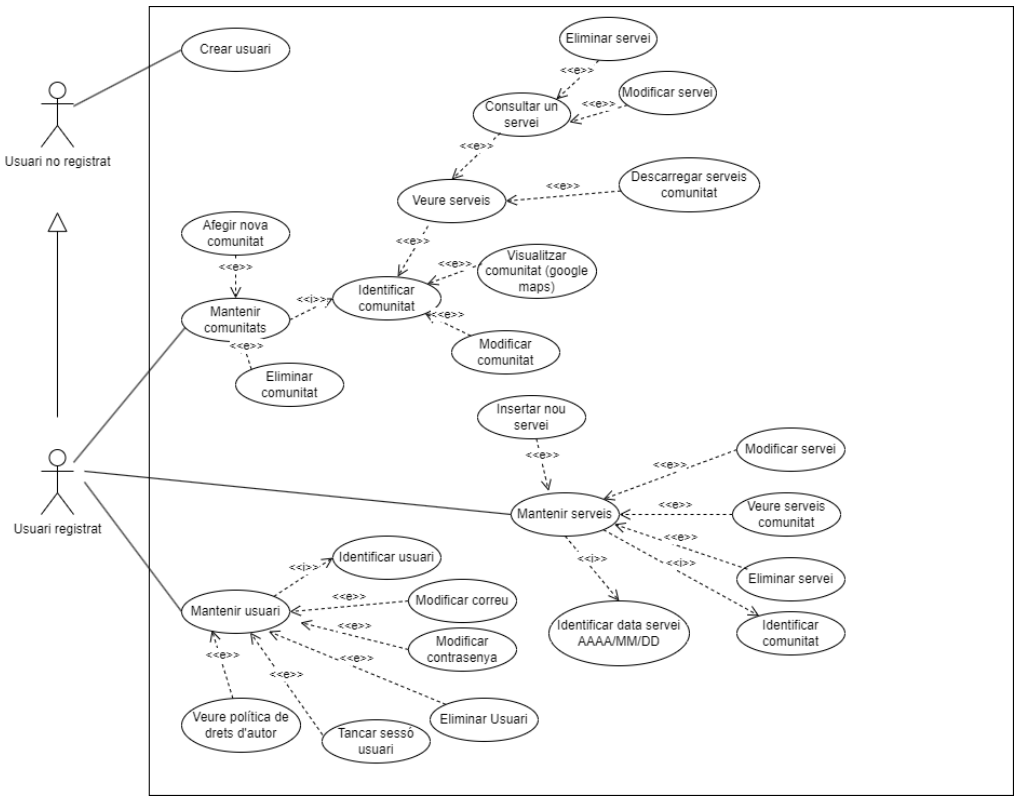


Figura 8.11: Diagrama de casos d'ús final

8.3 Model de dades requerides

Pel que fa a les seves dades, el projecte consta d'una base de dades de *Firebase*, la qual les divideix en dos llocs diferents però units entre si. Per una banda, s'hi troba el **Firebase Authentication** en el qual relacionen totes les dades dels usuaris que s'han registrat a l'aplicació. Per aquest motiu, es donarà permís a tots els usuaris registrats perquè puguin fer servir totes les funcionalitats que disposa l'aplicació, sempre que aquest usuari hagi verificat el compte tal com es demana en el registre de l'aplicació. A continuació la consola del *Firebase Authentication:*

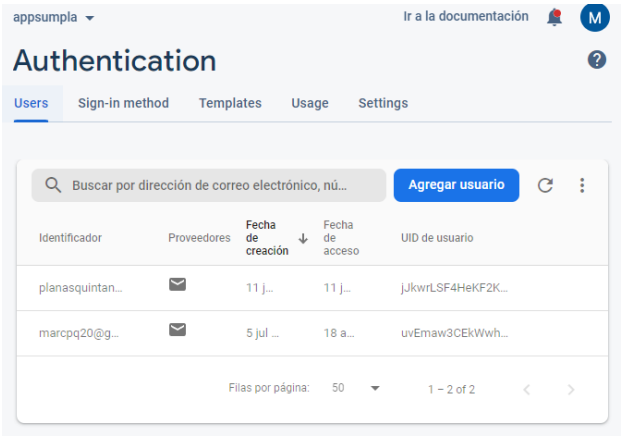


Figura 8.12: Captura de la consola de Firebase Authentication - usuaris que utilitzen la aplicació

Tal com s'ha pogut observar en la imatge anterior, quan un usuari s'ha registrat, és possible obtenir certa informació sobre el compte, com ara:

- **L'identificador:** En aquest cas el correu electrònic

- **El proveïdor**: mètode que ha utilitzat l'usuari per registrar-se i iniciar sessió. Depenent de l'aplicació, podria ser que s'hi trobi altres mètodes com *Yahoo*, *Gmail* o *número de telèfon*, entre molts altres. En el cas d'aquest projecte, només es troba disponible via correu electrònic

- **Data de creació:** Informa quan va ser creada el compte

- **Data d'accés:** Dóna informació sobre quina va ser l'última vegada que l'usuari es va connectar

- **UID de l'usuari:** Es podria definir com un *token* personal. L'usuari, indirectament utilitza aquest *token* per iniciar sessió. Es tracta d'una clau que es crea una vegada l'usuari es registra a l'aplicació

Per altra banda, hi ha el *Firebase Realtime Database* que és a on s'agrupen totes les dades que van introduint i modificant al seu compte d'usuaris, com les *Comunitats* o els *Serveis* [5] .

Per fer-ho més amè, s'ha dissenyat un *Diagrama de classes* per tal de veure amb més claredat la representació que més endavant es podrà veure des de la mateixa consola de *Firebase*:

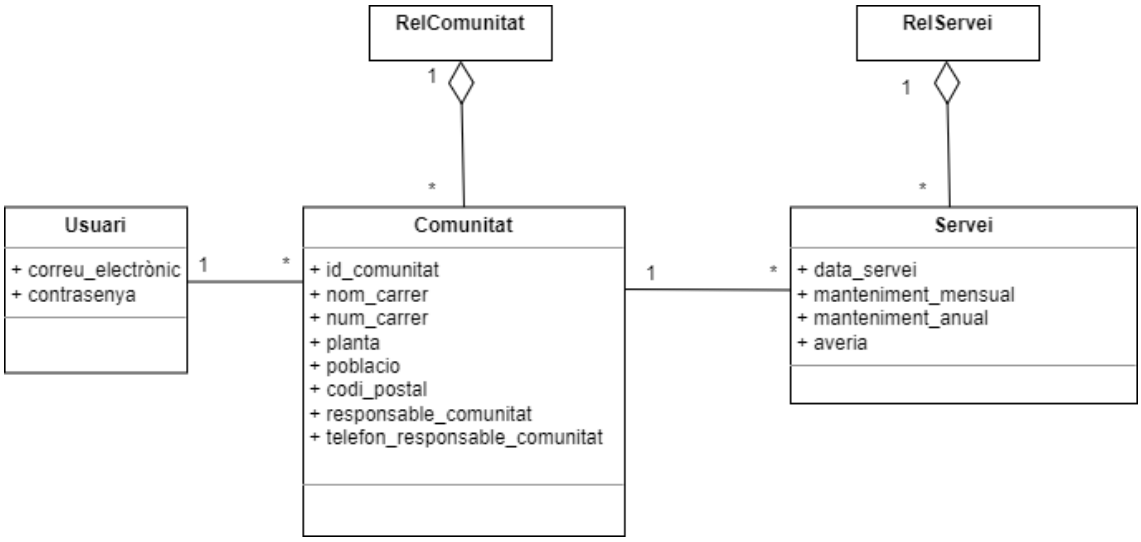


Figura 8.13: Diagrama de classes de l'aplicació

Totes aquestes dades es poden veure des de la mateixa consola de *Firebase*, tal i com es mostra a partir de la següent imatge:

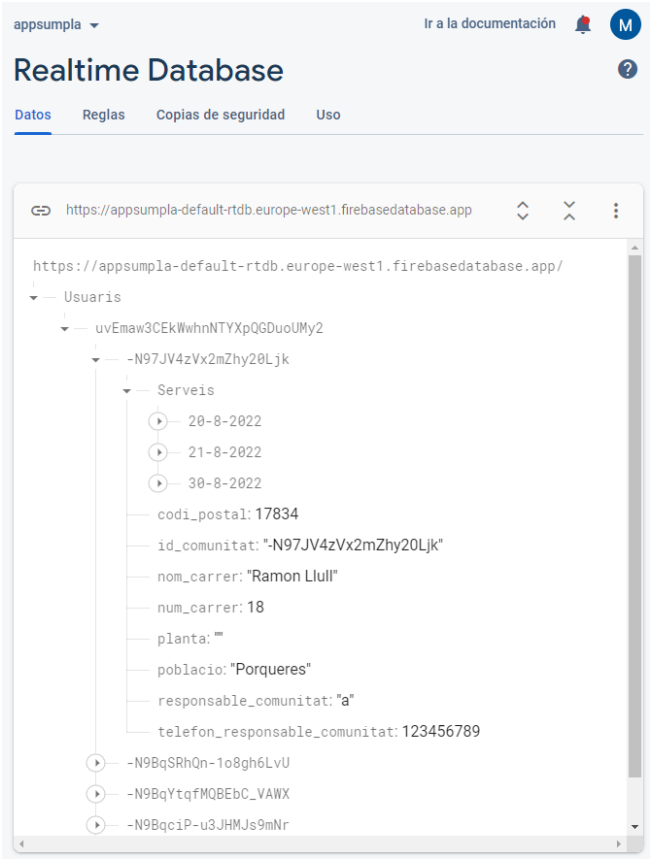


Figura 8.14: Captura extreta Firebase Realtime Database del projecte– Estructura amb algunes dades

Per tal d'entendre com està formada la base de dades, a continuació es farà una explicació dels camps semàntics que conté:

- **Usuaris:** Conté una llista de tots els usuaris existents a *Firebase Authentication* que hagin afegit alguna dada a l'aplicació, com podria ser alguna *comunitat* o algun *servei*. Els usuaris que s'hi afegeixin, ho faran amb la *clau UID* generada pel *Firebase Authentication.*

- **Clau UID de l'usuari:** Disposa de totes les dades d'aquell usuari que s'ha afegit: *Comunitats i Serveis.* Com a identificador únic, s'utilitza el *UID de l'usuari* per tal d'evitar possibles problemes entre claus iguals

- **Clau ID de Comunitat:** Hi té entrades totes les dades que té la comunitat: *Id_comunitat, codi_postal, nom_carrer, num_carrer, planta, població, responsable_comunitat, telèfon_responsable_comunitat i llista de Serveis.* Com a identificador únic, s'utilitza el mateix *ID* de la comunitat, així s'evita qualsevol problema de clau repetida.

Tanmateix, hi ha camps que poden estar buits, tal com es mostra en el cas de l'exemple següent en el qual es pot veure la variable “planta” sense cap caràcter.

Figura 8.15: Captura extreta del Firebase Realtime Database del projecte - Dades d’una comunitat

- **Serveis:** Llista amb tots els Serveis de la Comunitat

Figura 8.16: Firebase Realtime Database – Llista amb tots els Serveis d’una comunitat

- **Data d’un Servei:** Hi contindrà tota la informació del *Servei* que s’ha dut a terme en aquella data. Les dades que apareixeran són: *averia, data_servei, manteniment_anual i manteniment_mensual*. Està dissenyat per què hi hagi camps que puguin quedar buits, com seria el cas d’averia de la següent imatge:

Figura 8.17: Captura extreta del Firebase Realtime Database del projecte - Dades d’un Servei

9. IMPLEMENTACIÓ I PROVES

En aquest apartat es detallaran les implementacions que s'han anat realitzant juntament amb les proves de la seva correcte funcionalitat.

9.1 Implementació dels mètodes pels usuaris

9.1.1 Registre d'usuari i verificació de el compte

Per poder dur a terme el registre d'un nou usuari, primer es va haver d'integrar el *Firebase Authentication*, que serà on tindran cabuda tots els usuaris que faran servir l'aplicació. La manera d'integrar-li-ho és més pràctica del que realment sembla, gràcies al fet que AndroidStudio ja té una opció dintre de l'apartat **Tools** anomenada **Firebase**. Només cal anar en aquest apartat, i en ell s'obre un desplegable amb tot el que conté el *Firebase*: des de *l'Authentication* que necessitava fins a una integració pensada per tenir anuncis. Qualsevol de les opcions que ofereix *Firebase* es troba allà. Així doncs, només va ser qüestió d'anar seguint els passos que demanava, i ja ho tenies tot integrat [3] :

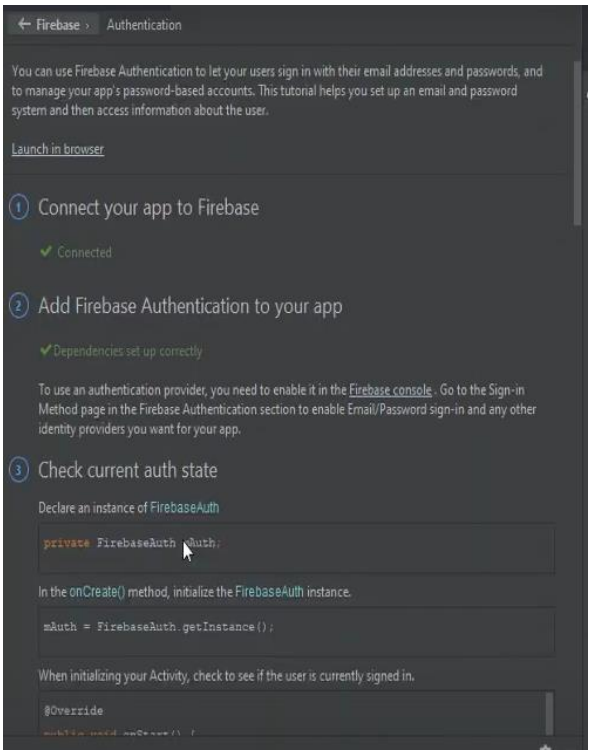


Figura 9.1: Captura de pantalla del projecte – Firebase Authentication connectat

El que té de positiu utilitzar aquesta plataforma és que a més de la integració, també et dóna alguns algorismes per utilitzar. Entre d'altres, hi trobem l'algorisme que permet el registre d'un nou usuari a *Firebase Authentication.*

Finalment, l'algorisme emprat en aquesta aplicació va ser el següent:

```
mAuth.createUserWithEmailAndPassword(email, contrasenya)
        .addOnCompleteListener( activity: ActivityFormulariRegistre.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    FirebaseUser currentUser = mAuth.getCurrentUser();
                    currentUser.sendEmailVerification();
                    Toast.makeText( context: ActivityFormulariRegistre.this,  text: "Usuari creat amb èxit. Accedeix al correu i verifica la conta per poder iniciar sessió",
                            Toast.LENGTH_LONG).show();
                    Intent i = new Intent( packageContext: ActivityFormulariRegistre.this, ActivityLogin.class);
                    startActivity(i);
                    ActivityFormulariRegistre.this.finish();

                } else {

                    Log.w( tag: "Error", task.getException());
                    if(contrasenya.length() < 6) {
                        Toast.makeText( context: ActivityFormulariRegistre.this,  text: "Error en el registre. La contrasenya ha de ser de 6 caràcters o més",
                                Toast.LENGTH_LONG).show();
                    }
                    else {
                        Toast.makeText( context: ActivityFormulariRegistre.this,  text: "Error en el registre.Aquest correu ja està registrat en un usuari",
                                Toast.LENGTH_LONG).show();
                    }
                }
            }
        });
```

Figura 9.2: Captura de pantalla del projecte – Algorisme registrar usuari

Una vegada un nou usuari s'acaba de crear una conta, se l'informa que se li ha enviat un correu electrònic per verificar el registre, i de què no podrà iniciar sessió sense abans haver-se verificat mitjançant un missatge temporal.

A partir de les imatges següents, es pot veure la implementació de l'algorisme anterior des de la mateixa aplicació una vegada realitzat el registre d'un nou usuari i el correu que ha rebut, pera poder procedir amb la verificació i l'avís que rebrà quan, si pel que sigui s'intenta iniciar sessió sense haver verificat el compte. Així doncs, continuació es veurà el diagrama d'activitat que cal seguir per *registrar un nou usuari* i també per *iniciar sessió*, a més del correu que es rebrà per poder verificar el compte:

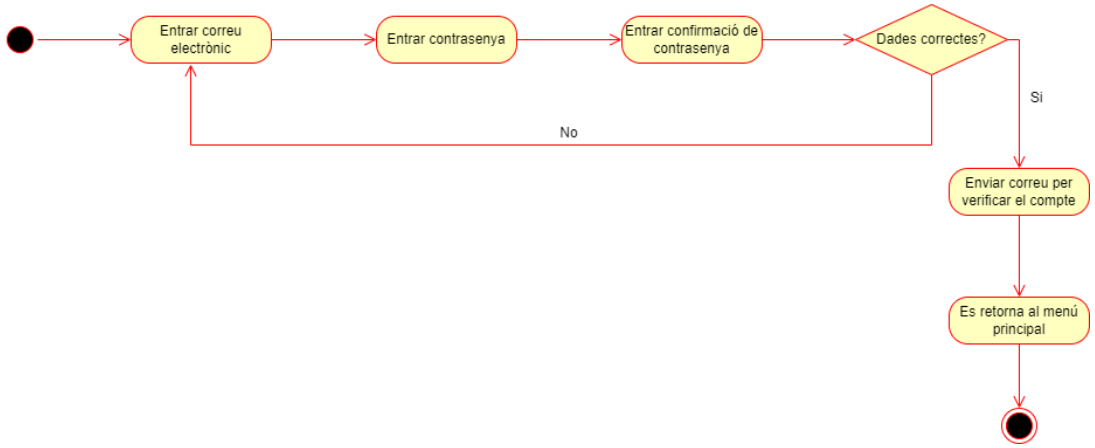


Figura 9.3: Diagrama d'activitats registre de nou usuari

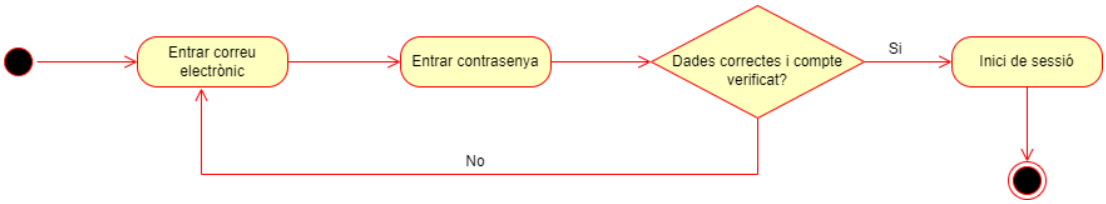


Figura 9.4: Diagrama d'activitats inici de sessió

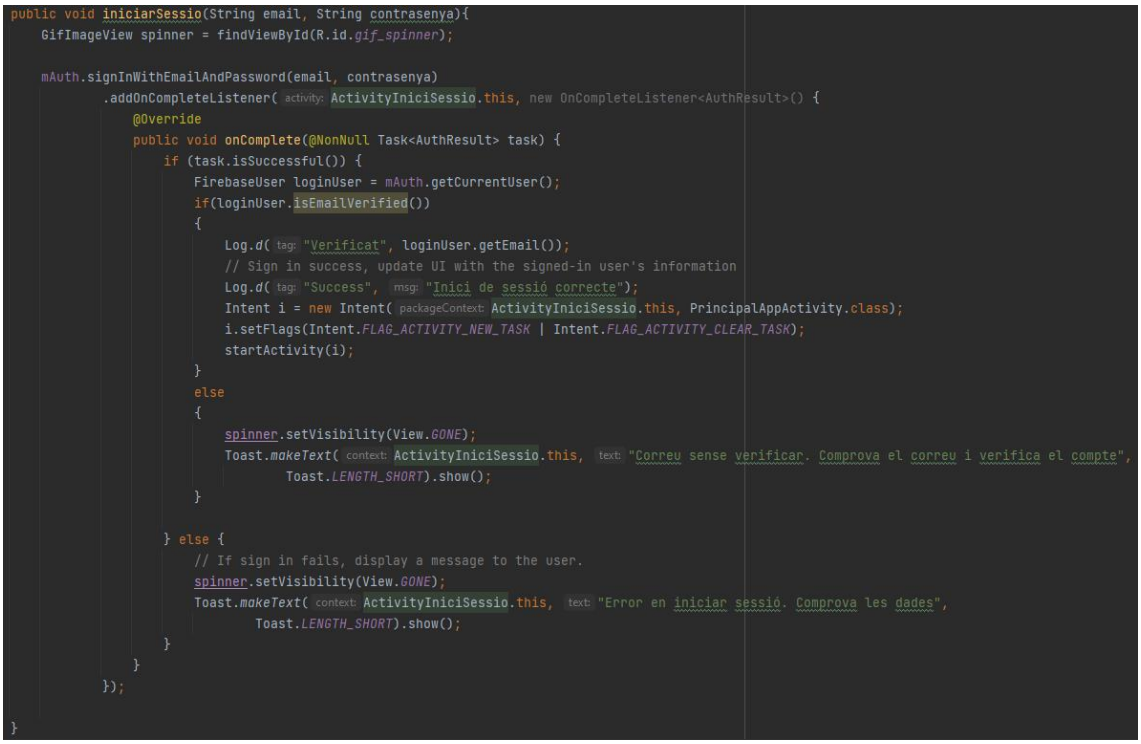
Figura 9.5: Captura de pantalla de Gmail personal – Correu de verificació rebut

9.1.2 Iniciar la sessió i tancar la sessió

Per poder iniciar la sessió, simplement cal haver verificat el compte, tal com es mostra a les imatges anteriors. Una vegada fet aquest pas, només cal accedir a l'aplicació i iniciar la sessió fent servir les dades correctament. Si es contempla algun error a l'hora d'introduir les dades d'acord amb les registrades, l'aplicació no completarà l'inici de sessió i mostrarà un missatge visual amb l'error trobat.

Per aquesta funcionalitat, només ha estat necessari consultar la documentació del mateix *Firebase Authentication*, i a partir d'aquesta, s'ha trobat un algorisme anomenat **signInWithEmailAndPassword** per llavors aplicar-lo. Un aspecte que s'ha hagut d'afegir perquè no anava inclòs en l'algorisme per defecte, és la comprovació de si les dades utilitzades per iniciar sessió, són correctes i una vegada comprovades, s'haurà de constatar que el compte d'usuari (en cas que les dades siguin correctes) estigui verificat. Si és així, serà possible iniciar sessió i en cas contrari, saltarà un error automàtic informant sobre per què no s'ha pogut finalitzar l'inici de sessió. El tros de codi que s'encarregarà de comprovar si el compte d'usuari està verificat o no, és l' **isEmailVerified**.

Així doncs, l'algorisme que s'ha emprat en l'aplicació és el següent:



```
public void iniciarSessio(String email, String contrasenya){
    GifImageView spinner = findViewById(R.id.gif_spinner);

    mAuth.signInWithEmailAndPassword(email, contrasenya)
            .addOnCompleteListener( activity: ActivityIniciSessio.this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        FirebaseUser loginUser = mAuth.getCurrentUser();
                        if(loginUser.isEmailVerified())
                        {
                            Log.d( tag: "Verificat", loginUser.getEmail());
                            // Sign in success, update UI with the signed-in user's information
                            Log.d( tag: "Success",  msg: "Inici de sessió correcte");
                            Intent i = new Intent( packageContext: ActivityIniciSessio.this, PrincipalAppActivity.class);
                            i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(i);
                        }
                        else
                        {
                            spinner.setVisibility(View.GONE);
                            Toast.makeText( context: ActivityIniciSessio.this,  text: "Correu sense verificar. Comprova el correu i verifica el compte",
                                    Toast.LENGTH_SHORT).show();
                        }

                    } else {
                        // If sign in fails, display a message to the user.
                        spinner.setVisibility(View.GONE);
                        Toast.makeText( context: ActivityIniciSessio.this,  text: "Error en iniciar sessió. Comprova les dades",
                                Toast.LENGTH_SHORT).show();
                    }
                }
            });

}
```

Figura 9.6: Captura de pantalla del projecte – Algorisme inici de sessió

Per l'acció de *tancar la sessió*, s'ha dissenyat un *Dialog,* el qual prèviament a fer-ho pregunta si estàs segur que vols continuar amb el tancament de la sessió. El que s'encarrega de continuar amb el tancament exitós d'aquesta, és el codi que es mostra a continuació, motiu pel qual aquest és tan important a l'hora de dur a terme aquesta acció: [4]:

```
FirebaseAuth.getInstance().signOut();
```

A més, serà convenient crear un nou Intent que permeti canviar d'Activitat i retornar a la pantalla principal de l'aplicació. D'altra banda, però, també s'hauran d'eliminar la pila d'Activitats que es tinguin fins al moment, i així arribar a procedir a una nova Activitat sense disposar de la possibilitat de retornar a un punt anterior, mentre s'hi estava connectant. Tot seguit, es deixa constància de l'algorisme encarregat de retornar i eliminar la pila d'Activitats actuals:

```
Intent i = new Intent(PerfilFragment.this.getContext(),ActivityLogin.class);
i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(i);
```

9.1.3 Elimina un compte d'usuari

L'opció d'eliminar un compte és de les més fràgils que conté l'aplicació, ja que es tracta d'una acció irreversible des del moment que s'aplica, i això implica que no hi haurà manera de recuperar-lo si llavors es desitja. Amb l'algorisme emprat i una vegada es confirma l'eliminació

del compte, aquest ja s'haurà eliminat definitivament. És per això mateix que l'algorisme consta d'**una segona autenticació d'usuari**, i amb una finalitat al darrere dona l'opció de confirmar que realment es vol esborrar el compte d'usuari. Aquesta confirmació és el que dóna una garantia a l'aplicació podent assegurar que aquesta acció l'ha demanat l'usuari i que, per tant, no ha estat una acció forçada.

D'aquí neix la importància per haver-se d'assegurar que el mateix usuari és l'encarregat d'executar el pas, justament perquè donar per completada aquesta acció implica haver de perdre absolutament totes les dades que el compte tingui emmagatzemades. Aquest, doncs, és el motiu principal pel qual s'adverteix prèviament a l'usuari sobre el qual comporta arribar a eliminar el seu compte definitivament.

Una vegada s'ha entrat novament l'usuari i la contrasenya en el *Dialog* que es genera i d'haver comprovat que les dades són correctes, es procedirà a l'eliminació definitiva del compte, i posteriorment es retornarà a la pantalla principal de l'aplicació a partir d'un Intent, que és molt similar al que s'ha mostrat anteriorment.

En aquest cas, el tros de codi que es necessita per a dur a terme la doble autenticació, és el següent:

```
FirebaseUser currentUser = mAuth.getCurrentUser();
AuthCredential credential =
EmailAuthProvider.getCredential(verificar_correu.getText().toString(),
verificar_contrasenya.getText().toString());
currentUser.reauthenticate(credential)
```

En cas que es confirmi la re-autenticació i sigui correcta, es procedirà a eliminar el compte completament. Contràriament, s'informarà de l'error produït en l'intent de fer-ho. A continuació, es deixa constància de l'algorisme d'eliminació de compte:

```
//BORRAR CONTA
FirebaseUser currentUser = mAuth.getCurrentUser();
AuthCredential credential = EmailAuthProvider.getCredential(verificar_correu.getText().toString(), verificar_contrasenya.getText().toString());
currentUser.reauthenticate(credential).addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if(task.isSuccessful())
        {
            currentUser.delete().addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if(task.isSuccessful())
                    {
                        Toast.makeText( context: ActivityBorrarConta.this,  text: "Compte eliminat correctament.", Toast.LENGTH_LONG).show();
                        Log.d( tag: "clicat",  msg: "Compte eliminat");
                        Intent i = new Intent( packageContext: ActivityBorrarConta.this, ActivityLogin.class);
                        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                        startActivity(i);
                    }
                    else
                    {
                        Log.d( tag: "clicat",  msg: "Error a l'intentar d'esborrar el compte");
                        Toast.makeText( context: ActivityBorrarConta.this,  text: "Error en eliminar el compte. Intenta-ho més tard", Toast.LENGTH_LONG).show();
                    }
                }
            });
        }
        else {
            Toast.makeText( context: ActivityBorrarConta.this,  text: "Error en l'autenticació. Comprova que les dades siguin correctes i torna-ho a provar", Toast

        }
    }
});
```

Figura 9.7: Captura de pantalla del projecte – Algorisme eliminar el compte

9.2 Implementació de les Comunitats

Amb la finalitat d'executar el model de *Comunitats* prèviament vist a l'apartat anterior, va ser necessària una implementació que es comuniqués amb el *Firebase Realitme Database*, per tal de veure una *Comunitat* o bé per poder posar en pràctica alguna de les seves funcionalitats. Es pot tenir en compte per exemple que **afegir una nova Comunitat**, en tots els casos ha estat important fer servir mètodes inclosos a *Firebase*.

9.2.1 Accedir a una Comunitat

Per realitzar l'accés a una comunitat, es va utilitzar un **Item** que se'l va anomenar **ComunitatItem**. El principal ús d'aquesta implementació, va consistir a poder evitar el fet d'haver d'anar cridant de nou la base de dades, per cercar-hi la mateixa comunitat a partir de les diferents accions que apareixen quan es consulta una comunitat en concret. Una vegada consultades les dades d'una comunitat, es passa a crear una *ComunitatItem* amb les dades més significatives d'aquesta i posteriorment es traslladen a una nova pròxima *Activitat*. Les dades que s'agafen d'una comunitat són les següents:

- private String id
- private String poblacio
- private String carrer
- private String num
- private String codiPostal

A partir d'aquesta creació i l'instant en el qual navegui a una nova *Activitat*, es podran obtenir aquestes dades sense necessitat d'accedir de nou a la base de dades, aconseguint molta més eficiència en l'aplicació. Totes les implementacions s'han fet a l'*adapter*, ja que, tal com s'ha esmentat anteriorment, és la llista que conté tots els **Items** que són independents entre si.

La implementació que s'ha fet per passar les dades en un nou *Intent* és la que tot seguit es mostra:

```
ComunitatItem currentItem = llistaComunitats.get(position);
intent.putExtra("comunitatClicada",currentItem);
```

La implementació que es realitza per poder obtenir les dades anteriors en una nova *Activitat*, és la següent:

```
ComunitatItem comunitatClicada;
comunitatClicada = (ComunitatItem)
getIntent().getExtras().getSerializable("comunitatClicada");
```

9.2.2 Inserir i eliminar una Comunitat

Per afegir una nova comunitat a l'aplicació, primer s'hauran de validar totes les dades que s'emplenen del formulari i, en cas que aquestes siguin totes correctes, seguidament es crearà un nou model de classes anomenat **DBComunitat**, i també s'afegirà a la base de dades de *Firebase* per completar el procés [6] .

Les dades que figuren en el model de classe *DBComunitat*, són les mateixes que les que apareixen en el model *Comunitat*, el qual prèviament ha estat reflectit per mitjà d'un **diagrama de classes**.

- private String id_comunitat
- private String nom_carrer
- private Integer num_carrer
- private String planta
- private Integer codi_postal
- private String poblacio
- private String responsable_comunitat
- private Integer telefon_responsable_comunitats

Una vegada comprovada la veracitat de les dades, s'accedirà a la base de dades de l'usuari i se li afegirà la nova comunitat mitjançant un **push.**

El procés que cal seguir per arribar a accedir dins la base de dades de l'usuari, és aquest:

```
DatabaseReference novaComunitat =
myRef.child(mAuth.getCurrentUser().getUid()).push();
```

Després d'accedir-hi, es crea un nou *DBComunitat* amb les dades del formulari i se l'afegeix a la *llista de Comunitats* de l'usuari, la qual s'ha anomenat **nova Comunitat.** Ja per anar acabant, s'elimina la pila d'Activitats que hi hagi fins al moment, i es retorna al menú principal de l'aplicació.

A continuació es mostra l'algorisme complet que s'ha utilitzat:

```
DatabaseReference novaComunitat = myRef.child(mAuth.getCurrentUser().getUid()).push();
id_comunitat = novaComunitat.getKey();
DBComunitat dbComunitat = new DBComunitat(id_comunitat,carrer,Integer.parseInt(numero_carrer),planta_pis,
        Integer.parseInt(codi_post), poble, responsable_nom,Integer.parseInt(responsable_tel));
novaComunitat.setValue(dbComunitat);
Log.d( tag: "Success",  msg: "Nova comunitat guardada");
Intent i = new Intent( packageContext: ActivityAfegirNovaComunitat.this, PrincipalAppActivity.class);
Toast.makeText( context: ActivityAfegirNovaComunitat.this, text: "S'ha afegit la nova comunitat amb èxit",Toast.LENGTH_SHORT).show();
i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(i);
```

Figura 9.8: Captura extreta del projecte – Algorisme afegir nova comunitat

Davant circumstàncies que obliguin a haver d'eliminar una comunitat, primer s'haurà d'accedir-hi, posteriorment seleccionar el mètode d'esborrar, i llavors ja s'obrirà un *dialog* preguntant si realment es desitja esborrar-la. En cas positiu, aquesta serà esborrada automàticament i altrament es cancel·la la operació.

Per portar-ho a terme, es va utilitzar l'algorisme que aprova l'accés a una comunitat a partir del *Firebase* i quant a mètode, el mateix que et dóna el mateix *Firebase* anomenat **removeValue().** Aquest mètode és proporcionat per la mateixa base de dades i permet esborrar l'element que s'estigui seleccionant. Per fer-ho més amè, es deixa constància del seu algorisme complet:

Figura 9.9: Captura extreta del projecte – Algorisme esborrar una comunitat

Per tal de garantir que l'element s'ha eliminat favorablement, es fa ús del mètode **onComplete()**, el qual permet comprovar si la tasca ha estat realitzada sense inconvenients. Aquest procés d'eliminació, serà completat a partir de la pregunta següent: **task.isSuccessful()**, i si tot funciona bé, posteriorment es tancarà l'Activitat actual així com es durà a terme l'eliminació de la comunitat en qüestió, però si pel que sigui hi ha algun problema es cancel·larà l'acció i automàticament informarà de l'error succeït.

9.2.3 Els serveis dintre Comunitat

La implementació realitzada en els Serveis dins l'apartat "Comunitats" és molt similar al sistema que s'ha emprat per poder visualitzar les *Comunitats*.

Aquesta implementació consisteix a generar la **llista** de tots els *Serveis* de la comunitat en qüestió, mitjançant un *adaptador*. Els serveis de la base de dades s'agafaran a partir del model de classes **DBServei**, en el qual cada un d'aquests tindrà els mateixos atributs que s'han vist a la classe "*Servei del diagrama de classes*". Per tenir-ho més present, a continuació es fa un breu recordatori dels atributs que formen part del model de classe DBServei:

- private String data_servei
- private String manteniment_mensual
- private String manteniment_anual
- private String averia

Cada nou servei que passi a formar part de la base de dades, se l'afegirà a la llista de serveis que posteriorment passarà a l'adaptador. Així doncs, cada un d'aquests serveis s'introduirà a la llista considerant-lo un nou **Item individual**, fent que aquests siguin de classe **ServeiComunitatItem**. Les variables que formen part de la llista són les següents:

- private String id_servei
- private String id_comunitat
- private String comunitatActual

El camp **id_servei** fa referència a la data del servei, i cada una de les dates disposa d'un únic servei. En el camp **id_comunitat**, si introdueix la mateixa clau de la comunitat en qüestió. Finalment, la **comunitatActual** serà el camp d'unió entre el de nom del carrer, número d'aquest..

Per tal d'accedir a les mateixes dades d'un *Servei*, s'utilitza el mateix mètode que s'ha utilitzat en el camp *comunitats*: es crea un **Item** i llavors es passa a la nova comunitat. L'única diferència entre un camp i altre, és que aquest segon és de tipus **ServeiComunitatItem**.

El tros de codi que assegurarà la recuperació de les dades en una nova *Activitat* és el següent::

```
ServeiComunitatItem comunitatClicada;
comunitatClicada = (ServeiComunitatItem)
getIntent().getExtras().getSerializable("comunitatClicada");
```

9.3 Implementació dels mètodes de Serveis

Aquest apartat pot semblar confús per que d'entrada sembla molt similar a tot el que s'ha esmentat en l'apartat anterior, però la diferència està en què aquestes implementacions són les que trobem dintre el **fragment de Serveis**, mentre que en l'apartat anterior només es parlava de les implementacions que sortien dintre el **fragment de Comunitats**.

Abans que res, cal destacar que si s'ha fet servir el mateix mètode en aquest apartat i l'anterior, és per tenir la possibilitat de visualitzar la llista amb les *Comunitats* que un usuari tingui actualment.

Una vegada feta aquesta implementació, es va començar a treballar per posar en marxa i donar una funcionalitat al *Calendari*.

9.3.1 Veure el calendari amb els serveis actuals

El calendari que es va crear per l'aplicació és de **codi obert** d'un usuari que havia penjat al seu *GitHub*, tal com ja es va esmentar a l'apartat 7.3 Llibreries i dependències. Aquest calendari tenia l'opció de poder ser utilitzat i personalitzat per part de terceres persones, ja que el propi d'*Android Studio*, en canvi, no permet fer tantes personalitzacions.

En aquest cas, la metodologia implementada va ser la mateixa que apareix en el *GitHub*, tot i que la manera d'implementar els esdeveniments i l'opció de visualitzar els *Serveis* des del mateix calendari, va ser una implementació totalment innovadora i diferent de la que ofereix el mateix usuari al repositori [**10**] .

Així doncs, l'algorisme més important sobre els esdeveniments realitzats va fer-se amb la intenció de reflectir els dies en els quals s'havia proporcionat un Servei. A continuació es mostra una captures d'aquest algorisme:

```
private void addEvents(int month, int year) {
    calendariActual.setTime(new Date());
    calendariActual.set(Calendar.DAY_OF_MONTH, 1);
    Date firstDayOfMonth = calendariActual.getTime();
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            ArrayList<Pair<Integer, Integer>> dates_amb_serveis = new ArrayList<>();
            for (DataSnapshot iterator: snapshot.getChildren()) {

                String[] parts = String.valueOf(iterator.getKey()).split( regex: "-");
                int mes_aux = Integer.parseInt(parts[1]);
                mes_aux = mes_aux - 1;

                String mes_any_servei = mes_aux + "-"+ parts[2];

                if(mes_any_servei.equals(month+ "-" + year)){
                    int serveis_realitzats = 0;
                    DBServei dbServeiTemp = iterator.getValue(DBServei.class);
                    serveis_realitzats = totalServeis(dbServeiTemp);
                    dates_amb_serveis.add(new Pair<>(Integer.parseInt(parts[0])-1, serveis_realitzats));
                }
                else {
                    Toast.makeText( context: ActivityServeiAComunitat.this, text: "S'ha produït un error al crear els edeveniments",Toast.LENGTH_LONG).show();
                }

            }
```

```
            if(!dates_amb_serveis.isEmpty()) {

                for (int i = 0; i < dates_amb_serveis.size(); i++) {
                    calendariActual.setTime(firstDayOfMonth);
                    if (month > -1) {
                        calendariActual.set(Calendar.MONTH, month);
                    }
                    if (year > -1) {
                        calendariActual.set(Calendar.ERA, GregorianCalendar.AD);
                        calendariActual.set(Calendar.YEAR, year);
                    }
                    calendariActual.add(Calendar.DATE, dates_amb_serveis.get(i).first);
                    setToMidnight(calendariActual);
                    long timeInMillis = calendariActual.getTimeInMillis();

                    List<Event> events = getEvents(timeInMillis, dates_amb_serveis.get(i).second);
                    calendari.addEvents(events);
                }

            }

        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });

}
```

Figura 9.10: Captures extretes del projecte – Algorisme per afegir esdeveniments al calendari

Cal destacar que quan s'afegeix qualsevol esdeveniment, es fa a partir de la funció **totalServeis**, tal com es mostra a la primera captura. Allà, s'hi afegirà el servei en funció de si

es tracta únicament una *Averia*, un *manteniment mensual*, un *manteniment anual* o bé un *mix* d'aquestes possibilitats.

Finalment, es contemplarà de quin tipus d'esdeveniment es tracta a partir de la funció **getEvents** i una vegada decidit el tipus, l'esdeveniment serà afegit a la llista. Seguidament, es mostraran els algorismes que s'han esmentat anteriorment:

```
private Integer totalServeis(DBServei dbServei){
    if(dbServei.getAveria().isEmpty() && dbServei.getManteniment_anual().isEmpty() && dbServei.getManteniment_mensual().isEmpty()){
        return 0;
    }
    else if(dbServei.getAveria().isEmpty() && dbServei.getManteniment_anual().isEmpty() && !dbServei.getManteniment_mensual().isEmpty()){
        return 1;
    }
    else if(dbServei.getAveria().isEmpty() && !dbServei.getManteniment_anual().isEmpty() && dbServei.getManteniment_mensual().isEmpty()){
        return 2;
    }
    else if(dbServei.getAveria().isEmpty() && !dbServei.getManteniment_anual().isEmpty() && !dbServei.getManteniment_mensual().isEmpty()){
        return 3;
    }
    else if(!dbServei.getAveria().isEmpty() && dbServei.getManteniment_anual().isEmpty() && dbServei.getManteniment_mensual().isEmpty()){
        return 4;
    }
    else if(!dbServei.getAveria().isEmpty() && dbServei.getManteniment_anual().isEmpty() && !dbServei.getManteniment_mensual().isEmpty()){
        return 5;
    }
    else if(!dbServei.getAveria().isEmpty() && !dbServei.getManteniment_anual().isEmpty() && dbServei.getManteniment_mensual().isEmpty()){
        return 6;
    }
    else {
        return 7;
    }
}
```

Figura 9.11: Captura extreta del projecte – Algorisme totalServeis per determinar el tipus de d'esdeveniment

```
private List<Event> getEvents(long timeInMillis, int totalServeis) {
    if (totalServeis == 1) {
        return Arrays.asList(new Event(Color.argb( alpha: 255,  red: 0,  green: 255,  blue: 0), timeInMillis,  data: "Event 1 at " + new Date(timeInMillis)));
    } else if ( totalServeis == 2) {
        return Arrays.asList(new Event(Color.argb( alpha: 255,  red: 255,  green: 0,  blue: 255), timeInMillis,  data: "Event 2 at " + new Date(timeInMillis)));

    } else if(totalServeis == 3){
        return Arrays.asList(
                new Event(Color.argb( alpha: 255,  red: 0,  green: 255,  blue: 0), timeInMillis,  data: "Event 1 at " + new Date(timeInMillis)),
                new Event(Color.argb( alpha: 255,  red: 255,  green: 0,  blue: 255), timeInMillis,  data: "Event 3 at " + new Date(timeInMillis)));
    } else if(totalServeis == 4){
        return Arrays.asList(new Event(Color.argb( alpha: 255,  red: 255,  green: 128,  blue: 0), timeInMillis,  data: "Event 3 at " + new Date(timeInMillis)));

    } else if(totalServeis == 5){
        return Arrays.asList(
                new Event(Color.argb( alpha: 255,  red: 0,  green: 255,  blue: 0), timeInMillis,  data: "Event 1 at " + new Date(timeInMillis)),
                new Event(Color.argb( alpha: 255,  red: 255,  green: 128,  blue: 0), timeInMillis,  data: "Event 3 at " + new Date(timeInMillis)));

    } else if(totalServeis == 6){
        return Arrays.asList(
                new Event(Color.argb( alpha: 255,  red: 255,  green: 0,  blue: 255), timeInMillis,  data: "Event 2 at " + new Date(timeInMillis)),
                new Event(Color.argb( alpha: 255,  red: 255,  green: 128,  blue: 0), timeInMillis,  data: "Event 3 at " + new Date(timeInMillis)));
    }
    else {
        return Arrays.asList(
                new Event(Color.argb( alpha: 255,  red: 0,  green: 255,  blue: 0), timeInMillis,  data: "Event 1 at " + new Date(timeInMillis)),
                new Event(Color.argb( alpha: 255,  red: 255,  green: 0,  blue: 255), timeInMillis,  data: "Event 2 at " + new Date(timeInMillis)),
                new Event(Color.argb( alpha: 255,  red: 255,  green: 128,  blue: 0), timeInMillis,  data: "Event 3 at " + new Date(timeInMillis)));
    }
}
```

Figura 9.12: Captura extreta del projecte – Algorisme getEvents per afegir l'esdeveniment que toca al calendari

9.3.2 Afegir, modificar i eliminar Serveis

Amb la finalitat de dur a terme implementacions com *afegir, modificar i eliminar serveis*, es va dissenyar un algorisme amb similituds al *d'afegir i eliminar una Comunitat*..

La idea que es contempla a l'hora **d'afegir un nou Servei**, consisteix a entrar les dades al formulari que apareix en *l'Activitat* del calendari. Una vegada introduïdes les dades i què l'usuari hagi premut el botó de guardar canvis, l'algorisme crearà un nou model **DBServei** amb les dades que constin en el formulari, i s'utilitzarà com a identificador la mateixa data en la qual es realitzi un servei.

Una vegada finalitzat aquest procés, es notificarà per mitjà d'un missatge temporal que el servei ha estat guardat, llavors s'actualitzarà el calendari i aquest mostrarà el nou esdeveniment en data del dia que l'hagis afegit. L'algorisme emprat en aquest cas, és el següent:

```
public void guarda_actualitza_servei(String dataSeleccionada){
    DBServei dbServei_actual = new DBServei(dataSeleccionada, m_mensual.getText().toString(), m_anual.getText().toString(), averia.getText().toString());
    myRef.child(dataSeleccionada).setValue(dbServei_actual);
    Log.d( tag: "Success",  msg: "Nou servei guardat");
    m_mensual.clearFocus();
    m_anual.clearFocus();
    averia.clearFocus();
    Toast.makeText( context: ActivityServeiAComunitat.this, text: "Nou servei afegit",Toast.LENGTH_SHORT).show();
    calendari.removeAllEvents();

}
```

Figura 9.13: Captura extreta del projecte – Algorisme per afegir un nou servei

Per **modificar** qualsevol dada d'un servei, s'ha d'actualitzar el camp del formulari en el qual vulguis fer canvis i una vegada s'hagin fet, hauràs de prémer el botó per guardar els canvis realitzats. A partir d'aquí, l'algorisme és qui s'encarrega d'accedir en el servei i modificar les seves dades pertinents.

Finalment, si el que es desitja és eliminar un Servei, l'usuari haurà de prémer a la icona d'esborrar, seguidament s'obrirà un nou *dialog* el qual li preguntarà de nou si es vol eliminar el servei, i així confirmar-ho per última vegada. En cas afirmatiu, s'utilitzarà el mètode **removeValue()** que es pot invocar a partir de l'ús del *Firebase* tal com també és possible en el cas de les comunitats. De la mateixa manera, l'acció també es confirmarà a partir de la funció **onComplete()** i posteriorment mitjançant el mètode **task.isSuccessful()**, amb la finalitat de determinar si s'ha pogut esborrar correctament. Si tot ha anat bé, el servei s'esborrarà de la base de dades, i això serà el que es notificarà a partir d'un missatge temporal. Finalment, s'actualitzaran de nou els esdeveniments per tal que ja no aparegui el que ha estat esborrat recentment. En cas contrari, l'algorisme retornarà un missatge temporal avisant de què l'acció ha fracassat.

A continuació es visualitzarà amb més detall l'algorisme que s'acaba de comentar:

```
public void showAlertDialogButtonClicked(View view) {

    // setup the alert builder
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Estàs segur que vols eliminar el servei?");
    builder.setMessage("Perdràs totes les dades i no podràs recuperar-les més endavant");

    // add the buttons
    builder.setPositiveButton( text: "Esborra", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Log.d( tag: "PROVA", dataSeleccionada);
            myRef.child(dataSeleccionada).removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if(task.isSuccessful()){
                        String[] data_separada = String.valueOf(dataSeleccionada).split( regex: "-");
                        int mes_aux = Integer.parseInt(data_separada[1]);
                        mes_aux = mes_aux - 1;
                        veureServei(dataSeleccionada, mes_aux,Integer.parseInt(data_separada[2]));
                        Toast.makeText( context: ActivityServeiAComunitat.this, text: "Servei esborrat correctament",Toast.LENGTH_SHORT).show();

                    }else{
                        Toast.makeText( context: ActivityServeiAComunitat.this, text: "Error en eliminar el servei. Intenta-ho més tard",Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    });

    builder.setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
    });

    // create and show the alert dialog
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

Figura 9.14: Captura extreta del projecte – Algorisme per eliminar un servei

10. IMPLANTACIÓ I RESULTATS

En aquest apartat s'hi mostrarà els resultats de les funcionalitats que s'han desenvolupat al llarg del projecte, mitjançant imatges extretes de la mateixa aplicació.

10.1 Resultats dels requisits de l'aplicació

En aquest apartat s'anirà mostrant el resultat final de l'aplicació per cada un dels requisits que s'havia acordat amb el client.

10.1.1 Pantalla a l'inici de l'aplicació

Quan s'inicia l'aplicació, es pot veure la icona de la mateixa empresa i seguidament una apareix una nova finestra la qual és un comprovant per saber si algú té oberta la sessió en el compte. En cas afirmatiu, es mostrarà el compte i es posarà en marxa la sessió directament. Altrament, es redirigirà a la pantalla principal de la sessió personal.

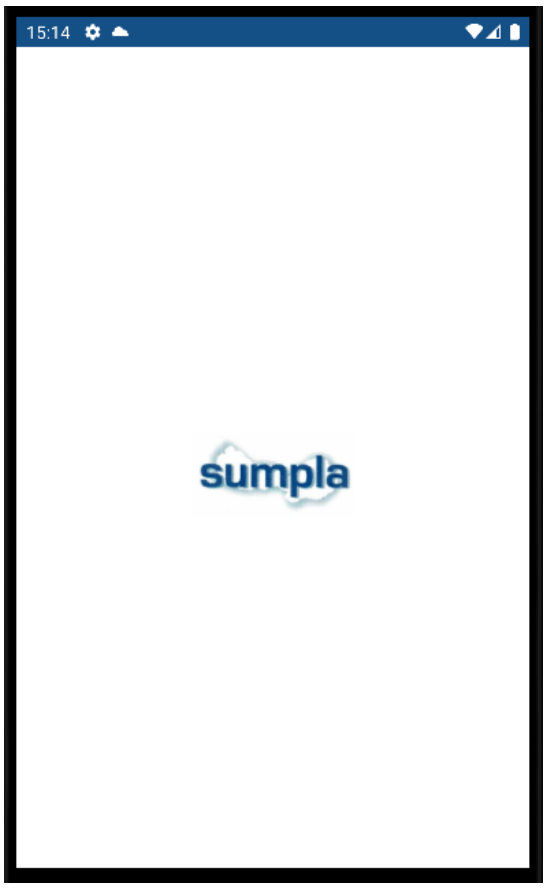


Figura 10.1: Captura extreta de l'aplicació – Inici de l'aplicació

Figura 10.2: Captura extreta de l'aplicació – Cap compte iniciat

Figura 10.3: Captura extreta de SUMPLAPP – Es disposa d'un compte actiu

10.1.2 Pantalla principal de l'aplicació

Aquesta pantalla apareix en el moment d'ingressar a l'aplicació sempre que no hi hagi cap altra compte obert. Però si, per contra, hi hagués un compte obert, directament s'obriria el menú de Serveis que més endavant serà mostrat.

Figura 10.4: Captura extreta de l'app – Pantalla inicial per usuari no identificat

10.1.3 Pantalla inici de sessió

A continuació la pantalla d'Iniciar sessió. Es mostraran els possibles errors que salten en cas d'entrar dades errònies o camps buits.

Figura 10.5: Captura extreta de l'aplicació - Intent d'inici de sessió (el fet de deixar camps buits fa saltar un error)

Figura 10.6: Captura extreta del software - Intent d'inici de sessió sense posar contrasenya

Figura 10.7: Captura extreta de l'aplicació - Intent d'inici de sessió amb dades errònies

10.1.4 Registrar nou usuari

Seguidament, es deixa constància de com és la pantalla en la qual apareix el formulari de registre d'un nou usuari. També s'hi pot veure possibles errors, els quals es retornaran si s'entren dades incorrectes o inexistent.

Figura 10.8: Captura extreta de l'aplicació SUMPLAPP – Pantalla registrar nou usuari

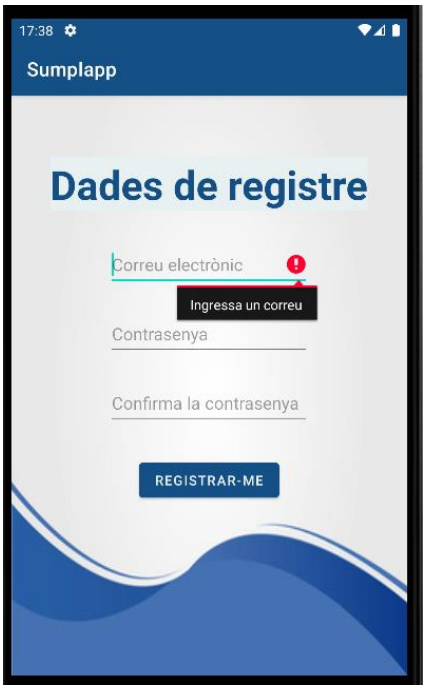


Figura 10.9: Captura extreta de l'aplicació – Error que retorna l'app, quan es prem a registrar-se, sense haver entrat cap dada prèviament

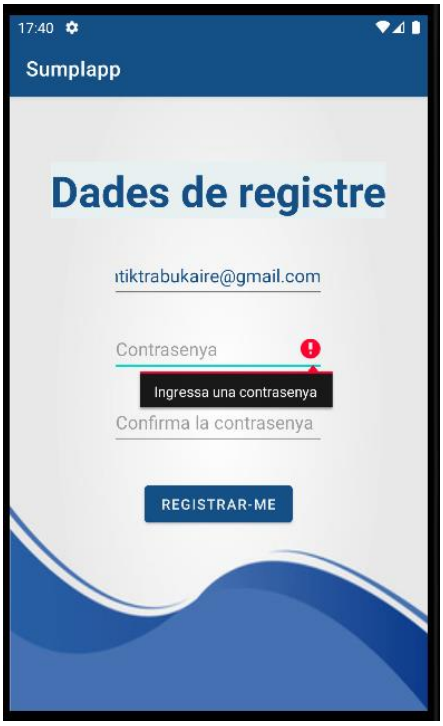


Figura 10.10: Captura extreta de l'aplicació – Error que retorna el software per no entrar la contrasenya

Figura 10.11: Captura extreta de l'app – Error que retorna l'app quan no coincideixen les contrasenyes

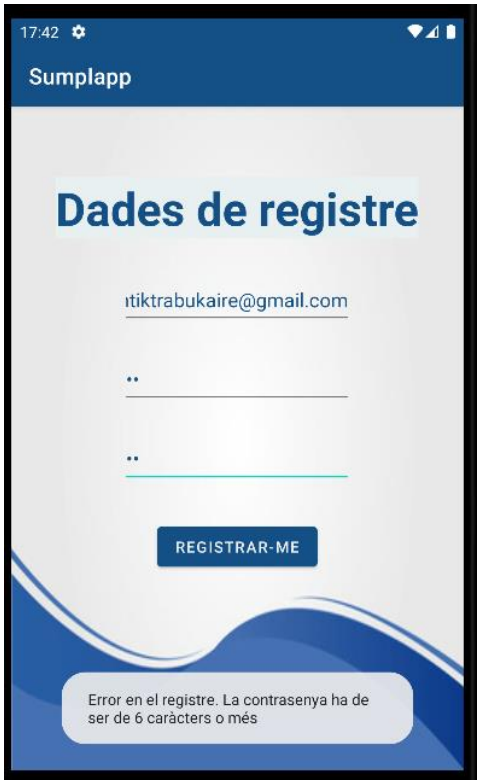


Figura 10.12: Captura extreta de SUMPLAPP – Error que retorna l'app quan la contrasenya és massa curta

Figura 10.13: Captura extreta de l'aplicació – Error que retorna l'app quan s'intenta crear un nou compte d'usuari amb un correu ja registrat

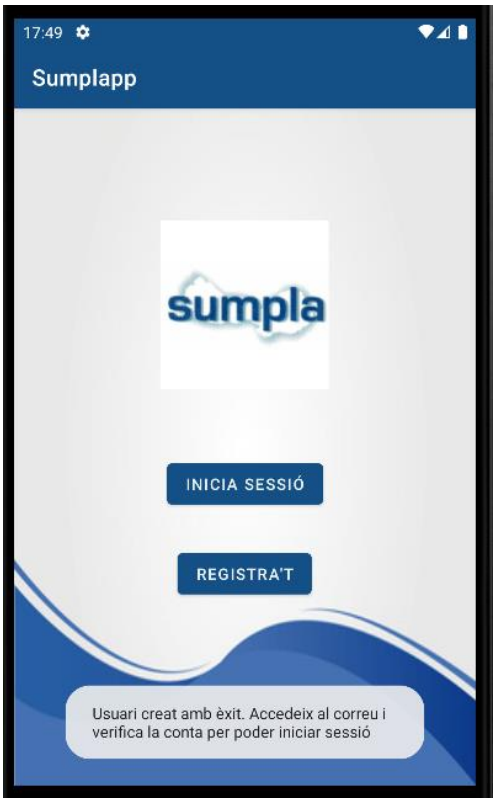


Figura 10.14: Captura extreta de l'aplicació – L'usuari s'ha pogut registrar i l'app retorna al menú principal

Arribats a aquest punt, s'espera que l'usuari accedeixi al correu que li han enviat per verificar el seu compte. Aleshores ja podrà començar a utilitzar l'aplicació, amb les diferents funcions que aquesta disposa.

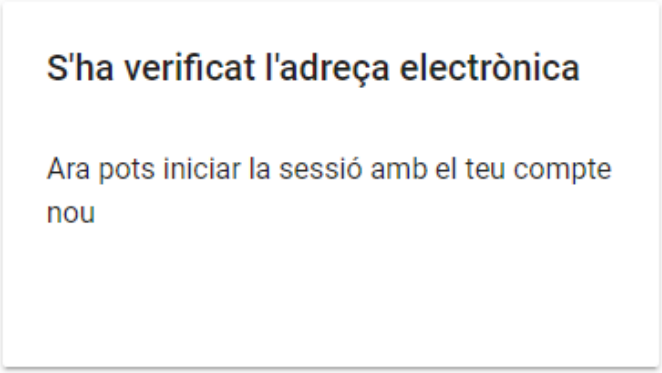


Figura 10.15: Captura extreta del correu Gmail – Missatge que es rep, quan el compte de l'aplicació ha estat verificat

10.1.5 Menú de perfil

En aquesta pantalla es veurà com ha quedat el menú *Perfil* que té l'usuari:

Figura 10.16: Captura extreta de l'aplicació – Menú perfil amb les diverses opcions

10.1.6 Editar el perfil

En aquest apartat, s'exposaran les diferents opcions que té l'usuari a l'hora de modificar les seves dades: canviar el correu i la contrasenya.

Figura 10.17: Captura extreta de l'app – Menú editar perfil

10.1.6.1 Canviar el correu electrònic

Per dur a terme aquesta funció, serà convenient entrar les dades actuals de l'usuari, i així confirmar que és ella la persona que sol·licita el canvi, a més d'afegir-hi un nou correuelectrònic.

Figura 10.18: Captura extreta de l'aplicació – Es demana actualitzar el correu sense posar al seu abast les dades necessàries

Figura 10.19: Captura extreta de SUMPLAPP – Dades incorrectes per autenticar el compte

10.1.6.2 Canviar contrasenya

Per canviar la contrasenya, només caldrà que l'usuari entri el seu correu. Es comprovarà que el correu sigui autèntic de l'usuari. En cas afirmatiu, s'enviarà un correu amb les passes per canviar la contrasenya.

Figura 10.20: Captura extreta de l'aplicació – Pantalla per canviar la contrasenya

Figura 10.21: Captura extreta del software – Error si es prem Restablir deixant el camp de correu buit

Figura 10.22: Captura extreta de l'aplicació SUMPLAPP – Error que es mostra en cas d'entrar un correu incorrecte

Figura 10.23: Captura extreta de l'aplicació – S'ha enviat un correu per poder restablir la contrasenya

Figura 10.24: Captura extreta de Gmail – Correu per canviar la contrasenya rebut

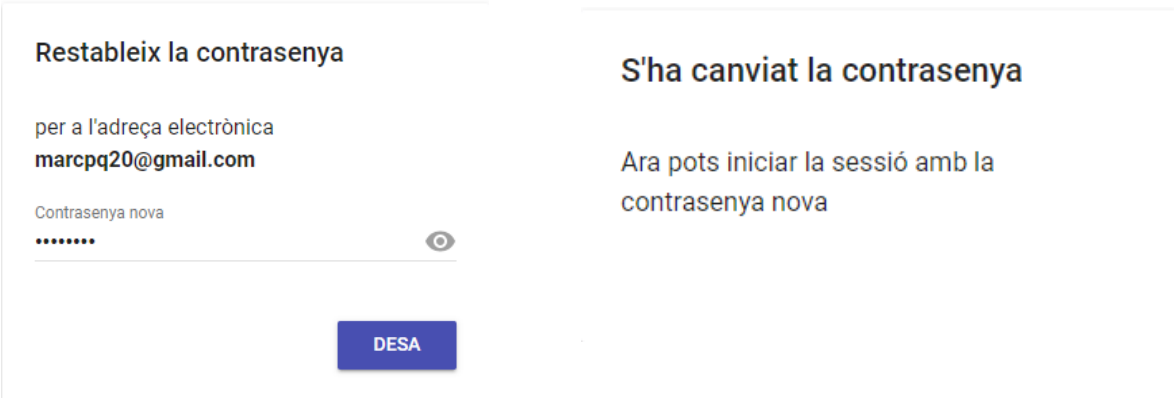


Figura 10.25: Captura extreta del Gmail – Formulari per restablir la contrasenya i missatge que arriba una vegada desada la nova contrasenya

10.1.7 Política de drets d'autors

En aquesta pantalla es mostrarà tota la informació relacionada amb la política de privacitat i l'ús correcte de l'aplicació, entre altres:

Figura 10.26: Captura extreta de l'aplicació – Política de drets d'autor

10.1.8 Tancar la sessió

S'obre un *Dialog* preguntant de nou si es vol tancar la sessió per confirmar l'acció. En cas afirmatiu, aquesta es tanca i es retorna a la pantalla principal de l'aplicació.

Figura 10.27: Captura extreta de l'app – Menú per tancar la sessió actual

10.1.9 Eliminar un compte

Aquesta pantalla mostra una explicació detallada de què suposa eliminar el compte definitivament. En cas de prémer al botó "Elimina el compte", s'obrirà un *Dialog* demanant una re-autenticació de l'usuari per confirmar que ha estat ell qui ha demanat que es faci aquesta operació. Així doncs, un cop validades les dades s'eliminarà el compte definitivament, i aquest ja no es podrà recuperar.

Figura 10.28: Captura extreta de l'aplicació SUMPLAPP – Elimina el compte

Figura 10.29: Captura extreta del software – Formulari de re-autenticació previ a esborrar un compte

10.1.10 Menú de comunitats

En aquest menú si podran veure totes les comunitats que l'usuari hagi anat afegint. Tanmateix, es podran anar seleccionant individualment, i això permetrà accedir a la seva informació o bé dura terme accions sobre aquesta.

Figura 10.30: Captura extreta de l'aplicació – Menú de comunitats sense tenir cap comunitat afegida a la llista

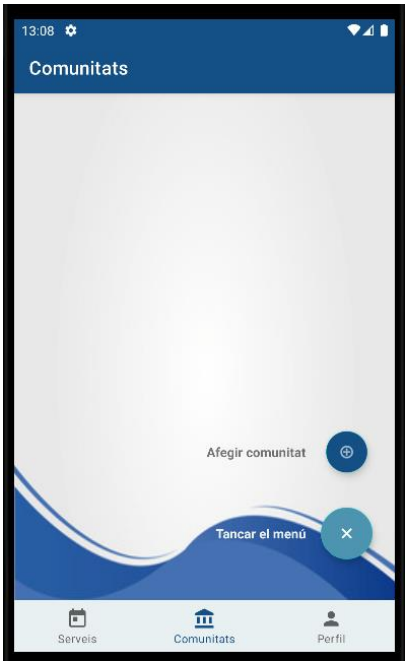


Figura 10.31: Captura extreta de SUMPLAPP – Botó del menú de comunitats desplegat

10.1.10.1 Afegir una comunitat

Si no s'omplen les dades correctament de la nova comunitat, es retornarà un error tal com s'ha vist en exemples anteriors, destacant quin és el camp erroni. Quan les dades d'aquesta siguin correctes, s'afegirà a la llista de comunitats i es retornarà al **menú de Comunitats** tal com a continuació veurem a partir de les imatges:

Figura 10.32: Captura extreta de l'aplicació – Formulari afegir una comunitat

Figura 10.33: Captura extreta de l'app – Menú comunitats una vegada afegida una comunitat

10.1.11 Consultar una comunitat

A continuació, es deixa una captura de sobre la pantalla a la qual s'accedeix en el moment de seleccionar una comunitat en concret. Com a exemple pràctic d'aquesta funció, s'entrarà a la comunitat "Ramon Llull", i una vegada allà es podrà veure totes les dades de la comunitat en qüestió:

Figura 10.34: Captura extreta de l'aplicació – Consultar una comunitat

Amb el botó "+" s'accedeix a totes les funcionalitats disponibles per la comunitat. Més concretament, les opcions possibles de realitzar, són aquestes: *editar les dades de la comunitat, esborrar la comunitat, veure la localització de la comunitat i veure els serveis de la comunitat.*

10.1.11.1 Eliminar comunitat

S'obre un *Dialog* tornant a preguntar si es vol eliminar definitivament la comunitat. En cas afirmatiu, s'elimina la comunitat i es retorna al menú principal.

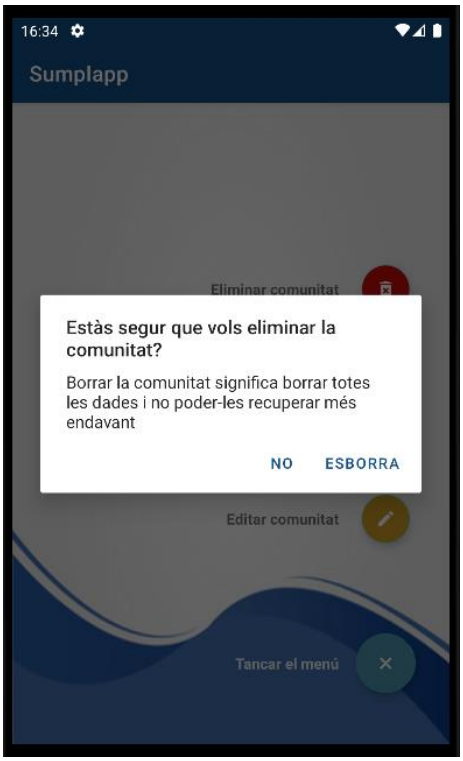


Figura 10.35: Captura extreta del software – Eliminar una comunitat

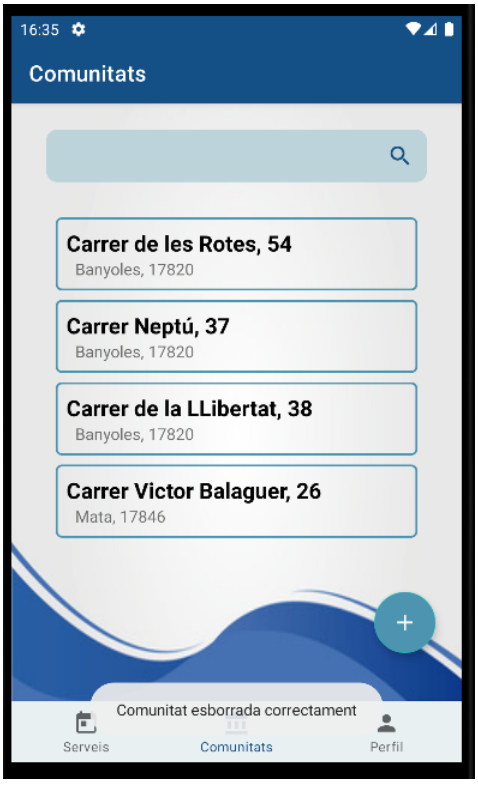


Figura 10.36: Captura extreta de l'aplicació SUMPLAPP – Comunitat eliminada correctament

10.1.11.2 Veure localització d'una comunitat

A partir de la següent pantalla, es mostra la localització de la comunitat via *Google Maps* [18] . Prèviament a la localització d'aquesta, serà imprescindible que l'usuari permeti l'accés al GPS per poder-ne fer un bon ús.

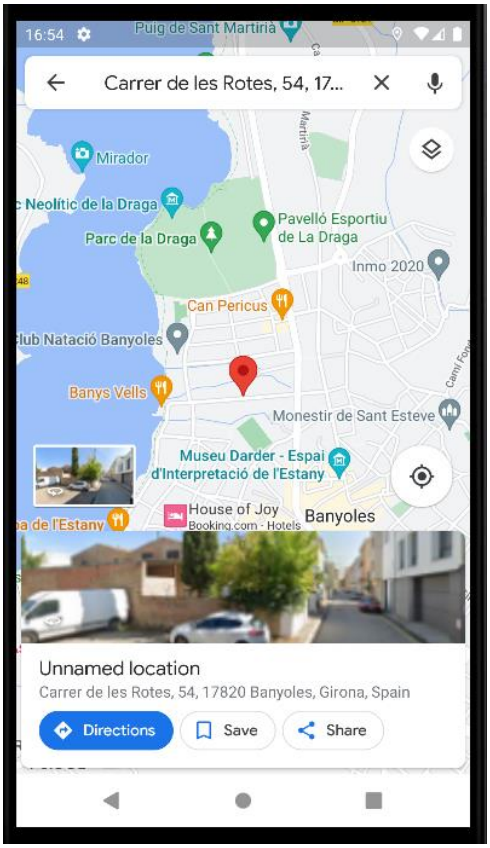


Figura 10.37: Captura extreta de l'app – Localització de la comunitat

10.1.11.3 Editar les dades d'una comunitat

Si s'arriba en aquesta pantalla, l'usuari podrà editar les dades de la comunitat, sempre que respecti els requisits específics dels camps que estigui modificant.

Figura 10.38: Captura extreta de l'aplicació – Editar dades d'una comunitat

10.1.11.4 Veure els serveis d'una comunitat

En aquesta pantalla, l'usuari podrà veure tots els serveis que hagi realitzat sobre la comunitat en qüestió. L'usuari també podrà eliminar i modificar els serveis si així ho desitja. Els serveis que s'ofereixi a una comunitat podran ordenar-se per data ascendent i descendent segons quan es facin, és a dir, que s'ordenaran per "serveis amb més antiguitat" i per "serveis més recents". Finalment, hi haurà l'opció de descarregar els serveis fets en una comunitat en format *Excel* [15] . A continuació es veuran les pantalles que s'acaben de definir per escrit:

Figura 10.39: Captura extreta de l'app – Historial de serveis quan encara no se n'ha afegit cap

Figura 10.40: Captura extreta del software – Historial de serveis amb alguns serveis registrats

Figura 10.41: Captura extreta de l'aplicació – Descarregant l'historial de serveis

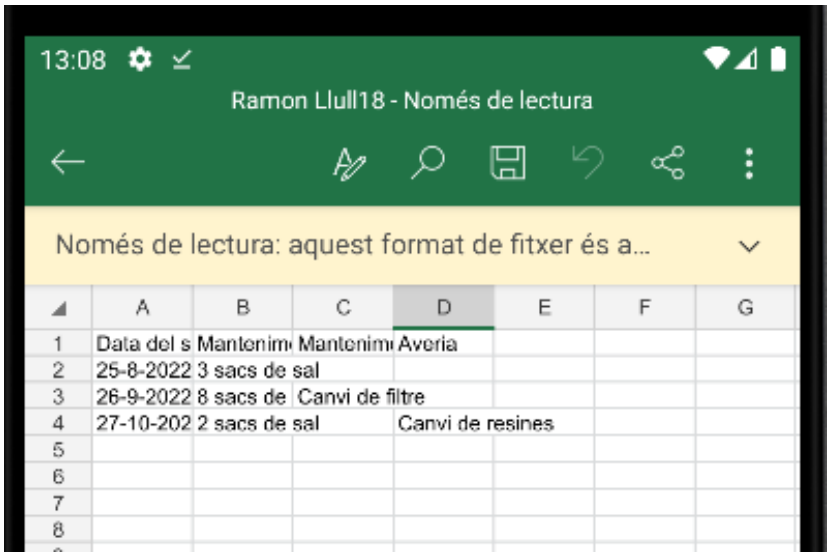


Figura 10.42: Captura extreta de l'emulador – Vista del fitxer Excel descarregat

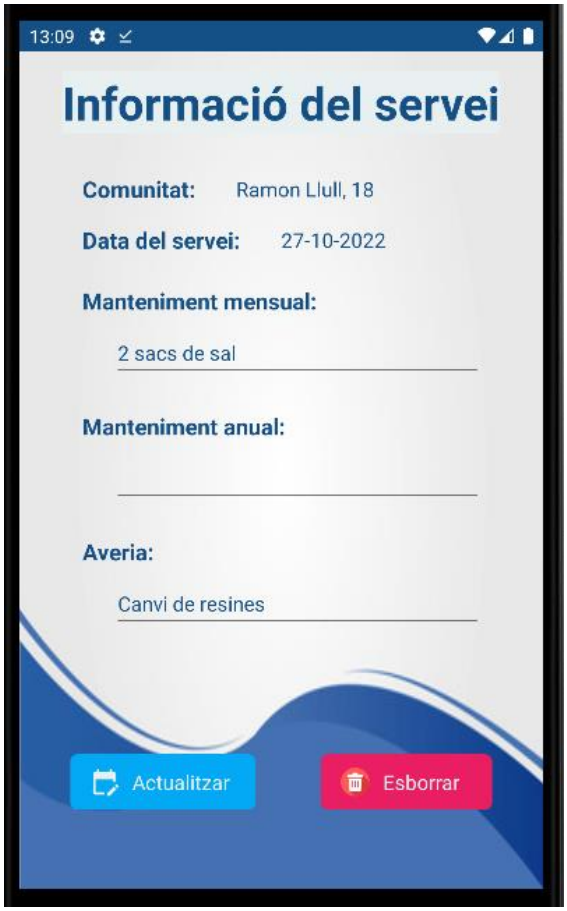


Figura 10.43: Captura extreta de l'aplicació – Consulta d'un dels servei de la comunitat

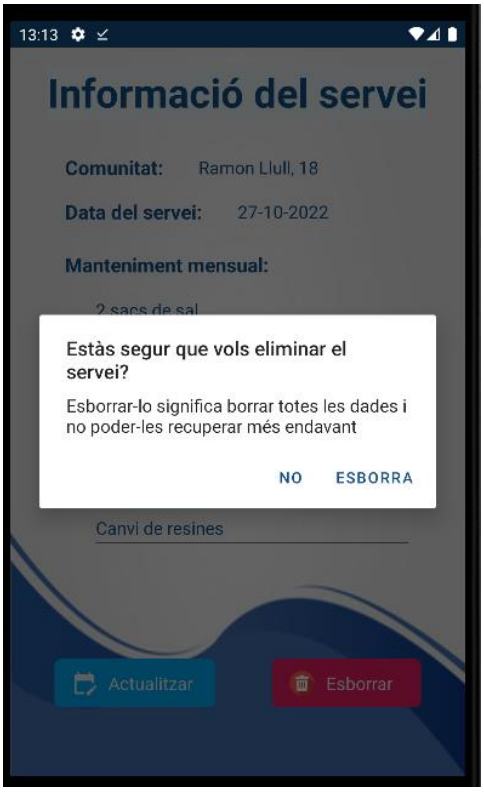


Figura 10.44: Captura extreta de l'app SUMPLAPP – Dialog per esborrar un servei de la comunitat

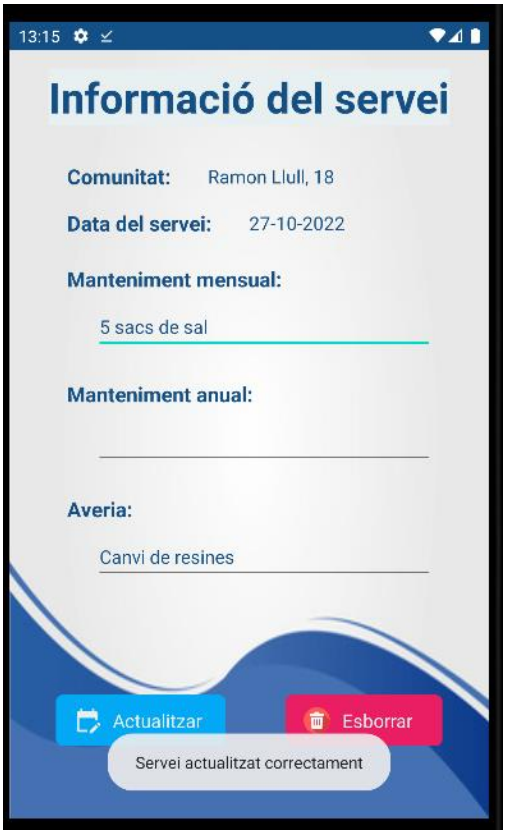


Figura 10.45: Captura extreta del software – Actualització d'un servei de la comunitat

10.1.12 Menú de Serveis

A partir de la següent captura, es pot veure la pantalla que mostra un cercador i una llista amb les diferents comunitats que l'usuari té entrades a la base de dades.

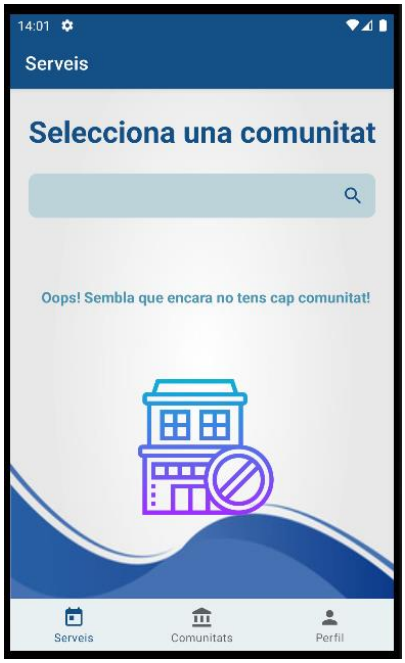


Figura 10.46: Captura extreta de l'aplicació – Menú de serveis sense serveis

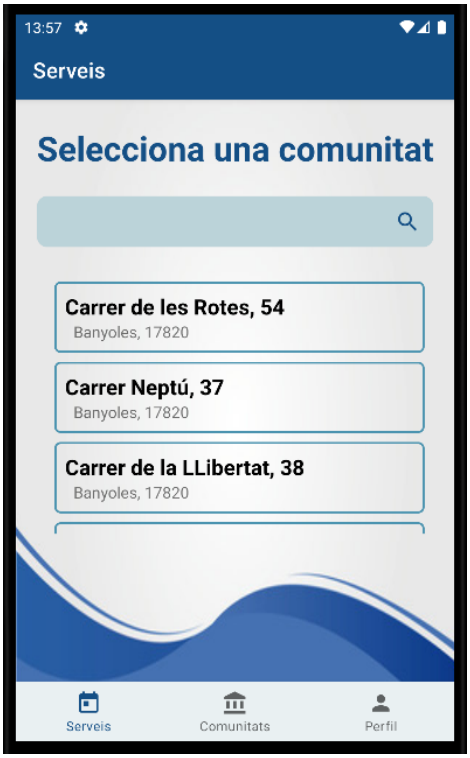


Figura 10.47: Captura extreta de l'app – Menú de serveis amb alguns serveis

10.1.12.1 Consultar el calendari d'una comunitat

En aquesta pantalla es mostrarà un calendari i un formulari amb els camps corresponents al servei.

En el calendari hi haurà marcats els dies en què hi apareix algun servei, i depèn del tipus de servei estarà marcat d'un color o un altra. Hi ha una llegenda per poder saber en tot moment de quin tipus de servei s'ha realitzat: el marcatge verd quan es tracta d'un manteniment mensual; el marcatge rosa si es tracta d'un manteniment anual i el taronja si es tracta d'una averia. Tanmateix, cal remarcar que es poden trobar dies els quals tinguin entrats varis d'aquests serveis.

Figura 10.48: Captura extreta de SUMPLAPP – Veure calendari amb els serveis

Tal com es pot observar a partir de la imatge anterior, el dia seleccionat està marcat en negre. Com que la data seleccionada i l'actual coincideixen no s'hi veu cap canvi, però quan se'n selecciona una altra, aquesta queda marcada de color negre, mentre que la data d'avui destacarà per mitjà del color vermell. Quan s'arribi a final de mes i aquest canviï, únicament es marcarà en negre el dia que hi hagi seleccionat [8] .

Figura 10.49: Captura extreta de l'aplicació – Data seleccionada diferent a la data actual

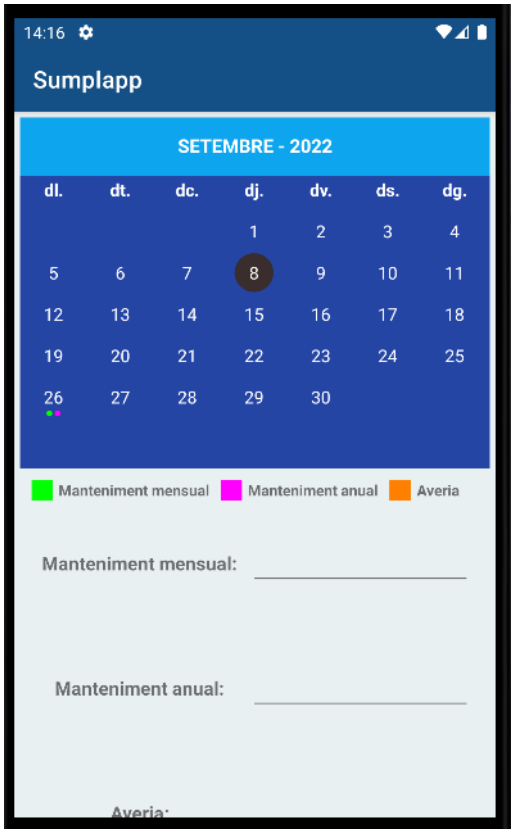


Figura 10.50: Captura extreta del software – Canvi de mes en el calendari

10.1.12.2 Consultar un dia amb un servei

Una altra de les moltes funcions que ofereix l'aplicació, és la de consultar un dia per veure quin servei s'hi ofereix. Així doncs, només cal seleccionar el dia que es desitgi consultar. A partir d'aquí, el formulari de la part inferior del calendari s'omplirà amb les dades que constin en el servei clicat.

Figura 10.51: Captura extreta de l'aplicació SUMPLAPP – Consultar un servei des del calendari

10.1.12.3 Afegir o modificar les dades d'un servei

Per afegir un nou servei o bé modificar-ne les dades, només cal seleccionar una data que no tingui entrat cap servei i posteriorment omplir el formulari amb les dades del servei realitzat. Ara bé, no serà possible afegir un servei si tots els camps del formulari siguin buits, però, en canvi, es podrà fer en cas que algun d'aquests camps contingui informació i no estigui buit. Un cop s'hagi afegit un servei només s'haurà de clicar al botó "Actualitza" per afegir-lo, i llavors apareixerà un text temporal avisant de què l'entrada del nou servei que ha estat afegit, s'ha fet amb èxit.

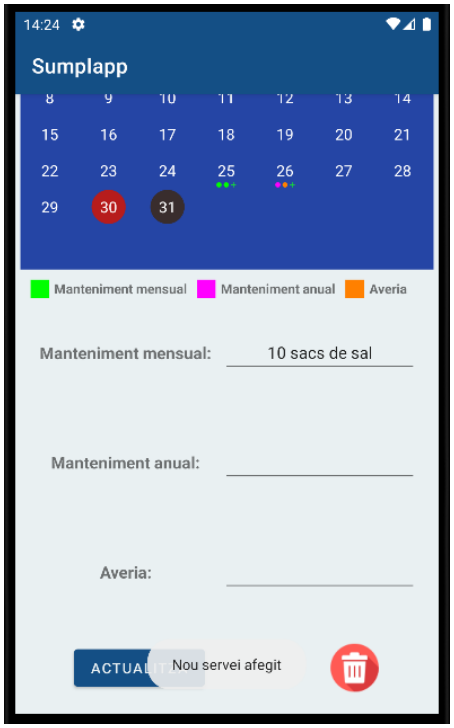


Figura 10.52: Captura extreta de l'app – Nou servei afegit

10.1.12.4 Modificar un servei

En el cas de voler modificar un servei, s'haurà de seleccionar una data en la qual hi hagi constància d'un servei i modificar-ne el text. Una vegada aquest servei hagi estat modificat, es procedirà a clicar "Actualitza", acció la qual farà aparèixer un missatge temporal avisant del canvi que s'ha realitzat.

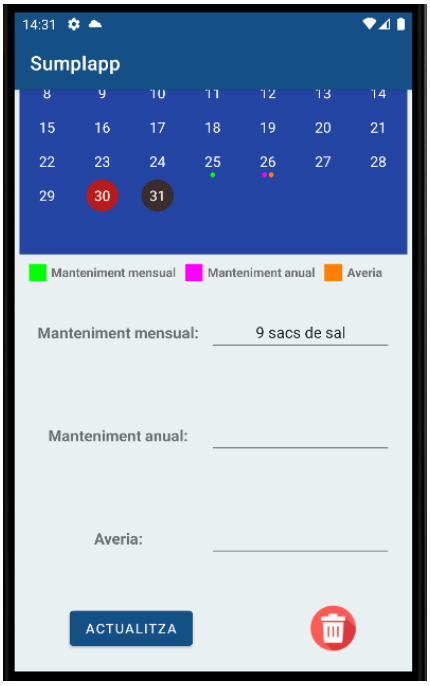


Figura 10.53: Captura extreta de l'aplicació – Servei actualitzat

10.1.12.5 Eliminar un servei

Per tal d'eliminar un servei si es desitja, s'haurà de seleccionar la data del servei que es vol esborrar i prémer a la icona de les escombraries. Una vegada s'hagi fet aquest primer pas, s'obrirà un *Dialog* preguntant si realment es vol eliminar el servei. En cas afirmatiu, s'esborrarà el servei i aquest deixarà d'aparèixer en el calendari tot suprimint l'esdeveniment que si havia assignat, i es mostrarà un missatge temporal per notificar a l'usuari que s'ha esborrat un servei. En cas contrari, es cancel·la l'eliminació i aquesta no es durà terme.

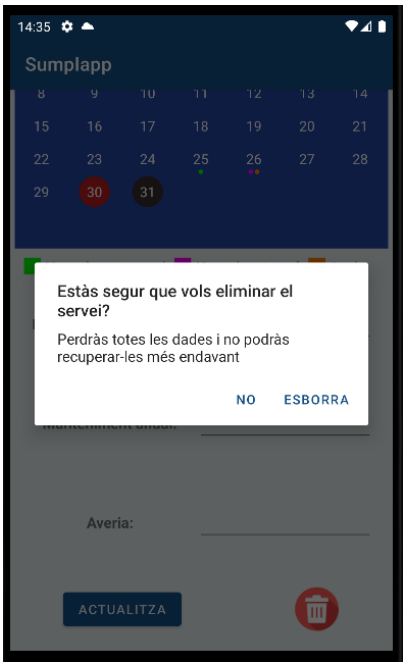


Figura 10.54: Captura extreta del software – Dialog previ a esborrar un servei

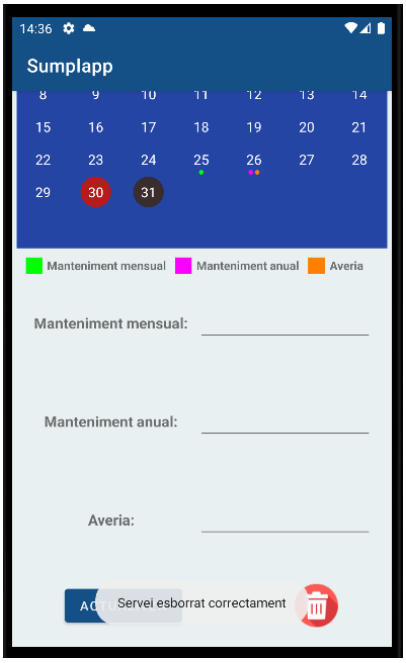


Figura 10.55: Captura extreta de l'aplicació SUMPLAPP – Servei esborrat

11. CONCLUSIONS

El caràcter principal d'aquest treball tenia relació amb el fet de crear una aplicació des de zero. Més concretament, aquesta va ser pensada perquè fos compatible amb sistema operatiu *Android*, i amb una finalitat tan clara com ho és poder-la utilitzar en un període de temps a curt termini, per satisfer i cobrir certes tasques del món laboral ordinari.

Així doncs, a partir d'aquest enfocament previ, es van definir tres diferenciades entre si fases per aquest projecte: la fase inicial, la fase de desenvolupament i la fase final.

La fase inicial, en la qual s'ha realitzat tot el plantejament global de l'aplicació, les reunions amb el client que van permetre acordar l'ús que es volia donar a l'aplicació i també quines funcionalitats requeria aquesta. Tanmateix, es va fer investigació i recerca de mètodes i algorismes que afavorissin el fet de dur a terme les diferents practicitats del projecte i un primer disseny de l'aplicació per tenir una primera idea inicial de com aquesta hauria de ser en el moment de donar-la per acabada.

La següent fase és la de desenvolupament, i al llarg d'aquesta s'ha dut a terme tota la creació de l'aplicació, tant pel que fa a la part lògica de les diferents funcionalitats, com també la part de disseny de les diferents pantalles.

Finalment i ja entrant a comentar la fase final d'aquest projecte, aquesta és la fase que ha ofert la possibilitat d'arreglar petits *bugs* i errors que encara estaven presents a l'aplicació, però amés, i també s'ha acabat fet un canvi de disseny de l'aplicació, per tal que visualment fos de més bon veure, és a dir, procurant que tingués un aspecte més elegant i actualitzat.

Així doncs, tenint en compte les conclusions que s'han extret es podria considerar que a trets generals s'han complert els objectius proposats inicialment:

- L'aplicació s'ha pogut deixar enllestida fins i tot abans del temps que s'havia previst en un principi, tot obtenint bons resultats pel que fa a la implementació i disseny final, el qual també s'ajusta al gust i demandes del client

- Totes les funcionalitats i requisits que s'havien plantejat inicialment per complir durant tot el projecte, realment s'han dut a terme i amb una bona posada en pràctica ara que ja s'ha arribat al final del projecte. No obstant això, cal destacar que per assolir-ho es va haver de refer el primer plantejament del diagrama de casos d'ús que s'havia esmentat inicialment. Una vegada arreglat aquest plantejament, ja es van dur a terme totes les funcionalitats i les implementacions tal com requeria el client.

- Les implementacions i mètodes cercats durant l'inici d'aquest projecte, van ser implementats amb èxit i deixant anar la imaginació amb pinzellades de personalització personal per poder-ho enfocar el disseny del projecte d'acord amb el gust del client.

12. TREBALL FUTUR

Com a treball futur, s'espera arribar a profunditzar en la millora de l'aplicació i treballar-hi perquè deixi entrar-hi més dades que també són importants a tenir en compte de cara a l'empresa client, com ara *l'IBAN* de les comunitats entre altres dades de les quals encara hi ha pendent una validació. Tanmateix, s'espera poder passar l'aplicació actual a Producció perquè sigui possible descarregar-la, sigui mitjançant una *APK* o la mateixa *Play Store*, i així poder-li donar encara més ús a escala de funcionalitats, que de fet és el propòsit pel qual ha estat creada.

En un futur també hi ha la intenció de canviar una mica el disseny de l'aplicació, amb la finalitat de poder-la personalitzar més del que ara està. A més, no es descarta afegir l'opció de poder canviar entre el que seria el tema actual, i un nou tema més Fosc.

Un altre factor sobre el qual es preveu treballar per millorar-lo, és el de la seguretat. Tot i que les dades ja estan vinculades amb *Google* i les dades són sempre verificades per aquest, es preveu fer una millora important pel que fa a les contrasenyes de manera que l'usuari estigui obligat a fer-ne servir com a mínim de 8 caràcters i que forçosament, aquesta també hagi de tenir una lletra majúscula, una minúscula i un número.

Pensant en possibles implementacions diferents de les que ja s'han fet, ha sorgit la idea de crear una nova opció la qual permeti realitzar trucades des de l'aplicació, a partir del número que s'introdueix en el camp de telèfon del responsable de la comunitat, perquè d'aquesta manera l'usuari s'hi pugui posar en contacte àgilment sense haver de recórrer a buscar i trobar el número a la mateixa agenda del telèfon.

No obstant això, com que aquestes són algunes de les funcionalitats que tinc pensades proposar al client de cara a les pròximes reunions per comprovar que realment s'ajusten a les seves necessitats actuals, aquestes encara no es contemplen ni estan incloses dins de les funcionalitats demanades al principi del projecte SUMPLAPP.

13. BIBLIOGRAFIA

Aquesta bibliografia ha estat elaborada a partir del programari de gestió de referències *Zotero* el qual és lliure i de codi obert, seguint la *7a edició d'estil APA*.

[1] *Uizard | Design Wireframes & Mockups In Minutes | Uizard*. (s.d.). Recuperat 28 abril 2022, de https://uizard.io/. (Citat a la pàg. 41)

[2] *Documentación de Firebase*. (s.d.). Firebase. Recuperat 3 juny 2022, de https://firebase.google.com/docs?hl=es-419. (Citat a la pàg. 35)

[3] CodeWithMazn (Director). (2020, juliol 14). *#6 Login and Registration Android App Tutorial Using Firebase Authentication—Reset Password*. Recuperat 11 juny 2022, de https://www.youtube.com/watch?v=w-Uv-ydX_LY. (Citat a la pàg. 49)

[4] Suragch. (2020, maig 19). *Android Alert Dialog with one, two, and three buttons* [Forum post]. Recuperat 28 juny 2022, de Stack Overflow. https://stackoverflow.com/q/43513919. (Citat a la pàg. 52)

[5] Foxandroid (Director). (2021, abril 7). *How to Delete Data From Firebase Realtime Database in Android Studio || Firebase Realtime Database*. Recuperat 7 juliol 2022, de https://www.youtube.com/watch?v=L3u6T8uzT58. (Citat a la pàg. 45)

[6] *Guarda datos | Firebase Realtime Database*. (s.d.). Recuperat 9 juliol 2022, de https://firebase.google.com/docs/database/admin/save-data. (Citat a la pàg. 55)

[7] *Get Started | Geocoding API*. (s.d.). Google Developers. Recuperat 17 juny 2022, de https://developers.google.com/maps/documentation/geocoding/start. (Citat a la pàg. 37)

[8] Shunan. (2016, desembre 28). *How to get current year in android?* [Forum post]. Recuperat 26 juliol 2022, Stack Overflow. https://stackoverflow.com/q/41356279. (Citat a la pàg. 88)

[9] SundeepK. (2022). *CompactCalendarView* [Java]. https://github.com/SundeepK/CompactCalendarView/blob/e74e0eeb913744bdcaea6e9df53268441f9dbf8d/README.md (Original work published 2015). (Citat a la pàg. 37)

[10] Victor Wooding (Director). (2016, octubre 18). *Android Studio Calendar to Highlight Events No API Required*. Recuperat 28 juliol 2022, de https://www.youtube.com/watch?v=xs5406vApTo. (Citat a la pàg. 58).

[11] Edgar, S. (2019, abril 29). *Duplicate class android.support.v4.app.INotificationSideChannel found in modules classes?* [Forum post]. Recuperat 28 juliol 2022, Stack Overflow. https://stackoverflow.com/q/55909804. (Citat a la pàg. 37)

[12] DAVE, D. (2021, novembre 4). *Opening download directory using Intent* [Forum post]. Recuperat 28 juliol 2022, Stack Overflow. https://stackoverflow.com/q/69834219. (Citat a la pàg. 43)

[13] Cambo Tutorial (Director). (2022). *How to Create Float Notification in Android Studio | Float Notification Android 2022 #android*. Recuperat 1 agost 2022, https://www.youtube.com/watch?v=v1s36wmqP8M. (Citat a la pàg. 37)

[14] Iconfinder. (s.d.). *6,700,000+ free and premium vector icons, illustrations and 3D illustrations*. Iconfinder. Recuperat 3 agost 2022, de https://www.iconfinder.com/. (Citat a la pàg. 30)

[15] Programmer World (Director). (2020, novembre 14). *How to create an Excel file from your Android App?* Recuperat 28 juliol 2022, https://www.youtube.com/watch?v=MSgJJzhSI-A. (Citat a la pàg. 83)

[16] *ANDROID STUDIO: CURSO COMPLETO DESDE CERO - YouTube*. (s.d.). Recuperat 15 juny 2022, https://www.youtube.com/playlist?list=PLI4dAv2GvnrRiN3M6KUUI1M59rsI4h3wl. (Citat a la pàg. 35)

[17] SmallAcademy (Director). (2019, octubre 18). *Read and Display Excel Data in RecyclerView | Android Studio Tutorials*. Recuperat 19 juny 2022, https://www.youtube.com/watch?v=kxdVo4RH3nE. (Citat a la pàg. 37)

[18] Coding With Tea (Director). (2021, febrer 25). *Geocoding an Address to its coordinates—Google Maps API*. Recuperat 21 juny 2022, https://www.youtube.com/watch?v=AVdtzVGmTE4. (Citat a la pàg. 82)

[19] Android Tutorial. (2021, febrer 23). *GeeksforGeeks*. https://www.geeksforgeeks.org/android-tutorial/. (Citat a la pàg. 22)

[20] Sgoliver, P. (2013, gener 25). *Interfaz de usuario en Android: Fragments*. sgoliver.net. https://www.sgoliver.net/blog/fragments-en-android/. (Citat a la pàg. 25)

[21] *Android—Activities*. (s.d.). Recuperat 17 agost 2022, de https://www.tutorialspoint.com/android/android_acitivities.htm. (Citat a la pàg. 25)

[22] Tobias. (2018, desembre 4). *What is the difference between min SDK version/target SDK version vs. Compile SDK version?* [Forum post]. Recuperat 17 agost 2022, de Stack Overflow. https://stackoverflow.com/q/24510219. (Citat a la pàg. 38)

[23] *Componentes en Android*. (s.d.). Recuperat 17 agost 2022, de https://desarrolloweb.com/articulos/6-componentes-basicos-android.html. (Citat a la pàg. 26)

[Desarrolladores de Android]: Introducción a Android Studio | Desarrolladores de Android. (s.d.). Android Developers. Recuperat 25 agost 2022, de https://developer.android.com/studio/intro?hl=es-419. (Citat a la pàg. 24)

[TutorialsPoint]: Android—Fragments. (s.d.). Recuperat 17 agost 2022, de https://www.tutorialspoint.com/android/android_fragments.htm. (Citat a la pàg. 25)

[Google Developers]: Google Maps Platform Documentation | Geocoding API. (s.d.). Google Developers. Recuperat 22 agost 2022, de https://developers.google.com/maps/documentation/geocoding. (Citat a la pàg. 26)

[Desarrolladores Android]: Descripción general de los avisos | Desarrolladores de Android. (s.d.). Android Developers. Recuperat 24 agost 2022, de https://developer.android.com/guide/topics/ui/notifiers/toasts?hl=es-419. (Citat a la pàg. 27)

[Code Envato Tuts]: Android Desde Cero: Comprendiendo Vistas y Grupo de Vistas. (s.d.). Code Envato Tuts+. Recuperat 23 agost 2022, de https://code.tutsplus.com/es/tutorials/android-from-scratch-understanding-views-and-view-groups--cms-26043. (Citat a la pàg. 29)

[Desarrolladores de Android 2]: Cómo agregar gráficos vectoriales de varias densidades | Desarrolladores de Android. (s.d.). Android Developers. Recuperat 23 agost 2022, de https://developer.android.com/studio/write/vector-asset-studio?hl=es-419. (Citat a la pàg. 30)

[Desarrolladores de Android 3]: Cómo agregar gráficos vectoriales de varias densidades | Desarrolladores de Android. (s.d.). Android Developers. Recuperat 23 agost 2022, de https://developer.android.com/studio/write/vector-asset-studio?hl=es-419. (Citat a la pàg. 31)

[Firebase]: Información básica | Documentación de Firebase. (s.d.). Firebase. Recuperat 17 agost 2022, de https://firebase.google.com/docs/guides?hl=es-419. (Citat a la pàg. 35)