Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

MOVESHARE: APLICACIÓ ANDROID PER COMPARTIR DADES MENTRE ES CONDUEIX

Gabriel Rodriguez Chafino

Director: Sergio Sayago

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 27 de juny de 2018

Resum

Avui en dia la població, amb l'auge dels *smartphones*, necessita estar connectada constantment i segons quines situacions es genera una preocupació per manca de comunicació entre usuaris.

Un moment on podem dir que la comunicació pot ser reduïda és en la conducció, ja que un conductor ha de focalitzar-se en realitzar aquesta tasca. En conseqüència, altres usuaris externs a l'entorn del conductor poden alterar la seva concentració i generar una situació de perill sense que aquests siguin conscients.

Per aquesta raó sorgeix la necessitat de desenvolupar una aplicació basada a compartir dades entre un conductor i una o vàries persones externes al vehicle i, com a segon propòsit, visualitzar les dades d'un usuari que està conduint. D'aquesta manera, un usuari extern al vehicle podrà saber primerament que està conduint i, en segon lloc, el seu estat actual a partir de les dades compartides. Això pot fer reflexionar al visualitzador per no realitzar segons quines accions (per exemple, una trucada) a una persona que està centrant les seves facultats físiques i mentals en conduir.

Finalment, es desitja, a partir d'aquest treball, realitzar una aplicació Android que pugui fer el funcionament comentat en temps real sense necessitat d'utilitzar el dispositiu mòbil per part del conductor i situar al visualitzador en el context actual d'aquest.

Resumen

Hoy en día, en la población, con el auge de los smartphones, hay una necesidad de estar conectados constantemente. Dependiendo de la situación, si no hay comunicación directa se genera una preocupación entre usuarios.

Un momento donde podemos decir que la comunicación puede ser reducida es en la conducción, ya que un conductor tiene que centrarse en realizar esta tarea. En consecuencia, otros usuarios externos al entorno del conductor pueden alterar su concentración y generar una situación de peligro sin que estos sean conscientes.

Por esta razón surge la necesidad de desarrollar una aplicación basada en compartir datos entre un conductor y una o varias personas externas al vehículo y, como segundo propósito, visualizar los datos de un usuario que está conduciendo. De este modo, un usuario externo al vehículo podrá saber primeramente que está conduciendo y, en segundo lugar, su estado actual a partir de los datos compartidos. Esto puede hacer reflexionar al visualizador para no realizar según qué acciones (por ejemplo, una llamada) a una persona que está centrando sus facultades físicas y mentales en conducir.

Finalmente, se desea, a partir de este trabajo, realizar una aplicación Android que pueda realizar el funcionamiento comentado en tiempo real sin necesidad de utilizar el dispositivo móvil por parte del conductor y situar al visualizador en el contexto actual de este.

Abstract

Needless to say we feel and need to be connected with other users. However, depending on the situation, this communication might not always be possible and be dangerous too. An example is driving. A moment where we can say that communication can be reduced is in driving, since a driver should focus on performing this task. Consequently, other users outside the driver's environment can alter their concentration and generate a dangerous situation without them being aware of.

For this reason, there is a need to develop an application based on sharing data between a driver and one or more people outside the vehicle and, as a second purpose, visualize the data of a user who is driving. In this way, a user external to the vehicle will be able to know first what he is driving and, secondly, his status from the shared data. This can cause the viewer to think about not doing what actions (for example, a call) to a person who is focusing their physical and mental faculties on driving.

Finally, it is desired, from this work, to make an Android application that can perform the commented operation in real time without using the mobile device by the driver and place the viewer in the current context of it.

Voldria agrair a tots els companys del grau, que s'han convertit en amics, que han fet que aquesta etapa sigui molt especial. També agrair al tutor d'aquest treball i la meva parella l'ajuda proporcionada en tot moment.

Gràcies a tothom.

Sumari

1. Introducció

Les noves tecnologies d'aquest segle XXI han originat una necessitat a la societat, la necessitat d'estar connectats en tot moment. Concretament amb l'auge dels *smartphones* (telèfons intel·ligents) aquest efecte va permetre la creació de moltes plataformes, xarxes socials, etc., per facilitar la connexió no física entre persones en tot moment.

D'altra banda, existeix un context a la nostra vida quotidiana on no és fàcil estar connectats, mentre conduïm. En aquest moment, no és fàcil estar comunicats amb altres persones externes al vehicle, ja sigui per mantenir-se concentrat a fi d'evitar distraccions, del compliment de les normes de circulació i la manca de tecnologia incorporada en el vehicle.

Per aquesta raó, l'aplicació, per a dispositius mòbils Android, que sorgirà d'aquest treball de recerca es basarà, primerament, en compartir dades entre un conductor i una o vàries persones externes al vehicle i, com a segon propòsit, visualitzar les dades d'un usuari que està conduint. Tot això en temps real. El desenvolupament d'aquest software servirà per saber si un familiar o amic està conduint i quines són les dades del seu viatge. Aquestes dades les definirem en la fase d'anàlisi.

Finalment, avui en dia els cotxes de nova fabricació incorporen tota mena de tecnologies per a considerar-los cotxes intel·ligents. La línia de desenvolupament tendeix a realitzar sistemes per permetre que els cotxes estiguin interconnectats. Per aquest motiu es vol seguir aquesta tendència amb aquest treball, però incloent una diferència, connectar usuaris que estan conduint amb d'altres que no ho estan, encara que el desenvolupament a realitzar pot ser utilitzat en un futur per comunicar conductors amb d'altres i generar una xarxa de cotxes intel·ligents interconnectats.

1.1. Motivació

La motivació principal d'aquesta aplicació neix del repte de realitzar un software que pugui aportar quelcom a tots els públics. Concretament, penso que les aplicacions d'avui dia no estan adaptades per a persones d'edat avançada, que solen tenir més dificultats per entendre el funcionament com altres persones més joves.

La gran majoria d'aplicacions estan destinades a ser utilitzades exclusivament mentre es condueix, això implica que l'usuari disposi de llicència de conducció, per aquesta raó la quota de mercat es redueix. Aquesta restricció motiva a la realització d'una aplicació on tenim dos tipus d'usuari, el conductor i el visualitzador, on el visualitzador no té l'obligatorietat de tenir la llicència de conducció. Aquesta idea de rols augmenta la quota de mercat respecte les aplicacions esmenades anteriorment.

Un altre punt és aportar tranquil·litat a aquells familiars i/o amics que estan pendents de saber si el seu familiar i/o amic ha arribat correctament a la seva destinació amb el seu vehicle. Especialment a aquells fills on veuen el deteriorament de la conducció del seu pare, aquells pares que viuen els primers mesos de conducció del seu fill.

En tercer lloc, al nostre país un 30% de les víctimes mortals d'accidents de trànsit (El mundo, 2018) son degut a les distraccions al volant. D'aquest percentatge un 94% es causat per l'ús del mòbil. Per aquesta raó, sorgeix la motivació d'aportar una aplicació que fomenti la utilització del mòbil sense generar distraccions a fi d'evitar accidents.

Finalment, la conducció pot generar estrès i fatiga en els conductors segons la situació i rebre una trucada en aquest moment pot ser positiu. En situacions de fatiga, rebre una trucada genera en el conductor un increment del nivell d'excitació i evitar una distracció. Per altra banda, en moments d'estres es poden produir quan un conductor està perdut i una ajuda externa pot reduir el nivell d'excitació i sortir d'aquesta situació. Per aquesta raó, sorgeix la necessitat de realitzar una aplicació diferent de les existents en el mercat per aportar informació a una persona externa que ajudarà a identificar la situació del conductor.

1.2. Objectius

L'objectiu principal d'aquest projecte és realitzar una aplicació per a dispositius Android per a compartir una sèrie de dades d'un usuari que està conduint amb un altre que no ho està. Finalment haurem d'obtenir un software funcional, de qualitat i escalable per seguir desenvolupant noves millores i funcionalitats a la finalització del treball fi de grau.

Per a poder aconseguir el nostre objectiu principal hem d'anar superant de secundaris. Un altre, és realitzar una aplicació totalment adaptada per als usuaris de dispositiu mòbil Android, de fàcil comprensió. Per això necessitarem analitzar dels requeriments, identificant que volem fer i com ho volem fer per satisfer les necessitats dels futurs usuaris. És important adquirir coneixements de l'àmbit tractat en l'aplicació llegint diferents articles d'investigació. A més a més, hem d'analitzar les aplicacions similars que hi ha al mercat actualment i extreure els seus punts forts i febles per a poder definir una interfície que maximitzi l'experiència d'usuari.

D'altra banda, hem de donar llibertat i privacitat a l'usuari per poder compartir les dades que ell vulgui amb els grups d'usuaris que desitgi. En conseqüència, els usuaris visualitzadors només tindran accés a les dades que li han permès visualitzar.

Amb motiu que l'usuari que comparteix dades està conduint, també necessitarem un sistema automàtic que comparteixi aquelles dades que poden canviar amb el temps, com per exemple la ubicació. De forma periòdica s'enviarà aquesta dada als usuaris autoritzats, on aquesta periodicitat podrà ser imposada per l'usuari.

Finalment, com a últim objectiu hem d'aplicar els coneixements apresos durant el grau, especialment de programació, disseny de software, enginyeria del software i factors humans.

1.3.Escenaris d'ús

Per mostrar el context de l'aplicació, a continuació es presenten dos escenaris d'us on s'imaginen dos tipus d'usuaris amb característiques molt diferents.

1.3.1. Primer escenari

En Josep un jubilat de 85 anys viu sol a Barcelona, però els caps de setmana visita al seu fill de Vilassar de Dalt. Per anar a visitar-lo utilitza el seu cotxe particular. El seu fill preocupat per l'edat avançada i el seu estil de conducció deteriorat, pacta amb ell la utilització d'una aplicació que permet saber quan el seu pare ha sortit amb el cotxe de Barcelona cap a Vilassar de Dalt. El fill mitjançant l'aplicació esta tranquil de la monitorització de la conducció del seu pare. Al final del cap de setmana, quan en Josep torna a Barcelona, l'aplicació permet saber al seu fill si ha arribat correctament a la destinació.

1.3.2. Segon escenari

La Lidia una noia de 20 anys, li han atorgat el permís de conduir fa una setmana. Fins ara ha anat a la Universitat amb transport públic, però té l'opció d'agafar el cotxe de la seva mare per anar-hi cada dia. Com la Universitat esta en el centre de la ciutat de Barcelona, la Lidia no es sent segura, ni tampoc la seva mare, viatjant cada dia sola ja que no té experiència en aquest tipus de conducció. És per això que pacta amb la seva mare la monitorització a través de l'aplicació. L'aplicació permetrà a la Lidia a sentir-se més segura al volant, sap que si té qualsevol contratemps la seva mare podrà saber-ho. A més a més, la seva mare podrà comprovar si arriba correctament la seva filla a la Universitat.

2. Desenvolupament

2.1.Planificació inicial

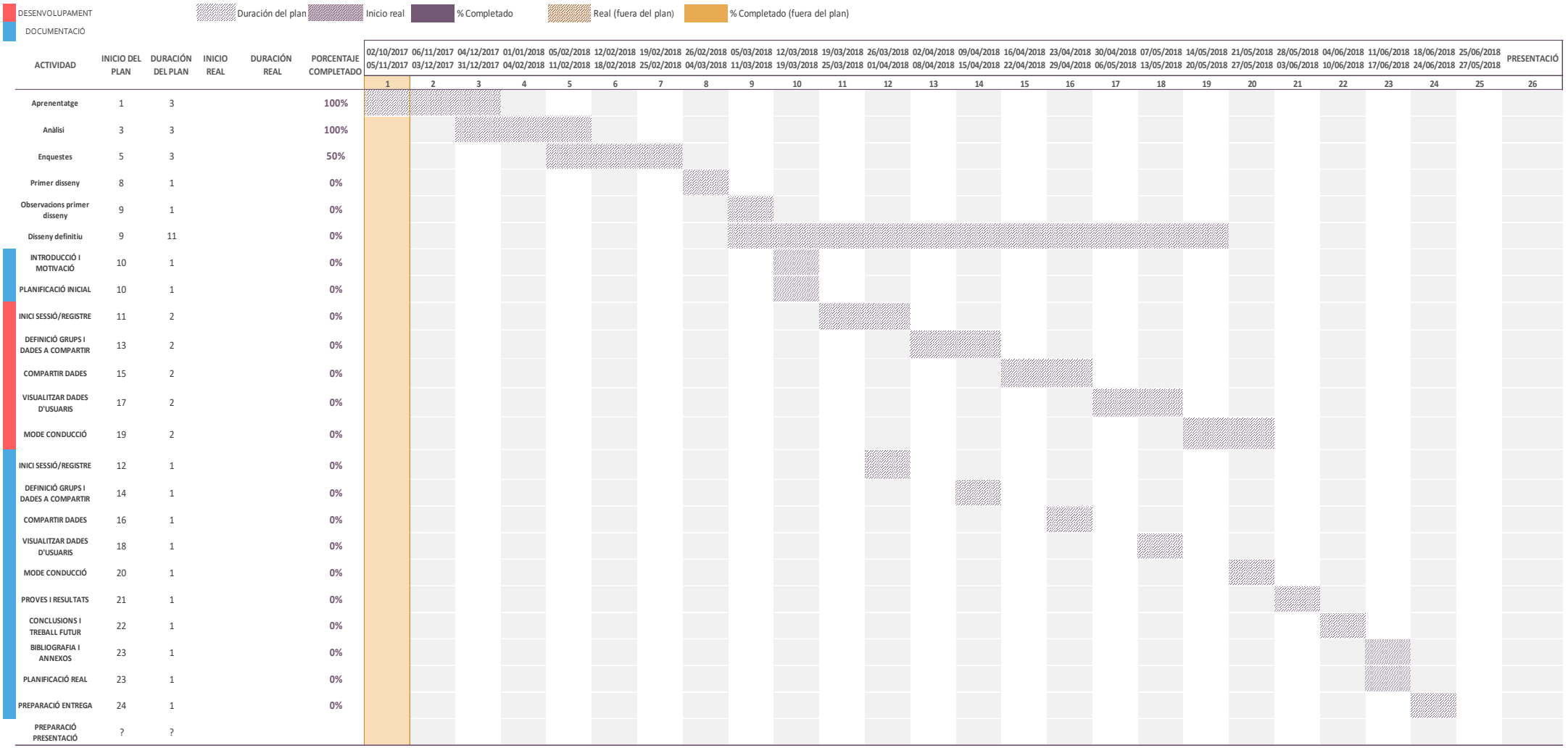

Figura 1: Planificació inicial.

Per a dur a terme aquest desenvolupament, ens hem marcat aquesta planificació. Es pot observar la dedicació inicial d'aprenentatge i anàlisi on ja ha estat realitzada en el semestre anterior a l'entrega del treball.

Primerament realitzarem un estudi per a poder afinar els requeriments de l'aplicació i també la interfície mitjançant enquestes i observacions. El disseny definitiu de l'aplicació es desenvoluparà fins a acabar les funcionalitats.

Finalment realitzarem una comparació entre la planificació inicial i real, prepararem l'entrega i la defensa del Treball Fi de Grau.

2.1.1. Metodologia

Per a realitzar els desenvolupaments utilitzarem un model de desenvolupament iteratiu incremental. Tal com ens diu aquest model realitzarem blocs de funcionalitats bàsiques en cada increment. Cada un d'aquest increment es dividirà sempre en les mateixes etapes:

1. **Anàlisi**: en aquesta primera fase es pensaran els requeriments funcionals de l'increment.
2. **Disseny de la interfície**: realitzarem el disseny abans de començar a desenvolupar les funcionalitats.
3. **Implementació**: després de l'anàlisi i el disseny ens posarem a desenvolupar les funcionalitats que ens hem marcat per aquest increment.
4. **Proves**: Un cop hem acabat el desenvolupament, donarem per acabat l'increment quan realitzem les proves adients i en cas de no passar les proves es tornarà al punt 3.

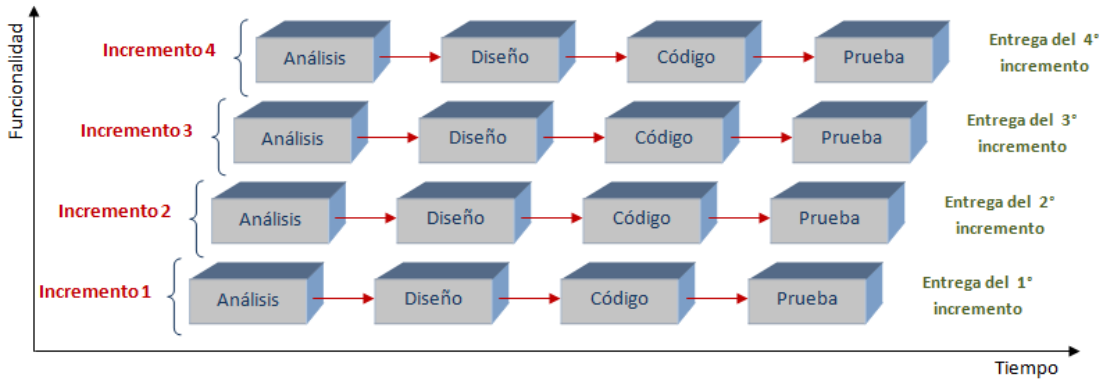

Figura 2: gràfic on representa la metodologia iterativa.

2.2.Planificació final

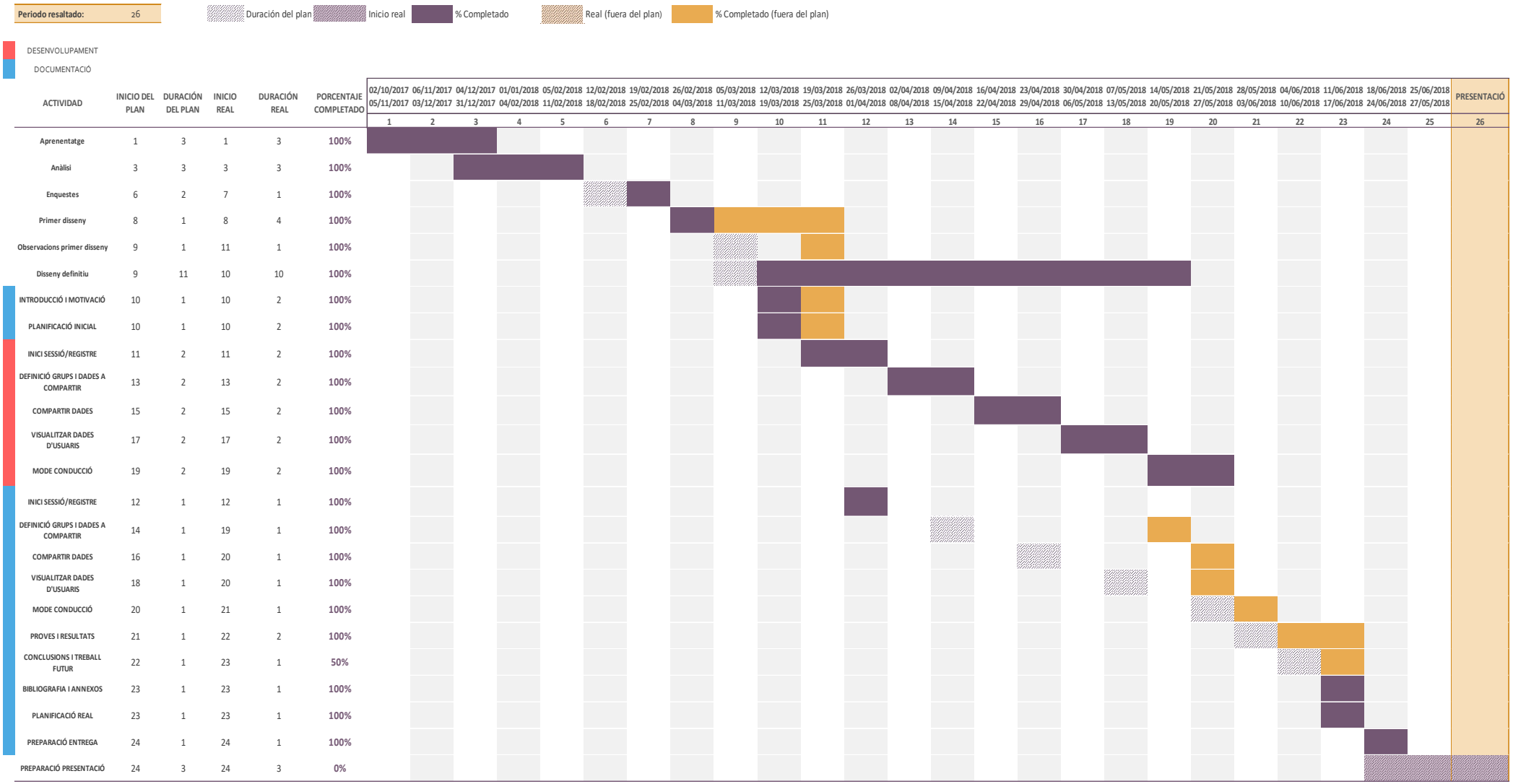

Figura 3: planificació final.

2.2.1. Diferencies respecte la planificació inicial

La planificació, dividida en períodes, que es va marcar inicialment hi ha una gran variació del que ha sigut finalment, però totes les etapes s'han superat finalment i en el temps marcat.

Per començar, podem veure que els 5 primers períodes, els mesos d'Octubre fins al Febrer, s'han complert degudament. Un cop acabada la fase d'anàlisi es va estimar en 2 períodes (dues setmanes) la realització d'enquestes, però finalment només va caler una setmana a causa de la ràpida difusió i participació que van provocar una recopilació de dades suficient.

Un cop es tenien les dades dels usuaris, es va procedir a realitzar el primer disseny però a causa d'una mala estimació del temps per aquesta tasca es va prolongar 3 períodes (indicat en groc a la Figura 3) més del que teníem pensat. Com les observacions tenien una dependència directa amb el primer disseny no es van poder realitzar fins acabar-lo. També va causar que la documentació trigués un període més del compte.

Després d'aquest primer contratemps a la nostra planificació, es va començar el desenvolupament on totes les fases van ser realitzades en el seu temps estimat. D'altra banda, vam planificar la documentació, de cada fase de desenvolupament, en l'última setmana de l'increment, però només es va poder respectar pel primer increment. A causa de la complexitat de la tecnologia, havíem d'invertir més temps per cercar documentació i proves. La resta de documentació es va veure afectada, però es va recuperar en els períodes 19, 20 i 21.

Finalment, a causa d'aquest últim contratemps la resta de documentació es va posposar fins a acabar la documentació tècnica però es va realitzar un sobreesforç (sobretot al període 23) per a posar-nos al dia amb la documentació final. Després d'entrega la documentació d'aquest treball es té planificat preparar la defensa amb una estimació de 3 períodes.

2.3.Tecnologies utilitzades

En primer lloc, l'aplicació s'implementarà per al Sistema Operatiu Android, amb una quota de mercat del 80% és el més utilitzat del món l'any 2017 (Android, 2018). El llenguatge el qual ens permet realitzar aplicacions Android és Java. Com a complement per a la programació Google ens proporciona Android Studio per desenvolupar aplicacions.

Per a poder fer realitzar l'aplicació utilitzarem l'eina Firebase, desenvolupada per Google, que és una plataforma de desenvolupament ens aporta una base de dades en temps real, gestió d'usuaris, anàlisi i moltes altres funcionalitats de forma gratuïta.

D'altra banda, per als elements visuals ha calgut utilitzar l'eina d'Adobe anomenada Adobe Photoshop.

Un altre aspecte molt important és l'emmagatzematge de codi font. Sincronitzarem el nostre codi font amb el software de control de versió Git, ens aportarà eficiència i manteniment del codi mitjançant l'allotjament gratuït i públic del portal GitHub.

Figura 4: Logo d'Android

Figura 5: Logo de Firebase.

Figura 6: Logo d'Adobe Photoshop.

Figura 7: Logo de GitHub

2.4. Pressupost

	Setmanes	Hores/setmana	Hores	Preu/hora	Preu
Aprenentatge	3	15	45	5,00 €	225,00 €
Anàlisi	5	15	75	5,00 €	375,00 €
Desenvolupament					
Increment 1	2	25	50	5,00 €	250,00 €
Increment 2	2	25	50	5,00 €	250,00 €
Increment 3	2	25	50	5,00 €	250,00 €
Increment 4	2	25	50	5,00 €	250,00 €
Increment 5	2	25	50	5,00 €	250,00 €
Documentació	10	15	150	5,00 €	750,00 €
Equipament					600,00 €
Proves					200,00 €
Publicació					21,23 €
Entorn de desenvolupament					0,00 €
TOTAL	**28**	**21,25**	**520**		**3.421,23 €**

Figura 8: pressupost segons les hores dedicades.

El pressupost que es pot veure a la Figura 8 representa el cost per a desenvolupar l'aplicació segons la planificació inicial. El nombre de setmanes per apartat ho determina la planificació i el nombre d'hores és una aproximació basada en la informació general del Treball Fi de Grau. D'altra banda, el preu per hora es basa en el preu mínim que estipula la Facultat de Matemàtiques (Pràctiques en empreses, 2018) a l'hora de realitzar pràctiques en empresa. D'altra banda, hauríem d'incloure al preu del pressupost els 25$ (21,23€) (Google Play, 2018) per publicar una aplicació a la tenda d'aplicacions Android.

En segon lloc, per a poder desenvolupar l'aplicació caldrà tindre un equip suficientment potent per suportar els requeriments del programa Android Studio (totalment gratuït) i els emuladors de dispositius Android (Android Studio, 2018). Aproximadament un equip amb aquestes característiques el taxem en 600€ basat en un equip d'una tenda online d'equips informàtics.

En tercer lloc, com l'aplicació està destinada, en part, a ser utilitzada a la conducció necessitem fer proves en l'entorn de conducció i implica una despesa. Per a la realització de les proves disposem de 200 € exclusivament per a combustible.

Finalment, cal dir que tots els preus que es poden veure a la Figura 8 inclouen IVA.

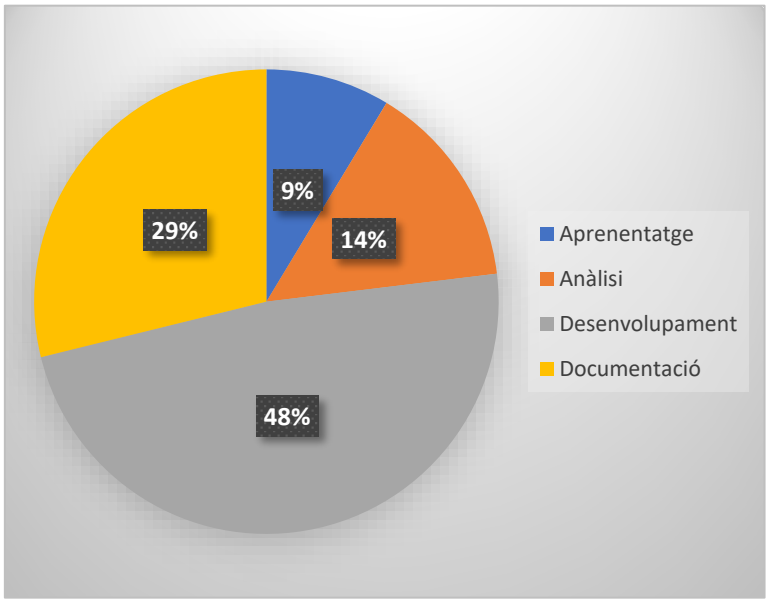

Figura 9: dedicació de cada fase del Treball Fi de Grau.

2.4.1. Manteniment de l'aplicació

Tota aplicació necessita un manteniment a causa dels costos d'allotjament de l'aplicació, bases de dades, etc. En el nostre cas, utilitzarem Firebase que ens aporta els serveis necessaris per mantenir una aplicació totalment gratuïta. Cal dir que el manteniment gratuït tenim una sèrie de restriccions marcades pel "*Pla Spark*". (Firebase, 2018)

Un cop sobrepassats els límits hauríem d'invertir 25 $ cada mes i en cas de tornar a superar els límits, haurem de contractar el pagament per ús.

2.5.Model de domini

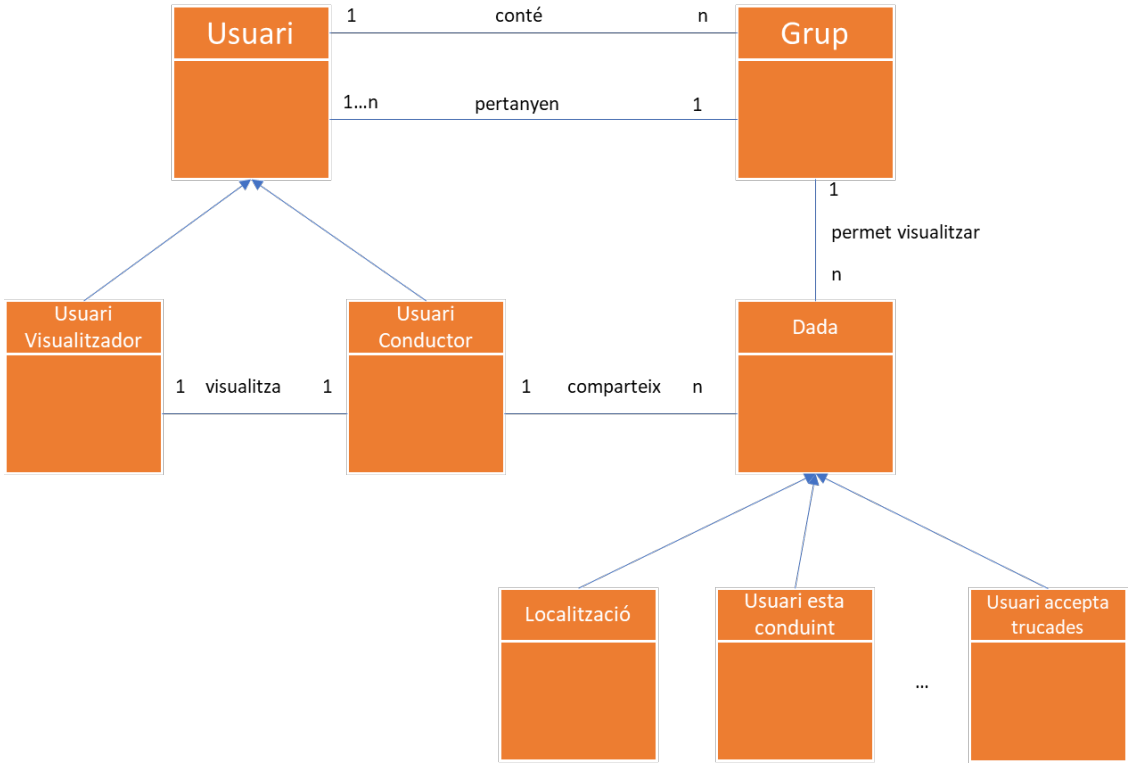

Figura 10: Model de domini de l'aplicació

Es pot observar que tenim dos conceptes d'usuari a l'aplicació. El visualitzador no existeix fins que un usuari conductor li permeti visualitzar dades. Les dades a compartir estan definides a un grup creat prèviament per l'usuari i consultades per un usuari visualitzador on només podrà veure aquelles dades que l'usuari conductor li autoritzi.

3. Anàlisi

En aquesta fase del desenvolupament partirem de les idees que hem extret després de llegir documentació i articles sobre l'àmbit de l'automoció i les interfícies (Andrew L. Kun, 2013) (Bastian Pfleging, 2013).

3.1. Aplicacions similars

A continuació s'exploraran aplicacions existents per a poder recopilar aspectes per a l'aplicació d'aquest projecte. Per a poder explorar aplicacions existents si han consultat les aplicacions existents en la tenda oficial d'Android *Google Play Store* i a persones que coneixen o han utilitzat qualsevol aplicació d'aquest àmbit.

Com a primera impressió, no he trobat aplicacions basades a compartir dades de forma massiva mentre es condueix, la majoria d'aplicacions es basen a transmetre la seva ubicació juntament amb la navegació. A continuació es presenten aplicacions similars.

- ***Google Latitude*** (Google Latitude, 2018) : va ser un servei basat en compartir la localització mòbil desenvolupat per Google. El seu us no es exclusiu només quan es condueix, però permet compartir la ubicació d'un usuari a un o varis usuaris si es desitja. El servei de l'aplicació es va suspendre al 2013 però va ser incorporat a Google+ en la secció d'ubicacions. La funcionalitat principal era compartir l'última localització registrada (s'actualitzava cada cert temps en segon pla) a contactes autoritzats.

- ***Gas biker*** (Gas biker, 2018): aquesta aplicació és la més pròxima al que volem realitzar en aquest projecte. És una aplicació disponible per Android i iOS destinada a motoristes, especialment quan realitzen un trajecte llarg, basada en compartir la localització actual, temps de trajecte, kilòmetres recorreguts i trajecte realitzat al grup (composat per altres usuaris de l'aplicació) creat prèviament. També incorpora un sistema d'alerta en cas de patir un accident o contratemps. Aquesta alerta pot ser activada de forma manual i/o automàtica on avisa als serveis d'emergències, centre d'alertes de *Gas Biker* i a tots els membres del grup de la situació del motorista juntament amb totes les dades recopilades. També permet adjuntar un missatge preestablert a l'alerta.

- ***Android Auto*** (Android Auto): aplicació desenvolupada per Google amb la finalitat de realitzar tota mena d'accions amb el telèfon mòbil mitjançant interaccions verbals amb aquest. La funcionalitat es simple, Android Auto presenta una interfície ocupant tota la pantalla mostrant molt pocs botons i textos amb tipografia de gran dimensions per minimitzar l'atenció del conductor sobre el dispositius.

Aquestes tres aplicacions ens aporten gran quantitat de dades que podem combinar i aprofitar per realitzar el projecte. Les dues primeres aplicacions similars podem extreure idees per les funcionalitats i, d'altra banda, podem agafar idees d'*Android Auto* per a la interfície.

Finalment, el que diferenciarà la nostra aplicació de la resta d'aplicacions semblants serà poder compartir altres dades (explicades a la secció 3.2), i no solament la localització, que poden resultar interessants per a aquells usuaris triats pel conductor prèviament. Les aplicacions anteriors se centren a compartir la localització, però la nostra futura aplicació té la intenció d'anar més enllà i fer sentir la sensació al visualitzador d'estar present en el context del conductor.

3.2. Dades a compartir

Primerament, hem de definir quines dades es volen compartir en l'aplicació. Com a primera opció es proposen les següents:

- **Localització**
  Dada més important, d'aquesta es poden extreure molta informació per al visualitzador. Pot saber en tot moment si el conductor està desplaçant-se i que no ha patit cap accident. També es pot extreure en quina via se situa, l'estat d'aquesta, meteorologia, etc.
  - Dades derivades:
    1. Tràfic
    2. Meteorologia
    3. Via actual
    4. Estat de la via
    5. Velocitat
    6. Kilòmetres recorreguts

- **Conduint / No conduint**
  Amb aquesta dada, el conductor informa als visualitzadors quan està conduint. Per a certs col·lectius com gent gran (o conductors novells), pot evitar que un familiar (o amic) el molesti en aquell moment (moltes trucades acaben quan l'emissor de la trucada sap que ha trucat a una persona que està conduint), ja que pot induir a distraccions.

- **Accepta trucades**
  Similar a la dada anterior, aquesta pot permetre als conductors a preparar-se per a rebre trucada (amb un mans lliures). Per al visualitzador és bastant útil perquè sap si el conductor està preparat o no per rebre trucades.

- **Hora d'inici de la conducció (amb això es pot aconseguir el temps de conducció actual)**
  Amb l'hora d'inici, acompanyada de la destinació, el visualitzador pot saber quant de temps porta de conducció. Per a conductors d'edat avançada, el visualitzador pot intuir si s'ha perdut, ja que si està trigant més del compte a dirigir-se a una destinació.

- **Destinació**
  La destinació pot saber per al visualitzador on es dirigeix el conductor i juntament amb la localització pot saber si li queda molt viatge.

- **Buscant aparcament**
  Amb aquesta dada, acompanyada de l'última localització registrada, es pot saber on ha aparcat el conductor, si és una persona gran, el visualitzador pot saber on ha aparcat i comunicar-li en cas d'oblidar-ho. Independentment de l'edat el visualitzador pot saber que el conductor ja ha arribat a la seva destinació i només li queda buscar aparcament.

- **Música que s'està reproduint**
  Pot ser interessant i enriquir l'experiència del visualitzador, de moment no trobem quelcom que destaqui d'aquesta dada, en la fase d'anàlisi d'enquestes veiem que més pot aportar als usuaris.

- **Nombre de passatgers**
  Pot ser interessant saber el nombre de passatgers. Permet conèixer el nombre d'acompanyants del conductor. En cas de voler comunicar quelcom al conductor, si el

visualitzador coneix als acompanyants, es pot comunicar directament amb ells per no distreure al conductor.

- **Imatges**
  Segons un article (Bastian Pfleging, 2013) el conductor no són partidaris de compartir moltes imatges o vídeos. Però si han de compartir-les prefereixen enviar imatges frontals a la conducció, mostrant la carretera. Aquesta dada podria situar al visualitzador en el context del conductor. Les imatges es podrien configurar per ser realitzades cada cert temps configurat per l'usuari.

3.3. Accions de l'aplicació

A l'hora de triar les accions hem de pensar que tenim dos contexts totalment diferents. Primer trobem a un usuari realitzant una tasca important com és la conducció, on no pot rebre cap distracció per part de l'aplicació.

A continuació es presenten unes accions per a ser realitzades mentre es condueix i a través de comandes de veu per no posar en perill la integritat del conductor:

1. Realitzar trucades
2. Enviar missatges
3. Escoltar notificacions entrants

3.4. Enquestes

Un cop proposat els requeriments de l'aplicació, realitzarem un estudi per poder conèixer l'opinió del futur públic de la nostra aplicació, això ens servirà per comprovar si hi ha gent interessada a compartir dades d'aquesta manera. Per dur a terme aquest estudi realitzarem una enquesta en línia en dos idiomes (català i castellà) a través de *Google Forms* per aconseguir un abast de públic important.

3.4.1. Preguntes i motivacions

Encara que l'enquesta original es pot visualitzar en l'apartat d'annexos, a continuació es presenta una taula que conté les preguntes juntament amb la motivació de cadascuna.

SECCIÓ	PREGUNTA	MOTIVACIÓ
1	Edat	Informació de l'usuari.
	Tens permís de conduir?	Ens permet saber si tindre més en compte el rol de conductor o visualitzador.
2	Has utilitzat algun cop el telèfon mentre conduïes?	En el cas que l'utilitzi pot ser un usuari adequat per a la nostra aplicació. En cas que no utilitzi el telèfon la nostra aplicació pot ser-li molt útil ja que no el posarà en perill ni incomplirà les normes de circulació, però potser es més escèptic a utilitzar-la.
	T'agradaria informar a familiars/amics propers de dades mentre estàs conduint? (sense posar-te en perill i respectant les normes de conducció)	Coneixerem l'interès de l'usuari en compartir dades mentre condueix.
	Que prefereixes? 1. Establir que compartir i amb qui 2. Compartir totes les dades amb usuaris determinats 3. Cap	Sabrem quina forma de compartir dades es la preferida per l'usuari.
	Compartir: l'usuari esta conduint	Obtindrem unes dades valuoses per saber si les dades que proposem per compartir son adequades o no.
	Compartir: localització	
	Compartir: l'usuari accepta trucades	
	Compartir: hora d'inici de conducció	
	Compartir: destinació	
	Compartir: música	
	Compartir: nombre de passatgers	
	Compartir: imatges frontals	
	Altres dades d'interès	Els usuaris poden proposar altres dades que potser no hem pensat.
	Quines de les següents accions incorporaries a l'aplicació encara que es podrien realitzar amb un mans lliures?	Aquestes accions es poden incorporar per a que l'usuari les utilitzi mentre condueix, coneixerem si els usuaris ho volen encara que ho poden utilitzar amb els mans lliures.
	Altres accions d'interès	Els usuaris poden proposar altres accions que potser no hem pensat.
3	Acceptaries conèixer dades d'un familiar/amic sabent que està conduint? (sense posar-te en perill i respectant les normes de conducció)	Coneixerem si els usuaris que no estan conduint tenen interès visualitzar les dades d'altres usuaris que estan conduint.
	Trucaries a un usuari si saps que està conduint?	Moltes trucades de telèfon acaben quan l'emissor (no esta conduint) coneix que el receptor esta conduint. Amb aquesta pregunta poden saber si l'aplicació acabarà amb aquest costum.
	Coneixes a un familiar d'edat avançada i a cada cop té més dificultats per conduir de manera perfecte. Quines dades t'agradaria conèixer?	Tornem a demanar el mateix que a la secció 2 però des de un altre punt de vista i d'una altra manera. Podrem conèixer si hi ha diferencies entre rols.
	Altres dades d'interès	Els usuaris poden proposar altres dades que potser no hem pensat.

3.4.2. Resultats de l'enquesta

Un total de 203 persones han contestat l'enquesta, 111 l'enquesta en català i 92 en castellà. Podem dir que és una participació alta.

- **Edat**

Hem de tindre en compte que les opinions que analitzarem a continuació provenen, majoritàriament, d'usuaris entre 40 – 65 anys.

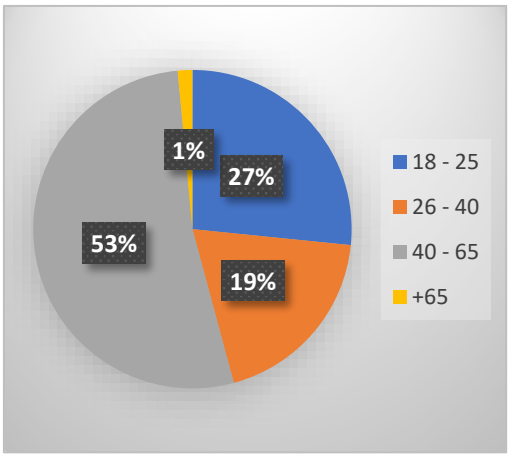

Figura 11: gràfic dels resultats de l'edat.

- **Tens permís de conduir?**

Només 18 dels 203 usuaris no tenen carnet de conduir. Podem dir que la majoria dels usuaris enquestats serien usuaris potencials, ja que utilitzarien l'aplicació tant en mode de conducció com de visualitzadors.

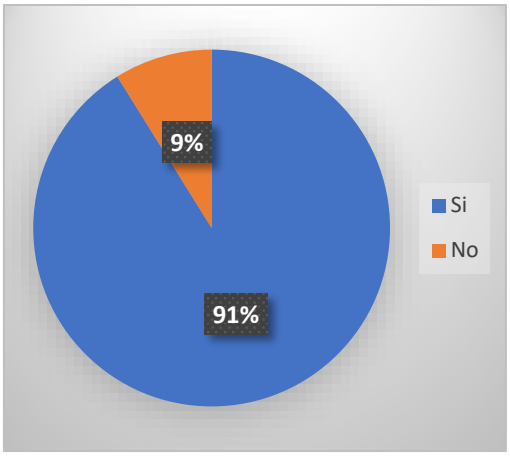

Figura 12: gràfic dels resultats de tindre carnet de conduir.

- Has utilitzat algun cop el telèfon mentre conduïes?

Gairebé 2 de cada 3 usuaris reconeixen haver utilitzat el telèfon algun cop mentre conduïa, és un índex alt per la perillositat que comporta. Amb la creació de l'aplicació un principal objectiu és rebaixar aquesta dada.

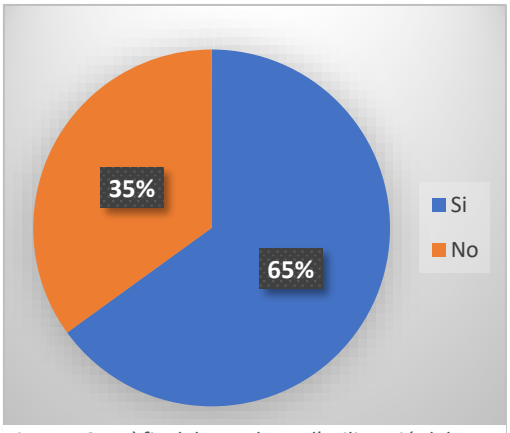

Figura 13: gràfic dels resultats d'utilització del telèfon mòbil.

- T'agradaria informar a familiars/amics propers de dades mentre estàs conduint? (sense posar-te en perill i respectant les normes de conducció)

Sembla que al 88% dels enquestats estaria d'acord en la finalitat de l'aplicació, compartir informació als seus familiars i/o amics.

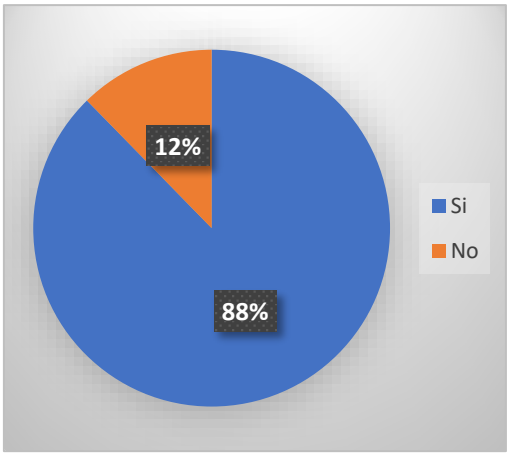

Figura 14: gràfic dels resultats d'informar a familiars/amics.

- Que prefereixes?

7 de cada 10 dels usuaris estan d'acord en compartir les dades com plantegem.

Gairebé 2 de cada 10 usuaris compartiria totes les dades amb usuaris determinats. Per complir aquesta preferència podem incorporar l'opció "seleccionar totes les dades" per satisfer les necessitats de gairebé el 90% dels usuaris enquestats.

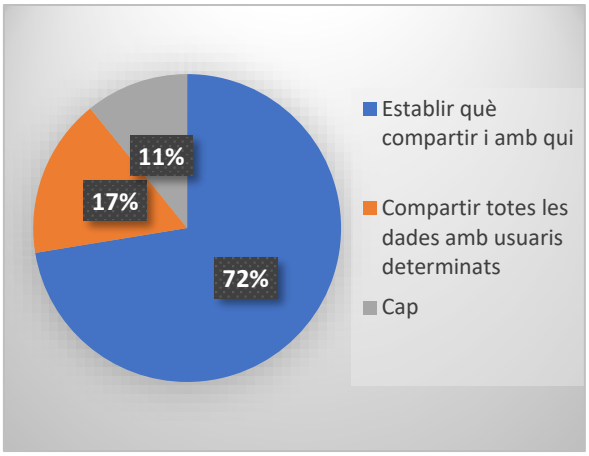

Figura 15: gràfic dels resultats de la preferència de com compartir les dades.

- Dades a compartir com a conductor

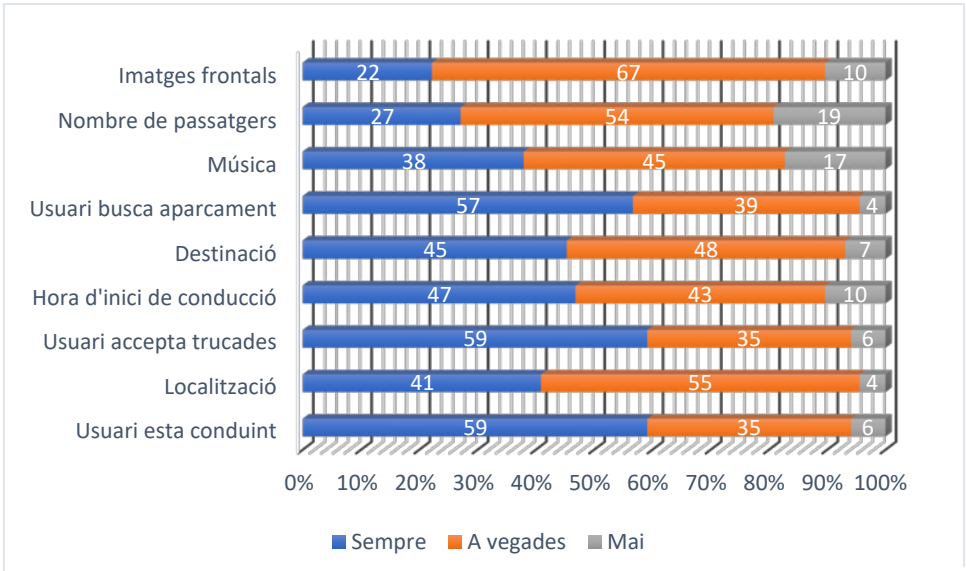

Figura 16: gràfic dels resultats de les dades a compartir

Primerament les dades més interessants pels usuaris serien les següents:

1. Usuari està conduint
2. Usuari accepta trucades
3. Usuari busca aparcament

Ja que aquestes dades han obtingut una rellevància alta, superant el 50% de compartir sempre aquesta dada, seran les úniques dades que apareixeran de forma predeterminada per a compartir, respectant sempre l'opció de deshabilitar-la.

Per altra banda, les dades que més rebuig han obtingut han sigut la música i el nombre de passatgers, per tant, no s'inclouran en l'aplicació.

Finalment, la resta de dades les considerarem a l'hora d'incloure-les a l'aplicació, ja que han obtingut un índex baix d'acceptació, però no tan alt com per habilitar-les sempre en l'aplicació.

- Quines de les següents accions incorporaries a l'aplicació encara que es podrien realitzar amb un mans lliures?

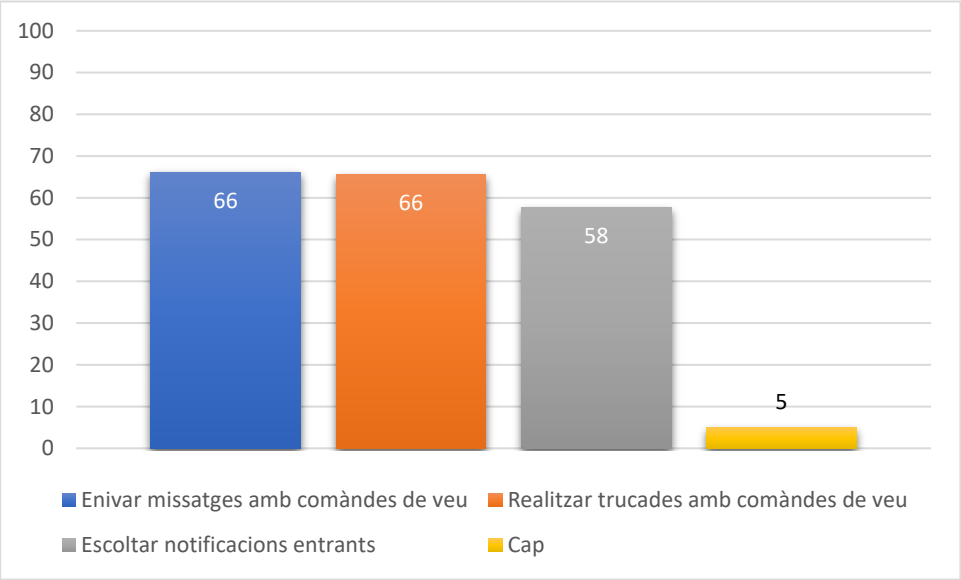

Figura 17: gràfic dels resultats de les possibles accions a incorporar.

Veiem que un 66% de les persones enquestades voldria incorporar l'enviament de missatges i la realització de trucades a l'aplicació. Escoltar els missatges també ha rebut una puntuació bona, però repensant aquesta funcionalitat podria causar estrès al conductor en cas de rebre moltes notificacions en un interval curt de temps.

- Acceptaries conèixer dades d'un familiar/amic sabent que està conduint? (sense posar-te en perill i respectant les normes de conducció)?

Semblant a la pregunta vinculada a la Figura 14, com a visualitzadors els usuaris tolerarien acceptar dades d'un conductor.

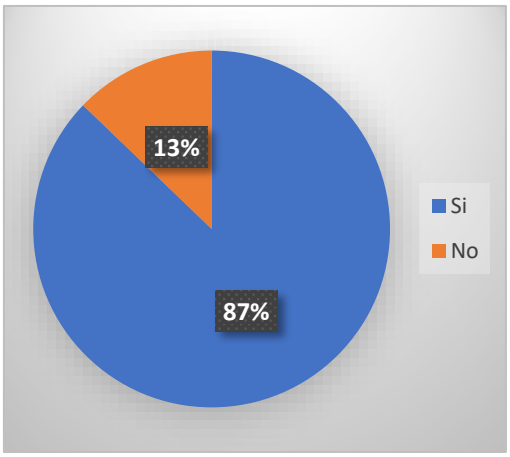

Figura 18: gràfic dels resultats d'acceptar rebre informació

- Trucaries a un usuari si saps que està conduint?

La majoria dels usuaris voldrien trucar al conductor només si és urgent. Però quan no és urgent, 2 de cada 3 persones no el trucarien. La informació d'aquesta aplicació rebaixaria les possibilitats de distracció del conductor.

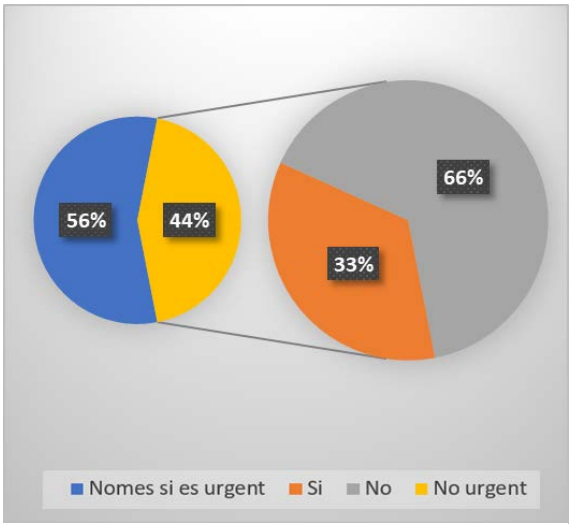

Figura 19: gràfic dels resultats de trucar a un usuari mentre condueix.

- Coneixes a un familiar d'edat avançada i a cada cop té més dificultats per conduir de manera perfecta. Quines dades t'agradaria conèixer?

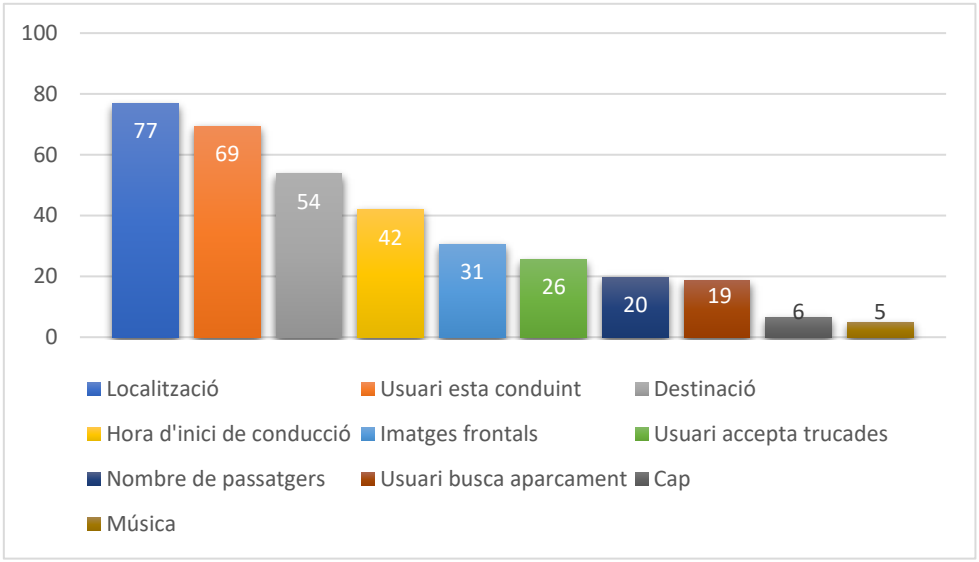

Figura 20: gràfic dels resultat de dades que li agradaria a l'usuari conèixer.

Com hem d'arribar a un punt intermedi entre les preferències com a conductor i com a visualitzador, valorarem aquest gràfic juntament amb el de la Figura 16.

En primer lloc, tant en la figura esmenada com en aquests resultats veiem que la música no genera interès en els usuaris (només ha sigut triada per un 5%), per tant, definitivament no l'inclourem en l'aplicació. D'altra banda, el nombre de passatgers no ha obtingut resultats dolents (20% dels usuaris) però tampoc tan bons com per millorar

les estadístiques de la Figura 16 (rebuig del 19%), llavors tampoc ho tindrem en compte en l'aplicació.

En segon lloc, hi ha discrepàncies entre les preferències del conductor amb les del visualitzador amb la dada d'"Usuari busca aparcament" i la "Localització". Anteriorment, havíem decidit incloure la primera dada com activada de forma predeterminada però ho desestimarem, ja que donarem més pes a les dades que volen visualitzar els usuaris de les que volen compartir (al cap i a la fi és una aplicació destinada a consultar l'estat dels conductors). Per aquesta raó, la localització estarà activada de forma predeterminada, podent desactivar-la a desig de l'usuari.

3.5.Indagació contextual

Per a poder definir l'aspecte i reforçar les funcionalitats de l'aplicació realitzarem observacions de les aplicacions similars que hem comentat en l'apartat 3.1. Un cop realitzades, mostrarem un prototip de l'aplicació que volem realitzar.

Les observacions, en primer lloc, tenen l'objectiu de realitzar un test d'usabilitat de l'aplicació *Gas Biker* i *Android Auto* (no podem realitzar aquest test per a *Google Latitude, ja que* no està disponible). Un segon objectiu, volem aconseguir informació qualitativa dels usuaris observats perquè ens informin dels aspectes més i menys rellevants de cada aplicació. Com a últim objectiu, volem veure si el primer disseny de l'aplicació que hem realitzat s'adapta a les necessitats de l'usuari observat.

En total s'han realitzat 3 observacions d'uns 30 minuts aproximadament per observació. Cada observació constava de 3 parts, on en cada part es buscava els objectius explicats anteriorment. A més a més, es va portar a l'usuari a l'interior d'un vehicle per situar-lo en context.

Com a les enquestes van predominar 3 rangs d'edats s'han cercat un representant de cada rang d'edat amb l'objectiu d'aconseguir informació més variada.

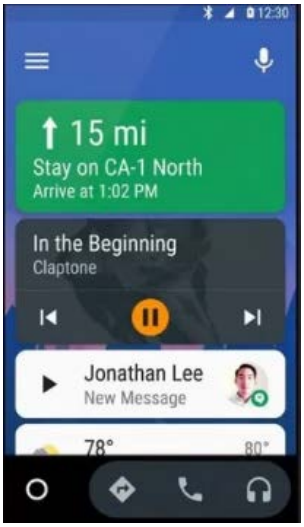

Figura 21: pantalla d'Android Auto que van destacar els usuaris.

3.5.1. Test d'usabilitat

De les dues aplicacions que van avaluar els usuaris van destacar la no sobrecàrrega de les pantalles. Per ells va ser fàcil diferenciar els elements de cada pantalla a causa de les agrupacions d'informació en rectangles, tal com mostra la Figura 21. D'altra banda van destacar el fons blanc de l'aplicació Gas Biker per davant dels colors d'Android Auto, ja que els permetia veure amb claredat els elements de la pantalla (l'usuari de més edat va rebutjar la combinació de colors *d'Android Auto*).

Aprofitant l'observació es va preguntar si estarien d'acord en rebre una ajuda inicial per entendre l'ús de l'aplicació i tots ho van desitjar.

3.5.2. Aspectes interessants de les aplicacions similars

Per part de *Gas Biker* la funcionalitat de l'alerta van mostrar molt d'interès però no voldrien alertes automàtiques. Per part de l'usuari més gran, no veu fiable que s'enviï una alerta de forma automàtica, ja que pot ser equivocada i generar preocupacions als membres del grup, però valora positivament que es pugui enviar una alerta de forma

manual. Per part d'*Android Auto* van destacar que es puguin realitzar tota mena d'accions amb les comandes de veu.

3.5.3. Presentació del primer disseny

Un cop acabades les observacions amb aplicacions similars, els hi mostrarem una primera versió del disseny de la nostra aplicació. Per mostrar la interfície utilitzarem un dispositiu mòbil Android on està instal·lada l'aplicació on només conté la interfície sense cap funcionalitat. La forma triada de mostrar-la és per posar a l'usuari en un context real amb una aplicació real. Com es pot veure a les figures 22 a 26, el color predominant és el blanc amb matisos grisos per separar les àrees de la pantalla. En el cas del mode conducció s'utilitzen colors per a diferenciar la informació actual del conductor per evitar grans intervals de temps mirant la pantalla.

Per veure les impressions dels usuaris els demanarem que realitzin una sèrie de tasques:

1. Activar i desactivar el mode conducció
2. A partir de la informació d'un usuari que està conduint que ens indiqui la seva localització i si accepta trucades.
3. Crear un grup autoritzant a dues persones a veure la localització, si està conduint i veure les imatges frontals.

Activar i desactivar el mode conducció

Per activar i desactivar el mode conducció els usuaris han de seleccionar la primera entrada del menú principal amb la icona del volant. Seguidament, tornar a polsar el volant de la part superior dreta de la Figura 23. La majoria d'usuaris trobaven la pantalla correctament de manera instintiva, ja que polsaven el primer botó que veien, però després trigaven uns 15 segons a localitzar el volant. El color del volant indica si està activat o desactivar el mode conducció però un usuari ens indica que estaria millor indicar-ho d'igual manera s'indica els altres elements d'aquesta pantalla.

Indicar la localització i si accepta trucades d'un usuari

A l'hora d'indicar la localització els usuaris han agraït la introducció d'un mapa expansible, ja que no sempre es vol tindre en pantalla completa. D'altra banda han entès on es localitza la resta d'informació de l'usuari però els hi agradaria veure aquelles dades binàries (sí o no) juntes i no com estan ordenades actualment. A més a més, han comentat que la localització actual del conductor desitjarien veure com es va actualitzant a mesura que canvia i no haver de seleccionar el botó de la part superior dreta d'actualitzar.

Crear un grup

Aquesta tasca els usuaris han trobat ràpidament l'entrada de menú a causa de la icona que representa a persones. De manera ràpida han vist com crear un grup i autoritzar dades a ser visualitzades, però a l'hora d'afegir usuaris han trobat dificultats de com introduir-los, ja que trobaven a faltar alguna etiqueta que indiqués "Usuaris autoritzats" o "Afegeix usuari al grup". A més a més, les icones que podem veure a la Figura 26 les agradaria distingir-los d'alguna forma (l'usuari més gran ens va indicar amb una vora o fons que determini els límits del botó).

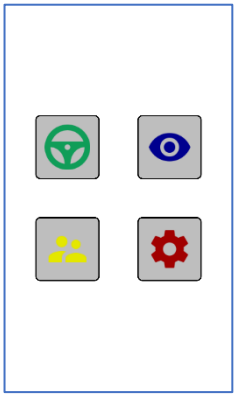

Figura 22: menú principal.

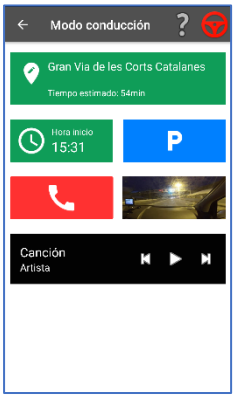

Figura 23: mode conducció.

Figura 24: visualització d'un usuari que esta conduint.

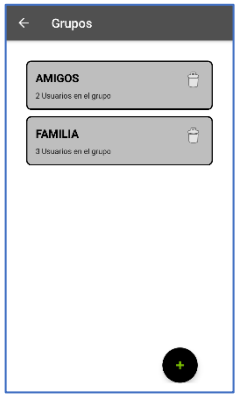

Figura 25: llistat de grups.

Figura 26: creació i modificació d'un grup existent.

Després de realitzar un estudi més directe, considerarem tots els canvis que ens han comentat els usuari per part de la interfície i afegirem una funcionalitat, no pensada a l'inici d'aquest apartat, d'enviar alertes de forma manual a la resta d'usuaris que hem seleccionat en el grup.

3.6. Diagrama de casos d'ús

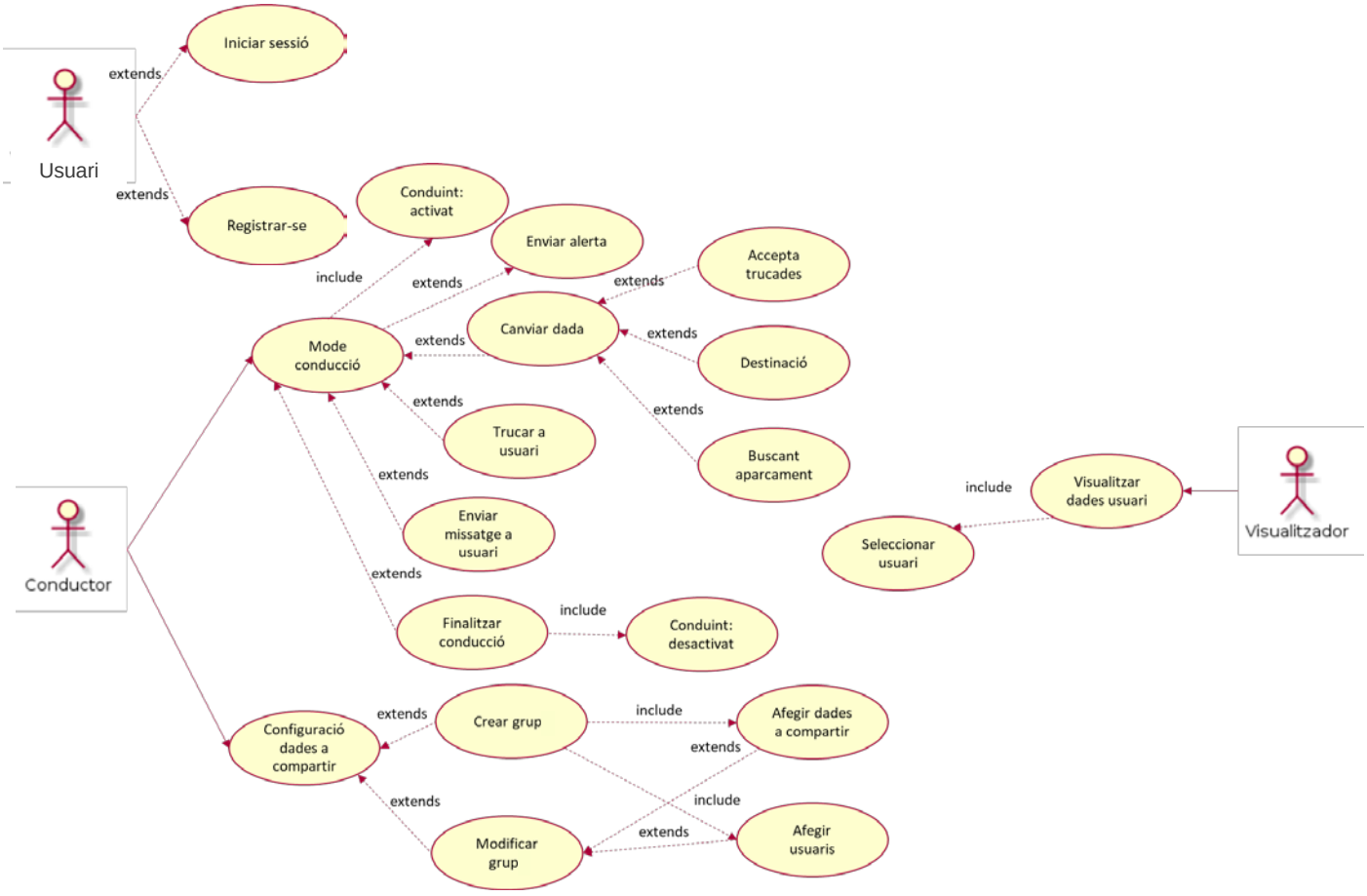

Figura 27: Diagrama de casos d'ús.

3.7.Explicació diagrama de casos d'ús

Nom	**UC1 – Iniciar Sessió**
Actor	Usuari
Descripció	Procés per accedir a l'aplicació.
Precondicions	L'usuari es troba a la pantalla d'inici de l'aplicació i té un usuari registrat.
Flux bàsic	1. L'aplicació mostra el formulari a l'usuari per a poder iniciar sessió. 2. L'usuari introdueix el correu electrònic i la contrasenya. 3. L'usuari envia les credencials. 4. El servidor comprova les credencials i inicia sessió.
Flux alternatiu	3.1. L'usuari introdueix un correu electrònic amb format invàlid. 3.2. El sistema mostra un missatge d'error a l'usuari 4.1. El servidor detecta que les credencials no són vàlides, s'atura l'inici de sessió i es mostra un missatge d'error a l'usuari.
Postcondicions	L'usuari té la sessió iniciada i se situa a la pantalla principal.

Nom	**UC2 – Registrar-se**
Actor	Usuari
Descripció	Procés per registrar un compte a l'aplicació.
Precondicions	L'usuari es troba a la pantalla de registre.
Flux bàsic	1. L'usuari selecciona l'opció de registrar-se. 2. El sistema mostra el formulari a l'usuari per a poder registrar-se. 3. L'usuari introdueix el nom, el correu electrònic i la contrasenya. 4. L'usuari confirma el registre. 5. El servidor comprova les dades, registra a l'usuari i inicia sessió.
Flux alternatiu	3.3. L'usuari introdueix un correu electrònic o contrasenya amb format invàlid. 3.4. El sistema mostra un missatge d'error a l'usuari 5.1. El servidor detecta que les dades no són vàlides, es mostra un missatge d'error a l'usuari.
Postcondicions	L'usuari té la sessió iniciada i se situa a la pantalla principal.

Nom	**UC3 – Mode conducció**
Actor	Conductor
Descripció	Procés on l'usuari procedeix a activar el mode conducció
Precondicions	El conductor es troba a la pantalla principal de l'aplicació.
Flux bàsic	1. El conductor selecciona l'opció de mode conducció. 2. El sistema mostra la pantalla del mode conducció. 3. El conductor activa la conducció.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció i compartint dades.

Nom	**UC4 – Conduint: activat**
Actor	Conductor
Descripció	Procés on s'activa la dada de conducció.
Precondicions	El conductor es troba a la pantalla del mode conducció i ha activat la conducció.
Flux bàsic	1. El sistema activa la dada de conducció i l'envia al servidor per a guardar-la a base de dades i compartir-la amb la configuració de grups establerta.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció

Nom	**UC5 – Enviar alerta**
Actor	Conductor
Descripció	Procés on s'envia una alerta a tots els usuaris dels grups configurats.
Precondicions	El conductor es troba a la pantalla del mode conducció amb la conducció activada.
Flux bàsic	1. El conductor indica que vol enviar una alerta a tots els usuaris dels grups configurats. 2. El sistema notifica a l'usuari que l'alerta ha sigut enviada correctament.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció.

Nom	**UC6 – Canviar dada**
Actor	Conductor
Descripció	Procés on el conductor canvia el valor d'una dada.
Precondicions	L'usuari es troba a la pantalla del mode conducció amb la conducció activada.
Flux bàsic	1. El conductor indica que vol enviar una alerta a tots els usuaris dels grups configurats. 2. El sistema pregunta a l'usuari quina dada vol canviar. 3. El conductor selecciona la dada que vol canviar. 4. El sistema envia el nou valor de la dada al servidor per a canviar-la a la base de dades.
Flux alternatiu	2.1. El conductor desestima el canvi de la dada.
Postcondicions	El conductor es troba a la pantalla del mode conducció.

Nom	**UC7 – Accepta trucades**
Actor	Conductor
Descripció	Procés on el conductor canvia el valor de la dada d'acceptar trucades.
Precondicions	El conductor es troba a la pantalla del mode conducció amb la conducció activada i ha seleccionat canviar la dada d'acceptar trucades.
Flux bàsic	1. El sistema canvia l'estat d'acceptar trucades.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció.

Nom	**UC8 – Destinació**
Actor	Conductor
Descripció	Procés on el conductor canvia el valor de la dada destinació.
Precondicions	El conductor es troba a la pantalla del mode conducció amb la conducció activada i ha seleccionat canviar la dada destinació.
Flux bàsic	1. El sistema pregunta a l'usuari quina es la nova destinació. 2. El conductor indica la nova destinació.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció.

Nom	**UC9 – Buscant aparcament**
Actor	Conductor
Descripció	Procés on el conductor canvia el valor de la dada de buscant aparcament.
Precondicions	L'usuari es troba a la pantalla del mode conducció amb la conducció activada i ha seleccionat canviar la dada de buscant aparcament.
Flux bàsic	1. El sistema canvia l'estat buscant aparcament.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció.

Nom	**UC10 – Trucar a usuari**
Actor	Conductor
Descripció	Procés on el conductor realitza una trucada a un usuari.
Precondicions	El conductor es troba a la pantalla del mode conducció amb la conducció activada.
Flux bàsic	1. El conductor indica que vol trucar a un usuari. 2. El sistema pregunta al conductor el nom de l'usuari que vol trucar. 3. El conductor indica el nom de l'usuari que vol trucar. 4. El sistema reconeix l'usuari i procedeix a realitzar la trucada.
Flux alternatiu	3.1. El sistema no reconeix el nom de l'usuari i informa al conductor que no ha trobat l'usuari amb el nom indicat
Postcondicions	L'usuari es troba a la pantalla del mode conducció.

Nom	**UC11 – Enviar missatge a un usuari**
Actor	Conductor
Descripció	Procés on el conductor envia un missatge a un usuari.
Precondicions	El conductor es troba a la pantalla del mode conducció amb la conducció activada.
Flux bàsic	1. El conductor indica que vol enviar un missatge a un usuari. 2. El sistema pregunta al conductor el nom de l'usuari que vol enviar un missatge. 3. El conductor indica el nom de l'usuari que vol enviar un missatge. 4. El sistema pregunta al conductor quin és el missatge. 5. L'usuari indica el missatge que vol enviar. 6. El sistema reconeix l'usuari i procedeix enviar el missatge.
Flux alternatiu	3.2. El sistema no reconeix el nom de l'usuari i informa al conductor que no ha trobat l'usuari amb el nom indicat
Postcondicions	L'usuari es troba a la pantalla del mode conducció.

Nom	**UC12 – Finalitzar la conducció**
Actor	Conductor
Descripció	Procés on el conductor finalitza la conducció.
Precondicions	El conductor es troba a la pantalla del mode conducció amb la conducció activada.
Flux bàsic	1. El conductor indica que vol finalitzar la conducció. 2. El sistema desactiva la conducció.
Flux alternatiu	
Postcondicions	L'usuari es troba a la pantalla del mode conducció.

Nom	**UC13 – Conduint: desactivat**
Actor	Conductor
Descripció	Procés on es desactiva la dada de conducció.
Precondicions	El conductor es troba a la pantalla del mode conducció i ha desactivat la conducció.
Flux bàsic	1. El sistema desactiva la dada de conducció i l'envia al servidor per a guardar-la a base de dades i compartir-la amb la configuració de grups establerta.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla del mode conducció amb la conducció desactivada.

Nom	**UC14 – Configuració dades a compartir**
Actor	Conductor
Descripció	Procés on el conductor configura els grups i les dades a compartir a aquest.
Precondicions	El conductor es troba a la pantalla principal de l'aplicació.
Flux bàsic	1. El conductor indica que vol configurar els seus grups. 2. El sistema mostra els grups que actualment té l'usuari.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla de configuració de grups.

Nom	**UC15 – Crear grup**
Actor	Conductor
Descripció	Procés on el conductor crea un grup.
Precondicions	El conductor es troba a la pantalla de configuració de grups.
Flux bàsic	1. El conductor indica que crear un grup. 2. El sistema mostra el formulari per crear un grup. 3. El conductor introdueix el nom del grup. 4. El conductor afegeix usuaris al grup. 5. El conductor selecciona les dades que vol compartir amb el grup. 6. El conductor guarda el grup. 7. El sistema comprova que el formulari sigui correcte, enviar el grup al servidor per a guardar-lo a base de dades i el porta a la pantalla de configuració de grups.
Flux alternatiu	7.1. El sistema detecta que hi ha un error en el formulari i informa a l'usuari amb un missatge d'error.
Postcondicions	El conductor es troba a la pantalla de configuració de grups.

Nom	**UC16 – Afegir dades a compartir**
Actor	Conductor
Descripció	Procés on el conductor afegeix dades a compartir a un grup.
Precondicions	El conductor es troba a la pantalla de crear un grup.
Flux bàsic	1. El sistema mostra la llista de dades que el conductor pot compartir. 2. El conductor selecciona aquelles dades que vol compartir
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla de crear un grup.

Nom	**UC17 – Afegir usuaris**
Actor	Conductor
Descripció	Procés on el conductor afegeix usuaris a un grup per a compartir dades a aquests.
Precondicions	El conductor es troba a la pantalla de crear un grup.
Flux bàsic	1. El conductor introdueix el correu electrònic de l'usuari que vol afegir al grup. 2. El sistema notifica al conductor que ha afegit l'usuari correctament.
Flux alternatiu	2.1. El sistema notifica al conductor que el correu electrònic té un format invàlid.
Postcondicions	El conductor es troba a la pantalla de crear un grup.

Nom	**UC18 – Visualitzar dades usuari**
Actor	Visualitzador
Descripció	Procés on visualitzador indica que vol veure les dades d'un usuari.
Precondicions	El visualitzador es troba a la pantalla principal
Flux bàsic	1. El visualitzador indica que vol veure les dades d'un usuari. 2. El sistema mostra un llistat dels conductors que han habilitat al visualitzador poder veure les seves dades. 3. El visualitzador selecciona un conductor. 4. El sistema mostra les dades del conductor.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla de d'un conductor visualitzant les dades.

Nom	**UC19 – Seleccionar usuari**
Actor	Visualitzador
Descripció	Procés on el visualitzador selecciona a un usuari per a veure les seves dades.
Precondicions	El visualitzador es troba a la pantalla que mostra el llistat d'usuaris que han habilitat al visualitzador poder veure les seves dades.
Flux bàsic	1. El visualitzador selecciona al conductor que vol visualitzar. 2. El sistema mostra les dades del conductor.
Flux alternatiu	
Postcondicions	El conductor es troba a la pantalla de d'un conductor visualitzant les dades.

4. Implementació

En el següent apartat explicarem el procediment dels diferents desenvolupaments de l'aplicació. Recordem que utilitzarem un model de desenvolupament iteratiu incremental. Per a cada increment es dividirà sempre en les mateixes etapes: requeriments funcionals, disseny de la interfície, implementació i proves.

4.1. Primera iteració

Com es pot veure a la Figura 1, el primer desenvolupament correspon a realitzar l'inici de sessió i el registre a la plataforma.

L'objectiu d'aquest desenvolupament és realitzar el disseny i tota la lògica on el resultat final d'aquesta iteració sigui totalment funcional i definitiu.

4.1.1. Requeriments funcionals

Per a poder diferenciar als usuaris dintre de l'aplicació i poder interaccionar amb altres necessitem que es registrin.

Per a dur a terme el registre caldrà que els usuaris introdueixin:

- **Correu electrònic**: identificador personal de cada usuari, li servirà per mantenir les seves dades guardades en qualsevol dispositiu. Un correu electrònic només pot tindre associada un compte.
- **Contrasenya**: per a poder identificar-se necessita introduir la contrasenya associada al correu electrònic. La contrasenya ha de tindre un mínim de 6 caràcters (establert per la tecnologia Firebase).
- **Nom**: el nom servirà per saber quin és el nom de la persona que gestiona el compte.

4.1.2. Disseny de la interfície

Per a realitzar el disseny ens assegurarem d'utilitzar aquells patrons de disseny propis del sistema Android que ajuden a fer una aplicació usable.

En primer lloc, quan entrem a l'aplicació veiem el logotip d'aquesta amb un formulari a la part inferior. El disseny segueix un estil minimalista i comú a les aplicacions d'avui dia on no pot faltar els elements que es poden veure a les figures 28 - 31 (inicia sessió, recuperar contrasenya i registrar-se). Totes les accions que es realitzen s'obté un feedback visual i textual a l'usuari per prevenció d'errors, que està passant, etc. seguint les heurístiques de Nielsen (Heurístiques de Nielsen, 2018) involucrades (aquestes heurístiques s'aniran estenent a la resta d'increments).

Figura 28: control d'errors alhora d'iniciar sessió amb el formulari buit.

Figura 29: missatge on dona visibilitat a l'usuari quan introdueix un correu que no te associada cap compte.

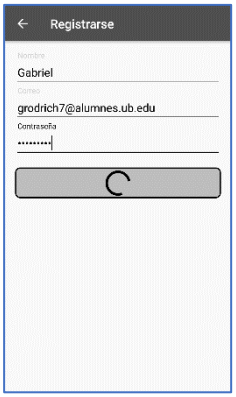

Figura 30: pantalla de registre, es pot observar com s'informa a l'usuari que s'està realitzant la petició amb la barra de progres.

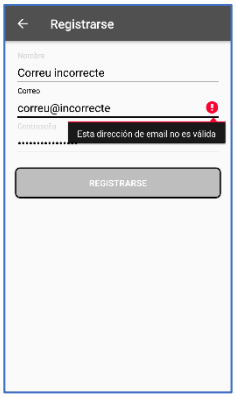

Figura 31: control d'errors alhora d'introduir un correu electrònic incorrecte.

4.1.3. Diagrama de classes

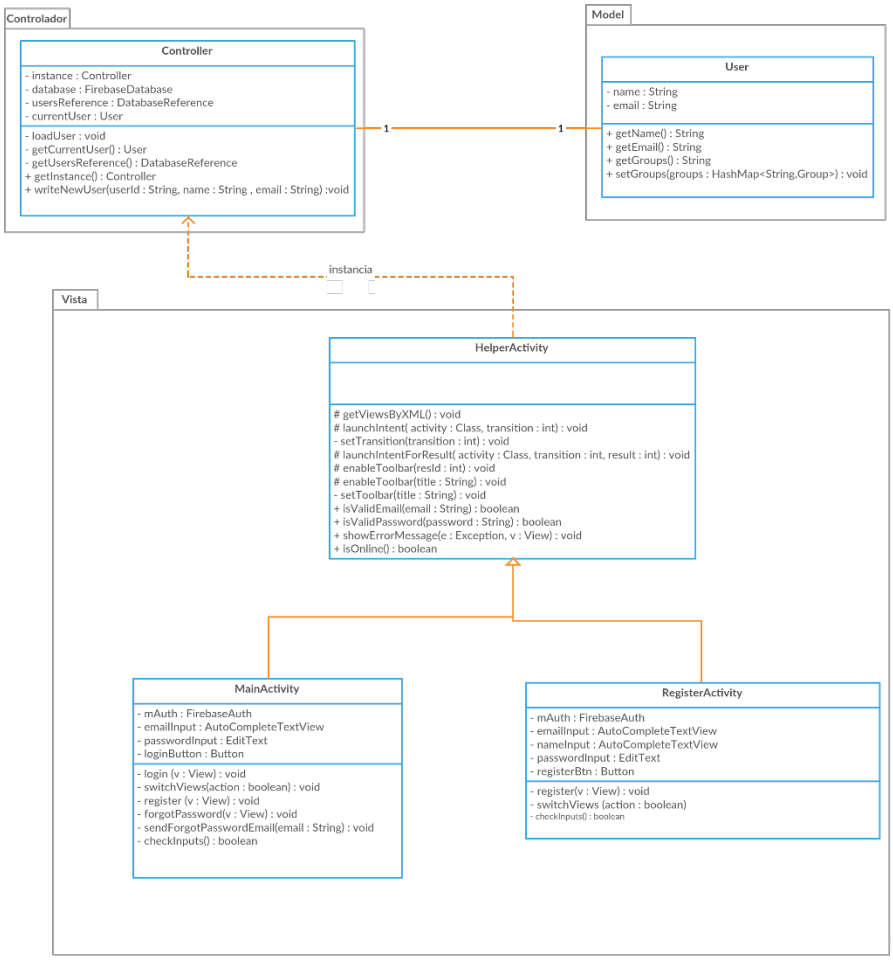

Figura 32: diagrama de les classes utilitzades en aquest desenvolupament.

4.1.4. Implementació

A continuació s'explicarà com ha sigut la implementació de les classes que observem a la Figura 32.

HelperActivity: aquesta classe aporta una base per a les *activities* filles que s'utilitzaran per manipular la interfície. Aquesta classe instancia a la classe *Controller* per realitzar accions sobre el controlador. Els mètodes que trobem els podem classificar en diferents apartats:

- Validacions: per a prevenció d'errors, es creen mètodes genèrics (*isValidEmail*, *isValidPassword*, *isOnline*) per estalviar codi i fomentar a la modulació del codi.
- Elements visuals: a l'aplicació tenim una barra d'accions a la part superior, és per això que es crea i es defineix en aquesta classe mare. A més a més, definim un mètode abstracte, per obligar a totes les classes filles a implementar un mètode per obtenir tots els elements per a poder tractar esdeveniments, validacions, etc. (*getViewsByXML*).
- Errors: en realitzar connexions amb la base de dades poden ocórrer diferents errors que han de ser tractats, és per això que creem el mètode *showErrorMessage*, on donat un error genèric donem un missatge específic tractant l'error genèric. Tota acció dirigida contra el servidor on s'allotja la base de dades serà tractada, i en cas de fallada es mostrarà un error específic amb la funció esmenada.

MainActivity: la primera classe que apareix a l'aplicació conté el formulari per a iniciar sessió o per dirigir-se al registre. És l'encarregada d'utilitzar les funcions que ens aporta *FirebaseAuth* per a poder iniciar a un usuari o recuperar la contrasenya d'aquest. També és l'encarregada de gestionar els errors que puguin aparèixer al formulari. Inclou el mètode login, on realitza aquesta comprovació dels errors abans i després d'iniciar sessió mitjançant el correu i la contrasenya proporcionada. En cas de ser un inici de sessió exitós dirigirà a l'usuari a la pantalla principal de l'aplicació (s'implementarà en el següent increment).

```
public void login(final View v){
    if (!checkInputs()) return;
    switchViews(false);
    hideKeyboard();
    mAuth.signInWithEmailAndPassword(emailInput.getText().toString(), passwordInput.getText().toString())
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    updateUI(task.isSuccessful() ? mAuth.getCurrentUser() : null);
                    switchViews(true);
                }
            }).addOnFailureListener(this, new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    switchViews(true);
                    showErrorMessage(e, emailInput);
                }
            });
}
```

Figura 33: mètode login de l'aplicació. Es pot veure la gestió d'error prèvia i posterior a l'acció.

RegisterActivity: de la mateixa manera que *MainActivity* aquesta classe té la funció de mostrar un formulari on l'usuari es pot registrar amb un correu electrònic. Quan un usuari realitza un registre satisfactori (al servidor que gestiona l'autenticació de l'aplicació), es procedeix a crear l'usuari a base de dades per primera vegada mitjançant la classe *Controller*. Això ens permet emmagatzemar les dades del registre i altres dades en un futur.

Controller: aquesta classe segueix el patró *Singleton* (1 instància en tota l'execució) i és l'encarregada de controlar totes les accions lògiques i les connexions al servidor de Firebase. També emmagatzema l'usuari actual amb una instància de la classe *User* que ens permet disposar de les dades de l'usuari un cop recuperades de la base de dades.

User: com a única classe del model trobem una classe simple per encapsular la informació d'un usuari. En un futur ens servirà per emmagatzemar els grups.

L'autenticació d'usuaris es realitza mitjançant la llibreria *FirebaseAuth* que incorpora mètodes per gestionar-ho i seguint el procés tal com ens indica la documentació de Firebase (Firebase Authentication, 2018).

4.2. Segona iteració

Un cop tenim els usuaris donats d'alta, en aquest segon desenvolupament realitzarem la configuració de grups i dades a compartir. Gràcies a aquest desenvolupament els usuaris podran dir a quines dades comparteixen i aquí les compartiran.

4.2.1. Requeriments funcionals

Per començar hem de proporcionar als usuaris una sèrie d'accions:

- **Veure els grups**: els usuaris han de poder visualitzar quins són els seus grups de forma clara.
- **Crear un grup**: l'usuari ha de tindre l'opció de crear un grup per especificar les dades que li vol compartir i a quins usuaris. No es podrà crear un grup amb usuaris repetits.
- **Modificar (o visualitzar) un grup**: la modificació i visualització d'un grup ho podem simplificar en la mateixa acció perquè l'usuari realitzi els canvis adients.
- **Eliminar grup**: hem d'habilitar a l'usuari poder eliminar un grup quan ho trobi oportú.

A més a més, quan un usuari afegeix a d'altres a un grup, aquests usuaris afegits passaran a ser visualitzadors de l'usuari el qual li ha afegit i, per tant, han de tindre la manera d'assabentar-se.

4.2.2. Disseny de la interfície

Per realitzar els objectius d'aquest increment necessitem noves pantalles que ens permetin realitzar les accions comentades a la secció anterior.

Per començar, després de l'inici de sessió hem de situar a l'usuari en la pantalla principal de la nostra aplicació, on una vegada hagi posat les credencials pugui començar a navegar lliurement. A l'hora de navegar, podem navegar a 3 seccions accessibles a partir dels botons del menú principal (Figura 34).

En primer lloc, la configuració de grups està composta per 2 pantalles, el llistat de grups (Figura 35) i la creació/modificació d'un grup (Figura 36 i Figura 37). Per reaprofitar dissenys i codi font la pantalla de creació i modificació són la mateixa, ja que no hi ha cap diferència funcional, però realitzarem el codi font de manera que es pugui diferenciar quan creem o modifiquem.

En segon lloc, la icona blava de la pantalla principal, ens portaria a la Figura 38. Aquesta mostra un llistat d'usuaris que ens han donat permís per visualitzar-los. D'aquesta manera veure si la vinculació d'usuaris a un grup ha sigut satisfactori.

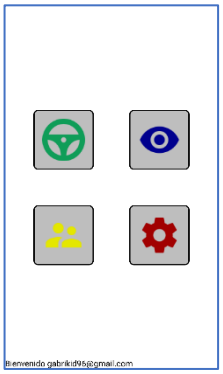

Figura 34: pantalla principal de l'aplicació.

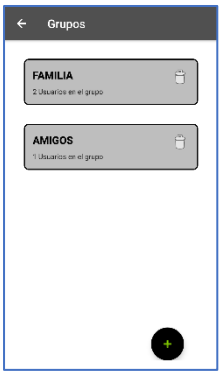

Figura 35: Un cop polsat la icona groga de configuració de grups, apareixerà el llistat de grups de l'usuari.

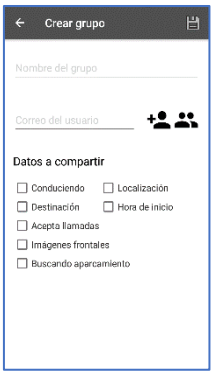

Figura 36: pantalla de creació d'un grup, es pot veure 3 zones, el nom del grup, afegir usuaris al grup i les dades a compartir.

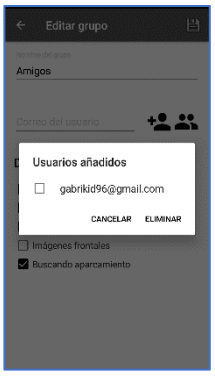

Figura 37: diàleg que ens permet veure els usuaris afegits. S'accedeix desde la icona de dues persones.

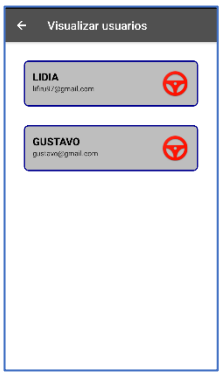

Figura 38: pantalla de visualització d'usuaris. Podem accedir des de la icona blava del menú principal.

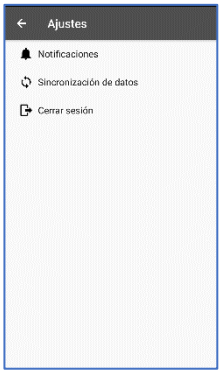

Figura 39: pantalla de preferències de l'aplicació. Ens permet tancar sessió per canviar d'usuari.

4.2.3. Diagrama de classes

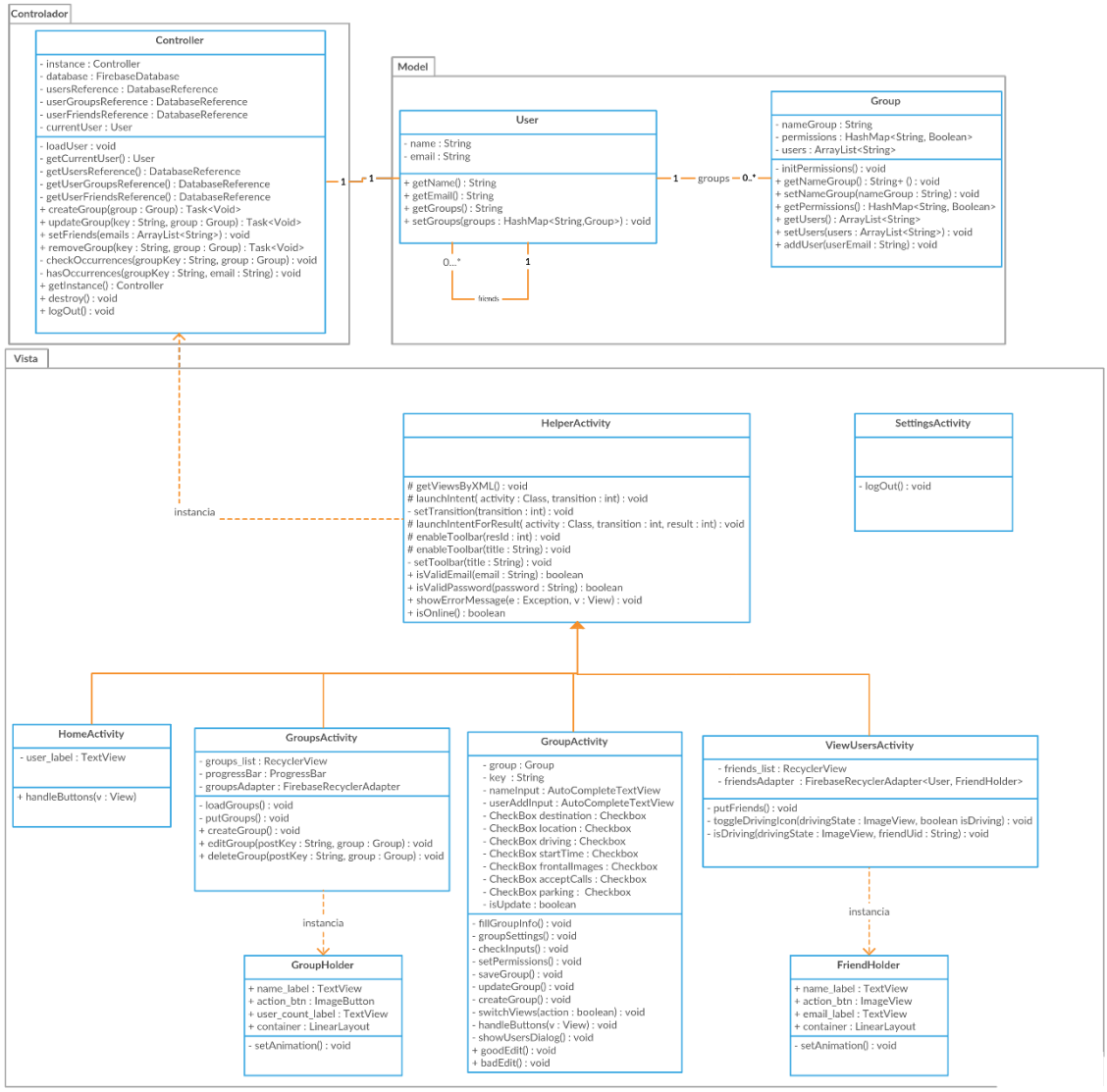

Figura 40: Diagrama de classes del segon increment.

4.2.4. Implementació

En primer lloc, hem afegit un nou atribut *groups* a la classe *User* per a poder relacionar aquesta classe amb la classe *Group*. La relació consta d'un diccionari a la classe *User* on la clau del diccionari pertany a una cadena de text (*String*) i com a valor tindrem un objecte de tipus *Group*. Gràcies a aquesta relació entre models, a l'hora de guardar informació a la base de dades, gràcies a l'API de Firebase, a l'hora de guardar o recuperar dades ens ho retornarà tal com hem realitzat els models.

Encara que la base de dades de Firebase és no relacional, intentarem simular aquest comportament a partir d'identificadors únics. En el cas de la relació *User-Group* a la base de dades ho guardarem en forma d'arbre, on la classe *User* serà el node pare dels nodes *Group*, cada node tindrà el seu propi identificador únic.

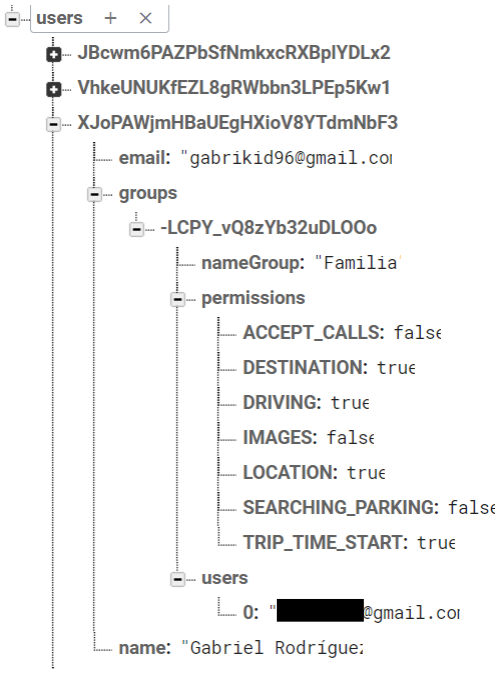

Figura 41: arbre de la base de dades per emmagatzemar usuaris i les seves relacions.

En la Figura 41, podem veure que guardarem tots els usuaris en un node pare anomenat "users". Aquesta referencia ens servirà de punter per emmagatzemar i obtenir dades dels usuaris. Podem veure que tenim una sèrie d'usuaris identificats per la clau única que ens proporciona el mòdul *FirebaseAuth* (comentat al primer increment) un cop es dona d'alta un nou usuari. En el node d'un usuari podem veure els atributs propis de la classe *User*. L'atribut *groups* conté una llista de nodes de tipus *Group* que representen els grups que té un usuari i la seva configuració.

Per emmagatzemar la configuració d'un grup, a la classe *Group*, crearem 3 atributs: nom del grup, dades a compartir i una llista d'usuaris. Les dades a compartir es basen en una llista de booleans i la llista d'usuaris consisteix a guardar el correu electrònic de l'usuari que vols afegir al grup.

En segon lloc, hi ha 4 noves classes per a noves pantalles i 2 classes auxiliars:

HomeActivity: classe que ens apareixerà quan iniciem l'aplicació si tenim la sessió d'usuari activa. Encarregada de capturar els esdeveniments dels 4 botons i de portar a l'usuari a la pantalla indicada.

GroupsActivity: classe que mostrarà la Figura 35, ens permetrà veure els grups que té l'usuari i gestionar-los. Per mostrar els grups demanarem a la classe *Controller* la referència dels grups de l'usuari a partir del mètode *getUserGroupsReference*. Aquest mètode ens permetrà apuntar a la zona de la base de dades on l'usuari té emmagatzemat els grups i a partir d'un objecte *FirebaseRecyclerAdapter* i la classe auxiliar *GroupHolder* (contenidor d'informació de cada grup) podrem mostrar la llista de grups de l'usuari actual.

GroupActivity: classe encarregada de mostra la Figura 36 a l'usuari. Aquesta classe ens servirà per crear un grup, modificar-lo o visualitzar-lo. En cas de voler visualitzar o modificar un grup, passarem un paràmetre des de la classe *GroupsActivity* amb l'objecte *Group* seleccionat. Aquest objecte l'utilitzarem per carregar la pantalla (mètode *fillGroupInfo*) amb la informació del grup.

En cas de crear un grup, realitzarem les validacions tal com vàrem fer a l'increment 1, cridant als mètodes de la classe abstracta *HelperActivity*. Un cop la informació entrada per l'usuari és correcta es procedeix a crear-lo.

Les accions que es poden realitzar sobre un grup les controla la classe *Controller*. Cada acció té un mètode (*createGroup*, *updateGroup* i *removeGroup*) associat que realitza l'acció desitjada i actualitza una informació important per a continuar desenvolupant aquesta aplicació. A continuació s'explica com es crea aquesta informació i on es visualitza.

Quan un usuari afegeix a un altre a un grup, l'usuari afegit necessita assabentar-se que pot visualitzar la informació d'un conductor. Per això es necessari realitzar un procés per indicar quins usuaris pot visualitzar, aquests usuaris els anomenarem amics. El mètode que realitza aquest procés s'anomena setFriends. Quan es crea o s'actualitza un grup hem de recorre la llista d'usuaris del grup per afegir a la referencia (node de l'arbre de base de dades) de cada usuari l'identificador únic de l'usuari que li ha afegit.

```
public void setFriends(ArrayList<String> emails){

    for (final String email : emails){
        usersReference.orderByChild("email").equalTo(email).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                try{
                    String uid = (String)((HashMap) dataSnapshot.getValue()).keySet().toArray()[0];
                    DatabaseReference ref = usersReference.child(uid).child("friends"). //uid del que vamos a añadirle como amigo
                            child(FirebaseAuth.getInstance().getCurrentUser().getUid());
                    ref.setValue(new User(currentUser.getName(),currentUser.getEmail()));
                }catch (NullPointerException ex){
                    Log.e("CONTROLLER", "unknown email: " + email);
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
}
```

Figura 42: mètode que escriu a la referencia de cada usuari afegit, la informació de l'usuari que li ha afegit.

En crear el grup, com es demana el correu electrònic no podem aconseguir una referència d'usuari, ja que no és l'identificador únic. Per aquesta raó, hem de fer una cerca a la referència d'usuaris a partir del correu electrònic (l'equivalent a fer una sentència *WHERE* de *SQL* pel camp *email*). Un cop tenim l'identificador únic, afegim al node que hem trobat la informació un atribut *friends* de l'usuari que ha creat el grup. En el cas d'afegir un correu electrònic que no existeix a l'aplicació simplement no s'informarà res a cap node.

Figura 43: node d'un usuari. És pot veure l'atribut friends on l'usuari Gabriel Rodriguez ha afegit a un grup l'usuari amb un correu.

ViewUsersActivity: classe encarregada de mostrar als usuaris que podrem visualitzar. Els usuaris que podem visualitzar estan a l'atribut *friends* gràcies al procés que hem explicat anteriorment. A la Figura 38, es llista els usuaris que podem visualitzar juntament amb una icona que representa si està conduint, fins al Quarta iteració no veurem aquesta icona canviar d'estat.

SettingsActivity: aquesta classe no hereta de la classe mare perquè no necessita cap funcionalitat d'aquesta. Simplement hereta de la classe ActivityCompat, és a dir, és una *activity* genèrica d'Android. La funció d'aquesta classe en aquest increment és permetre'ns canviar d'usuari fàcilment a partir de la funció *logOut* i emmagatzemar les preferències de l'usuari a l'aplicació.

4.3. Tercera iteració

En aquest tercer desenvolupament, realitzarem una part important de l'aplicació, ja que un cop acabat ens aproparem al nostre objectiu final. Un cop tenim els usuaris i la configuració dels grups, podem passar a compartir les dades.

4.3.1. Requeriments funcionals

- **Definir les dades a compartir**: per poder compartir dades, hem de proporcionar a l'usuari la forma fàcil de definir les dades com: destinació, accepta trucades, etc.
- **Mostrat l'estat de les dades**: en tot moment l'usuari ha de ser capaç de veure l'estat de les seves dades.
- **Compartir dades als grups**: l'usuari ha de configurar primerament els grups, ha d'especificar a quins usuaris vol compartir dades i quines.

Per aquest desenvolupament, definirem una estructura a la base de dades que ens permeti compartir aquestes dades.

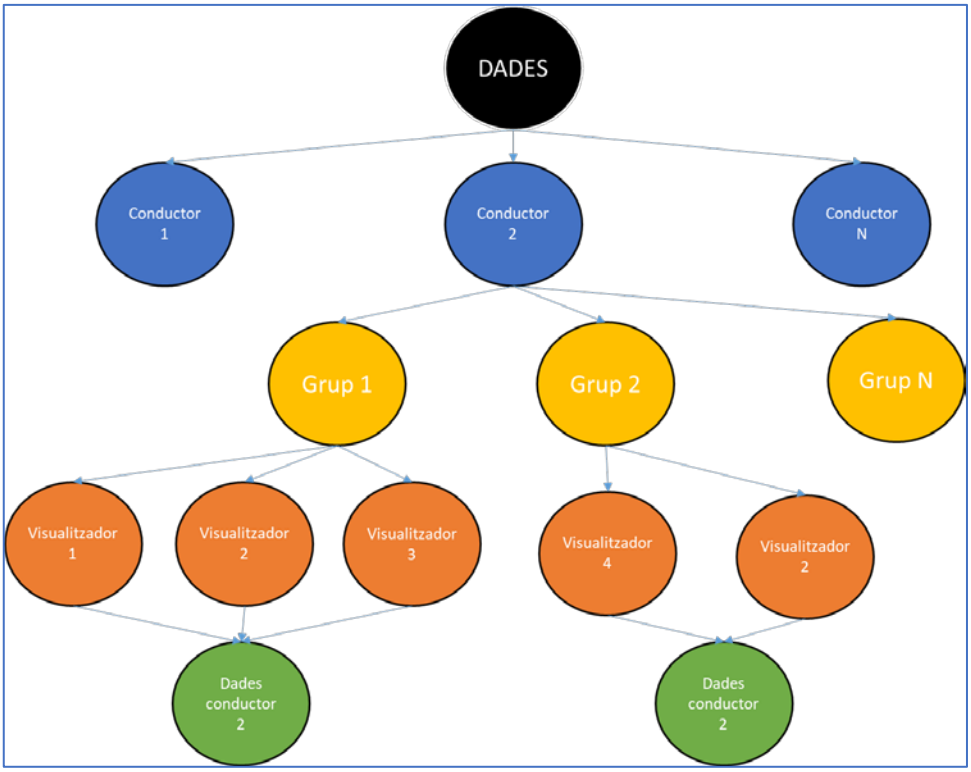

Figura 44: estructura de la base de dades que ens servirà per a emmagatzemar les dades a compartir depenent del grup i de l'usuari.

Realitzarem una estructura amb 4 nivells de profunditat on explicarem a continuació cada nivell:

1. Conductors: en aquest nivell (nodes blaus) s'identifica quin és el conductor del qual es vol emmagatzemar dades. Com a fills, es tindran els grups que ha configurat prèviament.
2. Grups: com els conductors tenen configurats diferents grups (nodes grocs) per a cada grup s'hauran de compartir les dades que el conductor hagi permès.
3. Visualitzadors: aquest nivell (nodes taronges) ens permet relacionar les dades que un usuari pot veure d'un altre. Sense aquest nivell no hi hauria forma d'aconseguir aquesta relació, ja que el visualitzador no coneix els grups del conductor i per tant, no es pot assabentar de les dades del conductor. Com a fills té les dades del conductor.
4. Dades: finalment arribem al node (node verd) més important per a poder compartir les dades. Cada node fulla contindrà només aquelles dades permeses al grup, és a dir, per a cada conductor pot existir fins a N nodes fulles diferents, on N és la quantitat de grups d'un usuari. També pot ocorre que un usuari estigui en diferents grups, llavors les dades permeses a visualitzar per aquest, hauria de ser el conjunt de les dades permeses dels diferents grups.

4.3.2. Disseny de la interfície

El context de compartir les dades se situa a la pantalla del mode conducció (primera entrada del menú principal). Encara que el mode conducció no ha sigut desenvolupat (últim increment) podem compartir les dades de forma manual, és a dir, accionant un botó. Per aquesta raó compartir dades es pot implementar de forma independent al mode conducció i ens servirà com a base pel desenvolupament futur. Quan el mode conducció estigui realitzat, es compartirà de manera automàtica i periòdica.

D'altra banda, al disseny permet modificar les dades que es compartiran, excepte la localització i les imatges (es desenvoluparà a l'últim increment). De moment no s'incorporaran imatges però per poder compartir un exemple de localització utilitzarem la localització de la Facultat de Matemàtiques.

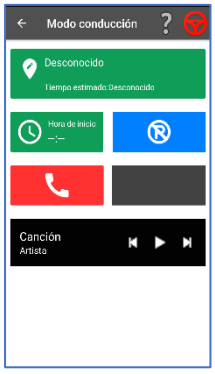

Figura 45: mode conducció, encara que no ha sigut desenvolupat, ens servirà per compartir les dades de forma ràpida.

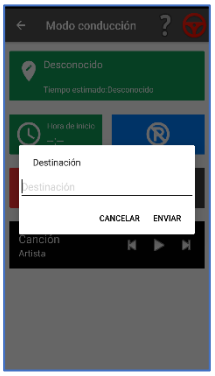

Figura 47: Com la dada destinació cal especificar-la, hem habilitat un diàleg per introduir-la.

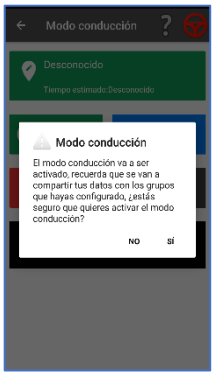

Figura 46: Per a compartir les dades, utilitzarem el botó de mode conducció, encara sense funcionament fins l'últim increment.

4.3.3. Diagrama de classes

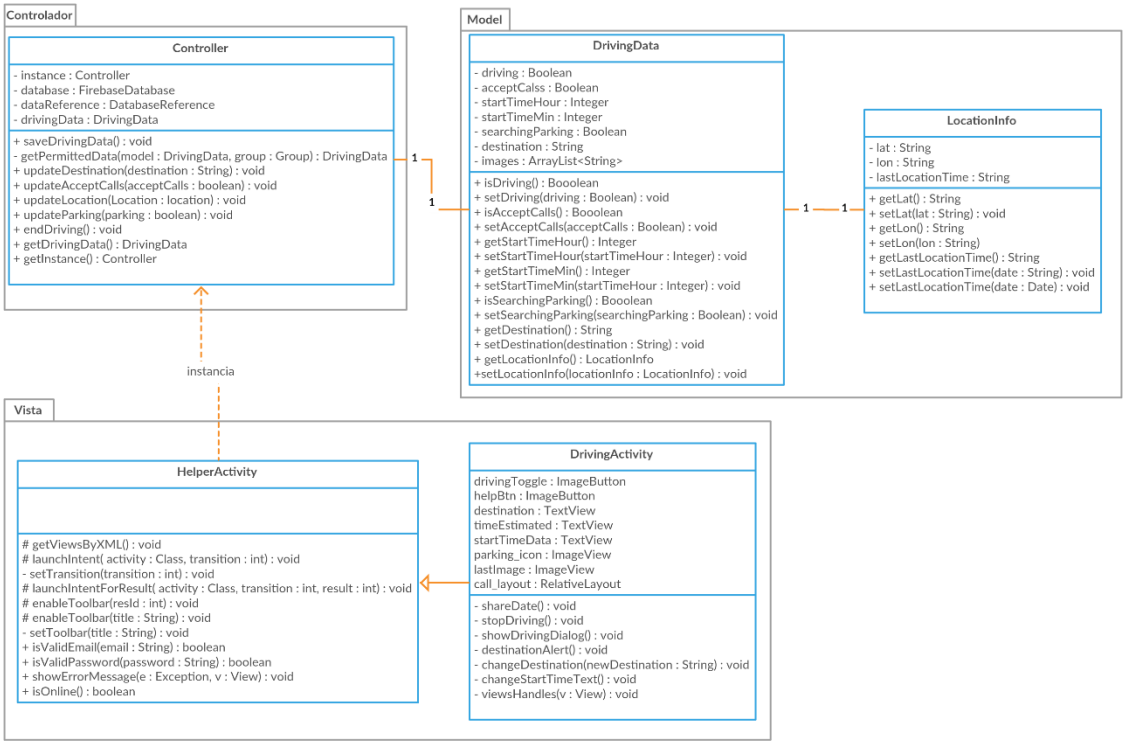

Figura 48: diagrama de classes del tercer increment.

4.3.4. Implementació

Per implementar aquest desenvolupament ens han calgut 3 classes noves: *DrivingData*, *LocationInfo* i *DrivingActivity*. També ha calgut modificar la classe *Controller*, per poder comunicar-nos amb la base de dades.

DrivingData: aquest model ens serveix per encapsular totes les dades que es poden compartir. Com la dada de localització està composta per més atributs conté la relació amb la classe *LocationInfo*.

LocationInfo: com hem comentat anteriorment, la localització es compon per diferents atributs. Serveix per a guardar l'última localització registrada del conductor a partir de la longitud, latitud i la data de la localització.

Controller: per a emmagatzemar les dades de la conducció la classe *Controller* s'encarregarà de mantenir-les i guardar-les a base de dades. Per aquesta raó, hem creat mètodes per guardar tota la instància de DrivingData i altres mètodes per guardar les dades individualment en cas que es modifiquin (no cal tornar a guardar tot l'objecte, només quan canviï, per exemple, acceptar trucades). Cadascun d'aquests mètodes es comprova si el grup té permisos per veure aquesta dada, en cas afirmatiu es guardarà, en cas contrari no.

```
for(final HashMap.Entry<String, Group> entry : currentUser.getGroups().entrySet()) {
    for (final String email : entry.getValue().getUsers()){
        usersReference.orderByChild("email").equalTo(email).addListenerForSingleValueEvent(new ValueEve
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                try{
                    String uid = (String)((HashMap) dataSnapshot.getValue()).keySet().toArray()[0];
                    DatabaseReference friend_ref = ref.child(entry.getKey()).child(uid);//TODO : uidUse
                    if (entry.getValue().getPermissions().get(Constants.Data.DESTINATION.toString())){
                        friend_ref.child("destination").setValue(drivingData.getDestination());
                    }
                }catch (NullPointerException ex){
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
```

Figura 49: exemple com es guarda a base de dades la dada destinació en cas que el grup ho permeti.

En aquet exemple es pot veure com es compleix l'estructura de la base de dades de la Figura 44, per a cada grup del conductor i per a cada usuari del grup (doble bucle) guardem les dades en cas que el grup ho autoritzi. Per a tots als mètodes d'actualització de dades, s'incorpora aquest control per poder seguir els requeriments funcionals que ens marcat.

DrivingActivity: aquesta classe del paquet Vista, ens permet establir les dades que es compartiran com: destinació, accepta trucades i cercant aparcament. La dada per saber si un usuari està conduint es canvia automàticament quan es polsa la icona del volant situada a la part superior dreta de la Figura 45, on el color ens indica el seu estat (verd activat, vermell desactivat). En el moment d'activar la conducció es comparteixen totes les dades de cop cridant al mètode *saveDrivingData* de la classe *Controller* des de el mètode *shareData* de la classe *DrivingActivity*. Si es desitja canviar una de les tres dades que hem comentat abans, s'actualitzaran de forma individual cridant al mètode *updateDestination*, *updateAcceptCalls* i *updateParking* de la classe *Controller*. Sempre que es canvia una d'aquestes dades es mostra visualment a l'usuari per saber sempre l'estat de les seves dades. Més endavant (últim increment), es mostrarà l'última imatge presa en aquesta pantalla. D'altra banda, la localització sempre s'especifica la mateixa (ubicació de la Facultat de Matemàtiques) fins que a l'últim increment puguem obtenir-la a partir dels sensors del dispositiu. Un cop es vol desactivar la conducció s'actualitza la dada d'estar conduint automàticament i es guarda a base de dades a partir del mètode *endDriving*.

4.4. Quarta iteració

En l'anterior desenvolupament, vam implementar la compartició de dades des d'un conductor cap a la base de dades. En aquest quart increment completarem el flux de l'aplicació: un conductor comparteix dades, s'envien a la base de dades i un visualitzador les recupera i les mostra.

4.4.1. Requeriments funcionals

- **Visualitzar un conductor**: un usuari ha de poder visualitzar les dades que el conductor està compartint.
- **Mostrar totes les dades disponibles**: l'aplicació ha de permetre visualitzar totes les dades que l'usuari té abast. Com hem comentat a l'increment anterior, pot succeir que un visualitzador estigui a diferents grups i, per tant, les dades a visualitzar han de ser una composició dels 2 grups.
- **Visualització ràpida**: si hi ha un conductor que està conduint, el visualitzador ha de poder veure una previsualització a la pantalla de selecció d'usuari si hi ha algun usuari en conducció.

4.4.2. Disseny de la interfície

Hem de proporcionar a l'usuari a la pantalla de selecció d'usuari que vàrem crear al Segona iteració per introduir una previsualització de si esta conduint un usuari. Per dur a terme aquest element, posarem en verd un volant quan esta conduint i de color vermell en cas contrari (a la Figura 50 es pot veure aquest element distintiu). A més a més, hem de crear una nova pantalla per a poder visualitzar les dades del conductor que vam definir a la fase d'anàlisi. El mapa de la localització ens aportarà el marcador de la localització actual i el tràfic de la zona. Finalment hem de poder visualitzar les imatges que s'han realitzat durant la conducció, de moment aquesta funcionalitat no esta implementada fins el següent increment, però disposarem de dades estàtiques.

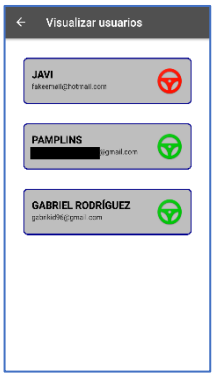

Figura 50: pantalla de selecció d'usuari per a veure les dades que comparteix.

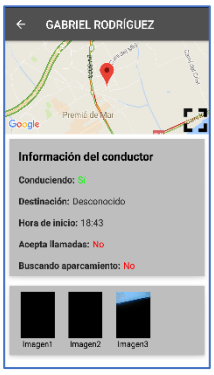

Figura 51: visualització d'un conductor.

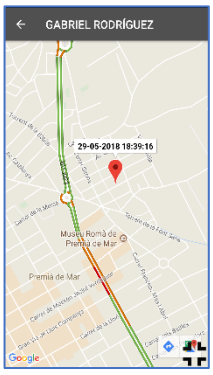

Figura 52: localització ampliada, millora la visualització.

4.4.3. Diagrama de classes

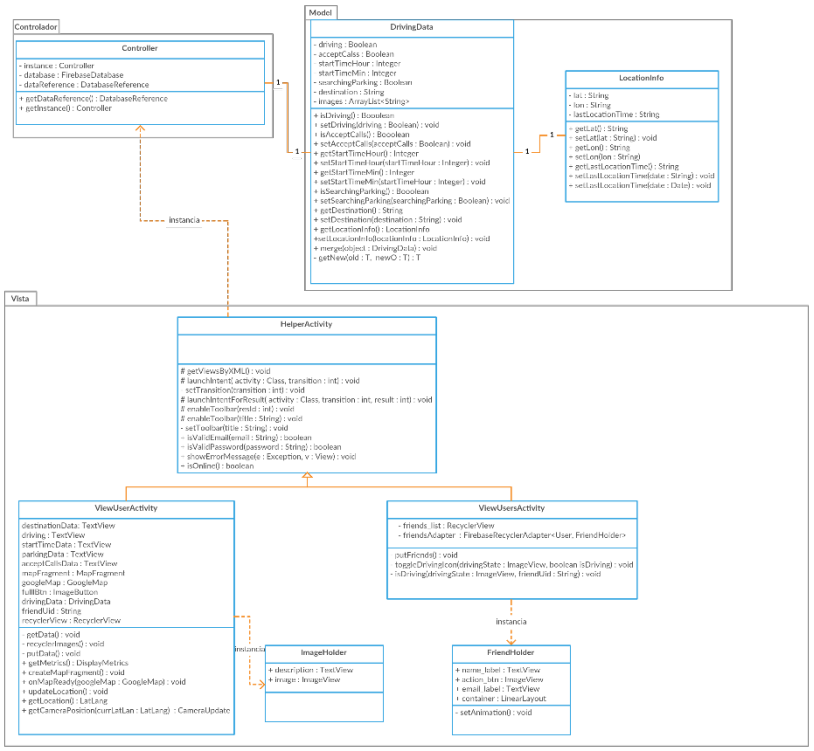

Figura 53: diagrama de classes del quart increment.

4.4.4. Implementació

Seguint la implementació de l'increment anterior, treballarem sobre la pantalla de selecció d'usuaris i la pantalla de visualització de dades d'un conductor.

ViewUsersActivity: recuperem la classe que vàrem fer al Segona iteració. El mètode *putFriends*, mostrem els usuaris que ens han donat permís per a veure dades. Per això es necessari utilitzar un *FirebaseRecyclerAdapter*, tal com vàrem utilitzar a la classe *GroupsActivity*. Per a mostrar la informació de cada usuari creem una instància de *FriendHolder* que conté el nom del conductor, el correu electrònic i la icona del volant que indica l'estat de la conducció.

D'altra banda, hem de seleccionar el color adient de la icona del volant de cada usuari (ho vam deixar pendent al Segona iteració), vermell si no esta conduint, verd en cas contrari. El mètode que determina de quin color serà el volant es *isDriving*, on rep per paràmetre la icona a modificar i la referencia a la base de dades del conductor. Amb la referencia a la base de dades es cerca la dada "usuari conduint" entre els nodes (nodes comentats a la Figura 44) que contenen la dada d'aquest usuari. Quan es troba la primera dada en estat a *true*, la icona passarà a ser verda mitjançant el mètode *toggleDriving*.

ViewUserActivity: la nova classe recollirà les dades de l'usuari seleccionat a la pantalla prèvia (passarem la referència a través de l'Intent). Per començar hem de recollir les dades a partir de la referència i buscarem totes aquelles dades que pertanyin a l'usuari visualitzador. Per això, hem de fer una cerca a la base de dades a partir de la referència de l'usuari conductor i la referència de l'usuari visualitzador. Com es pot veure a la Figura 54, hem remarcat en vermell la referència del conductor en vermell (les dades de qui volem visualitzar, passades com a argument de la pantalla anterior) i en verd la referencia de l'usuari visualitzador. Al ser el node fulla d'aquest fragment de la base de dades ens retornaria un o més objecte del tipus *DrivingData*. El mètode que ens aconsegueix tot això s'anomena *getData*.

Figura 54: dades reals a la base de dades.

Com podem rebre diversos objectes de tipus *DrivingData* amb més o menys informació entre si, hem de realitzar una composició de tots per a obtenir un objecte *DrivingData* que ens aporti les màximes dades per part del conductor. El mètode *merge* de la classe *DrivingData* ens permet fer composicions de parelles d'aquests objectes. Dins d'aquest mètode s'avalua cada camp de la classe amb el mètode *getNew* on selecciona el valor que aporta més informació de cada camp d'entre els dos objectes.

Un cop tenim les dades, ens queda mostrar-les per pantalla (es pot visualitzar a partir de la Figura 51). En primer lloc tenim la localització, on utilitzarem un mapa i les seves funcionalitats facilitat per la llibreria GMS.MAPS (MapFragment, 2018) de Google, el mètode *createMapFragment* ens inicialitzarà la visualització del mapa. Per a poder col·locar la localització al mapa farem us del mètode *updateLocation* on s'indicarà la posició actual del conductor a partir d'un marcador i l'hora d'aquesta localització. Per a poder veure el tràfic de la posició actual i els voltants ens caldrà activar l'opció d'aquesta manera *googleMap.setTrafficEnabled(true),* aquest mètode juntament amb el paràmetre *true* ens pintarà sobre el mapa les dades del tràfic. En segon lloc, tenim la resta de dades (exceptuant les imatges) en un bloc a la part inferior del mapa. En cas que no es tingui constància del valor d'una dada s'informarà amb el missatge "Desconegut". Finalment, tenim un bloc al final de la pantalla que mostra les imatges que s'han capturat durant la conducció i es visualitzen amb el mètode *recyclerImages*. Aquest mètode fa us de l'objecte *recyclerView* on s'utilitza com a contenidor de tots els

objectes *ImageHolder*. Aquesta classe conté la imatge que es podrà seleccionar en qualsevol moment per a visualitzar-la més còmodament fent ús de tota la pantalla del dispositiu. Per a realitzar aquesta última acció hem optat per utilitzar la llibreria *AndroidImagePopup* (AndroidImagePopup, 2018) compatible amb la llibreria *Glide* (Glide) recomanada per Google (Blog, 2018) que ens permet descarregar-nos la imatge des de el servidor (a l'últim increment es realitzarà, però implementem el codi en aquest increment).

Finalment, totes aquestes dades s'actualitzaran en temps real sense la necessitat de recarregar la pantalla gràcies a la implementació del mètode *getData* comentat anteriorment. Per aconseguir les dades es crida al mètode *addValueEventListener* i un cop té les dades ens les retorna en una classe anònima en el mètode *onDataChange*. Aquest últim mètode es cridarà al canviar qualsevol dada i aprofitarem per a cridar al mètode *putData* que ens mostra les dades per pantalla.

```
private void getData(String friendUid){
    drivingData = new DrivingData();
    controller.dataReference.child(friendUid).addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            try{
                for (DataSnapshot groupSnapshot : dataSnapshot.getChildren()) {
                    for (DataSnapshot drivingDataSnapshot : groupSnapshot.getChildren()){
                        if (drivingDataSnapshot.getKey().equals(FirebaseAuth.getInstance().getUid())){
                            drivingData.merge(drivingDataSnapshot.getValue(DrivingData.class));
                        }
                    }
                }
            }catch (Exception ex){
                Log.e("VIEW", ex.getMessage());
            }finally {
                putData(drivingData);
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
    });
}
```

Figura 55: mètode getData de la classe ViewUserActivity que ens permet recollir les dades per primer cop i deixar un escoltador per quan canvien.

4.5. Cinquena iteració

En aquest últim desenvolupament realitzarem allò que ens queda per a completar l'aplicació, l'enviament de la localització i les imatges, tot això en temps real.

4.5.1. Requeriments funcionals

- **Actualització periòdica de la ubicació**: hem de poder compartir la localització cada cert temps (configurat a partir de les preferències de l'usuari). Un cop compartida l'usuari visualitzador ha de poder veure-la a la pantalla de visualització.
- **Capturar i compartir imatges de forma periòdica**: l'aplicació ha de capturar imatges de forma transparent a l'usuari (sense mostrar el capturador de la càmera). Les imatges es realitzaran a partir de la càmera posterior del dispositiu i destinada a ser col·locada en una posició que permeti capturar la visió del conductor.
- **Compartir dades de forma no bloquejant**: com el propòsit de l'aplicació és enviar dades en el moment de la conducció, el conductor ha d'activar l'enviament de dades però després pot bloquejar el terminal perquè no l'utilitzarà però desitja que se segueixin compartint les dades. Per aquesta raó hem d'utilitzar un servei que no s'aturi fins que l'usuari ho especifiqui.

4.5.2. Disseny de la interfície

Per aquest desenvolupament no realitzarem cap canvi al disseny, ja que ho hem deixat preparat als increments anteriors. Únicament proporcionarem dades a aquells elements que estaven amb dades fixes, com per exemple el mapa que es pot veure a la Figura 58 juntament amb el bloc d'imatges que es pot visualitzar a la part inferior de la figura. També farem us de l'element de color gris que es pot veure a la Figura 56 just al costat de l'indicador d'acceptar trucades on mostrarem l'última imatge capturada. Finalment a la Figura 58 utilitzarem les preferències de l'usuari de la sincronització de dades que ens serviran per a definir la periodicitat per actualitzar la ubicació i la realització de fotografies.

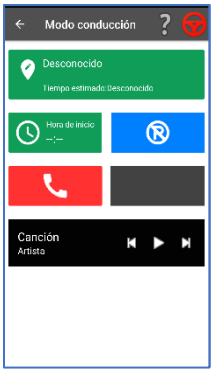

Figura 56: mode conducció.

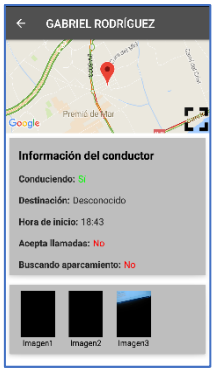

Figura 57: pantalla de visualització.

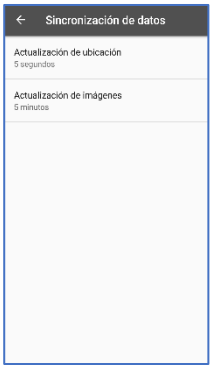

Figura 58: preferències del sistema per a la sincronització de dades.

4.5.3. Diagrama de classes

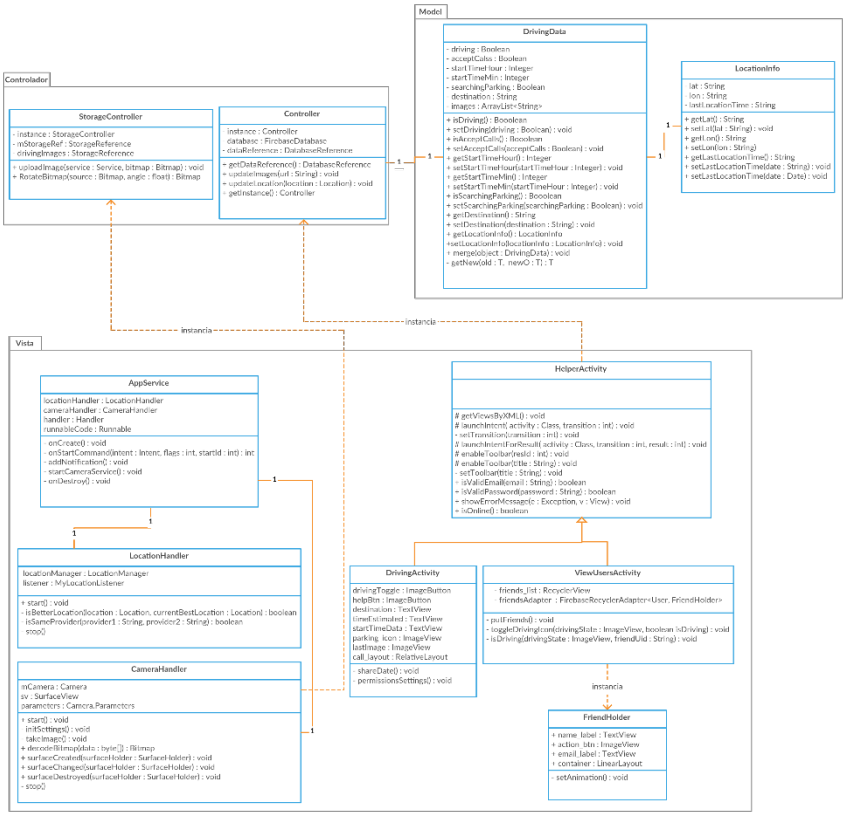

Figura 59: diagrama de classes que intervenen al cinquè increment.

4.5.4. Implementació

DrivingActivity: Per a poder realitzar la implementació de la ubicació i la realització de fotografies necessitem el consentiment de l'usuari per a poder compartir aquestes dades. Per aquesta raó s'incorpora el mètode *permissionsSettings* on pregunta a l'usuari aquests 3 permisos (si no els té ja l'aplicació):

1. Ubicació
2. Realitzar fotografies
3. Escriure per sobre d'altres aplicacions

Aquest últim permís el necessitem per a poder realitzar fotografies sense mostrar el capturador, és a dir, sense que l'usuari es percebi, és per això que necessitem el seu consentiment. En cas que l'usuari no ens atorgui el consentiment no compartirem tot allò que no ens hagi permès.

En el mètode *shareData* que vàrem comentar al Tercera iteració incorpora la crida *startService* que ens permetrà activar el servei *AppService* que anirà obtenint la ubicació i les imatges de forma paral·lela a l'aplicació

AppService: com hem comentat el servei ha d'executar-se de forma paral·lela a l'aplicació i per a fer-ho possible hem de realitzar una classe filla de la classe Service. Això ens permet realitzar accions no bloquejants fins i tot quant l'aplicació és aturada. Per indicar a l'usuari d'aquestes accions hem implementat un mètode *addNotification* on informa l'usuari que s'està compartint dades en segon pla i li permet finalitzar-lo. D'altra banda, s'instancien (si tenim els permisos) al mètode *onStartCommand* els

encarregats d'obtenir la ubicació (classe *LocationHandler*) i la realització de fotografies (classe *CameraHandler*). La primera classe comentada s'iniciarà i no s'aturarà fins que l'usuari ho indiqui. Al mateix temps la classe *CameraHandler* s'iniciarà cada cert temps (l'indicat per les preferències de l'aplicació) gràcies al mètode *startCameraService* i s'aturarà completament quan l'usuari ho indiqui.

LocationHandler: classe que s'encarrega d'obtenir les ubicacions del dispositiu. En el moment de començar a capturar ubicacions necessitem un interval de temps per a poder capturar la localització mitjançant la classe *LocationManager* d'Android. Aquest interval l'obtindrem de les preferències de l'usuari. Un cop tenim l'interval hem d'especificar una classe que capturarà els canvis d'ubicació amb els proveïdors disponibles, sensor GPS ajudat de la connexió a Internet si existeix (tot això es realitza en el mètode *start*). Per això hem d'incorporar una classe interna anomenada *MyLocationListener* que implementa el mètode *onLocationChanged* per a obtenir la ubicació. Un cop capturada la ubicació ens quedarà actualitzar-la de la mateixa manera que ho fem amb la resta de dades, amb un mètode d'actualització a la classe *Controller* anomenat *updateLocation*. Finalment l'usuari visualitzador podrà veure aquestes actualitzacions en temps real al mapa situat a la pantalla de visualització.

```
public void start(){
    locationManager = (LocationManager) service.getSystemService(Context.LOCATION_SERVICE);
    listener = new MyLocationListener();

    String time = PreferenceManager.getDefaultSharedPreferences(service.getApplicationContext())
            .getString("location_sync", "");
    int interval;
    try{
        interval = Integer.parseInt(time) * 1000;//seconds
    }catch (NumberFormatException ex){
        interval = Constants.DEFAULT_TIME_LOCATION;
    }
    locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, interval, 0, listener);
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, interval, 0, listener);
}
```

Figura 60: mètode start de la classe LocationHanlder que agafa l'interval de les preferències d'usuari i activa els proveïdors disponibles per a que la classe MyLocationListener capturi els canvis d'ubicació.

CameraHandler: classe que s'encarrega de realitzar una fotografia cada cop que s'instancia i es crida al mètode *start*. Quan s'instancia aquesta classe es crea una *SurfaceView* invisible que contindrà el capturador de la càmera. Al ser invisible, per a l'usuari semblarà que les imatges es treuen en segon pla, per aquesta raó necessitàvem el permís d'escriure sobre altres aplicacions. Quan aquesta *SurfaceView* és creada es crida al mètode *takeImage* que s'encarrega d'obrir i capturar la imatge en segon pla. Un cop tenim la imatge capturada ens arribarà en forma d'una llista de *bytes* que haurem de descodificar (mètode *decodeBitmap*) per obtenir la imatge com a mapa de bits, *Bitmap*. En aquest moment ja tenim la imatge i utilitzarem la classe *StorageController* per a pujar la imatge a la carpeta d'usuari que tenim a *Firebase* i com les altres dades cridarem al mètode *updateImages* de la classe *Controller* per actualitzar la llista d'imatges perquè els visualitzadors puguin veure-les.

StorageController: classe controladora per a pujar arxius al magatzem d'arxius que tenim amb la tecnologia *Firebase*. Aquests arxius es guarden i ens retornen un URL de descàrrega que es guardarà a la base de dades mitjançant l'atribut *images* de la classe

DrivingData. El mètode per a pujar imatges s'anomena *uploadImage* i un cop pujada la imatge cridem al mètode *updateImages* amb l'URL de la imatge pujada per paràmetre.

4.6. Patrons de disseny utilitzats

Per al desenvolupament de l'aplicació s'han utilitzat uns patrons de disseny concrets per a poder aportar eficiència i qualitat al codi. A continuació es presenten.

4.6.1. MVC

El patró MVC consisteix a estructurar el codi en tres grans blocs: Model, Vista i Controlador.

Model: és el conjunt de classes que representen una sèrie de dades. Per a poder emmagatzemar i recuperar dades de la base de dades hem d'especificar quina classe conté aquestes dades. Podem trobar les següents classes:

- *User*
- *Group*
- *DrivingData*
- *LocationInfo*

En cas de voler emmagatzemar dades, des del bloc Controlador haurem d'instanciar de forma explícita les classes primerament i després guardar-les a base de dades. En cas de voler recuperar, també des del bloc Controlador però sense crear la instància explícitament, ja que la llibreria de Firebase ens transforma les dades de la base de dades al model especificat de la següent manera:

User currentUser = dataSnapshot.getValue(User.class);

on *dataSnapshot* és una referència a la base de dades.

Controlador: el controlador és el bloc que comunica el Model amb la Vista. El controlador s'encarregarà d'emmagatzemar les dades de l'usuari actual i les seves referències a base de dades. També realitzarà tota acció amb la base de dades (lectura i escriptura). Només existirà una instància de la classe *Controller*, per tant, implica utilitzar un patró *Singleton*.

Vista: la vista és el bloc visible a l'usuari. S'encarrega de mostrar les pantalles a l'usuari amb la informació provinent del controlador. També s'encarrega d'enviar informació de manera correcta, implica tenir una lògica que comprova les dades introduïdes per l'usuari.

4.6.2. Singleton

Per poder tenir una instància de la classe controladora necessitem aquest patró que ens soluciona aquesta necessitat. El patró ens proporciona un punt d'accés global a la nostra classe controladora, *Controller* i *StorageController* en el nostre cas, i evitar múltiples instàncies. Per realitzar aquest patró és necessari posar com a privat el constructor de la classe *Controller i StorageController* per evitar múltiples instàncies i crearem un mètode estàtic, *getInstance*, que serà el punt d'accés global. Cal dir que les nostres classes controladores començaran a existir com a objecte quan l'usuari hagi iniciat sessió.

```
public static Controller getInstance() {
    if(instance == null && FirebaseAuth.getInstance().getCurrentUser() != null) {
        instance = new Controller();
    }
    return instance;
}
```

Figura 61: exemple de punt d'accés a una classe controladora. Si la instancia de la pròpia classe es nul·la i l'usuari ha iniciat sessió creem per primera i última vegada la classe.

El patró *Singleton* també s'utilitza a les llibreries de Firebase, a la Figura 61 es pot veure com s'obté la instància de la classe *FirebaseAuth*, a través del mètode *getInstance*. De la mateixa manera succeeix amb les classes *FirebaseDatabase* i *FirebaseStorage*.

4.7. Altres implementacions

4.7.1. Persistència de dades sense connexió.

A causa que l'aplicació té com a requeriment tenir connexió a internet per a poder realitzar qualsevol acció hem introduït una persistència de les dades. Gràcies a Firebase, podem mantenir sincronitzada una referència de la base de dades, és a dir, tot allò que descarreguem quan tenim connexió s'emmagatzema a la *cache* del dispositiu i l'usuari en cas de no tindre connexió li brindarà les últimes dades rebudes. D'altra banda, succeeix el mateix per guardar dades, el dispositiu quan realitza una acció que requereix guardar a base de dades l'aplicació funcionarà de la mateixa manera, sense esperes ni errors de connexió, i per l'usuari serà totalment transparent.

Per exemple, quan un usuari està conduint, entra a un túnel i perd la connexió el dispositiu segueix enviant dades però sense que la base de dades les rebi i encara no està disponible per als usuaris visualitzadors. Un cop ha sortit del túnel i la connexió torna, gràcies a la sincronització, s'envien totes aquelles dades que no es varen poder enviar i no es perdran en cap cas.

Per poder realitzar aquesta funcionalitat, cal fer aquesta crida inicialment:

FirebaseDatabase.getInstance().setPersistenceEnabled(true);

serveix per a informar a la base de dades que activem la persistència de dades sense connexió. Un cop ho tenim activat, podem mantenir sincronitzada les referències que desitgem. A l'aplicació mantindrem sincronitzada les dades de conducció i les dades de l'usuari mitjançant el mètode *keepSync* a la referència desitjada.

```
userGroupsReference.keepSynced(true);
userFriendsReference.keepSynced(true);
dataReference.keepSynced(true);
```

Figura 62: sincronització dels grups d'un usuari, els usuaris que podem visualitzar i les dades a compartir de la conducció.

4.7.2. Tractament d'errors amb la base de dades

Quan es realitzen accions a la base de dades que poden provocar errors i cal notificar a l'usuari estan controlats de la següent forma:

```
}).addOnFailureListener(this, new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        switchViews(true);
        showErrorMessage(e, emailInput);
    }
});
```

Figura 63: exemple per a capturar l'error de base de dades.

S'afegeix una classe anònima en el mètode *addOnFailureListener* de la crida a base de dades per a capturar l'error. El mètode *showErrorMessage* rep l'error per paràmetre i mostra un missatge específic de l'error.

```
public void showErrorMessage(@NonNull Exception e, View v){
    String message = e.getLocalizedMessage();
    if (e instanceof FirebaseAuthInvalidCredentialsException) {
        switch (((FirebaseAuthInvalidCredentialsException) e).getErrorCode()){
            case "ERROR_WRONG_PASSWORD":
                message = getResources().getString(R.string.error_message_password);
                break;
        }

    } else if (e instanceof FirebaseAuthInvalidUserException) {
        switch (((FirebaseAuthInvalidUserException) e).getErrorCode()){
            case "ERROR_USER_NOT_FOUND":
                message = getResources().getString(R.string.fui_error_email_does_not_exist);
                break;
            case "ERROR_USER_DISABLED":
                message = getResources().getString(R.string.error_message_disabled);
                break;
            case "ERROR_EMAIL_ALREADY_IN_USE":
                message = getResources().getString(R.string.error_message_email_in_use);
                break;
        }
    }else if (e instanceof FirebaseNetworkException){
        message = getResources().getString(R.string.no_connection);
    }

    Snackbar.make(v, message,Snackbar.LENGTH_SHORT).show();
}
```

Figura 64: mètode showErrorMessage on es pot veure tots els errors que poden succeir.

4.7.3. Notificacions

Quan l'aplicació s'atura s'activa un servei que s'executa en segon pla al dispositiu (si l'usuari mitjançant les preferències de l'aplicació ens ho permet) que comprova si un usuari ha començat a conduir. Aquest servei s'executa de la mateixa manera que el servei comentat en Cinquena iteració i només s'envia una notificació en cas que un usuari que anteriorment no estava conduint ha començat a conduir en aquell moment. Aquesta restricció limita el nombre de notificacions i no genera una molèstia a l'usuari.

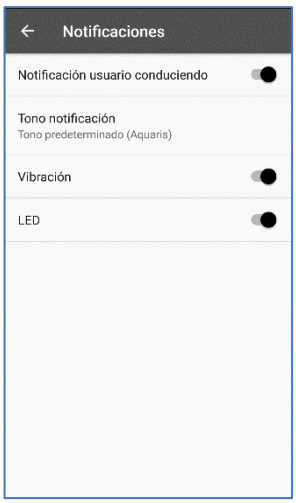

Figura 65: pantalla de preferències de notificacions. Només s'enviaran notificacions si l'usuari ens ho permet.

Figura 66: exemple de notificació.

Com es pot veure a la Figura 66, l'usuari ha rebut una notificació i li permet dirigir-se directament a la pantalla de visualització en cas de tocar-la.

4.7.4. Meteorologia

Per proporcionar més dades al visualitzador, encara que no és una dada prioritària a mostrar, s'incorpora la informació meteorològica a la pantalla de visualització d'un conductor. Per a mostrar aquesta informació és basem a l'última localització coneguda del conductor. Un cop tenim la localització mostrarem les dades que ens proporciona l'API de *OpenWeatherMap* (OpenWeatherMap, 2018) on a partir d'una petició *HTTP* amb els paràmetres adients ens retorna les dades meteorològiques en format *JSON* que obtindrem les dades que es poden veure a la Figura 67. Com es un servei gratuït tenim una limitació de 60 peticions per minut.

Figura 67: informació meteorològica de la posició del conductor.

A la petició HTTP es demana la informació meteorològica a partir d'aquests paràmetres:

- Latitud (lat): latitud de la posició del conductor.
- Longitud (lon): longitud de la posició del conductor.
- Unitats (units): unitats de les dades, indicarem el sistema mètric.
- Identificador d'aplicació (appId): clau generada per *OpenWeatherMap* per controlar el nombre de consultes
- Idioma (lang): idioma de la informació meteorològica. Agafarem l'idioma del dispositiu.

Aquests paràmetres generen una petició *HTTP* que ens retorna un objecte *JSON* que extreure'm les següents dades:

- Descripció: descripció del temps, per exemple assolellat.
- Temperatura: temperatura actual en l'ambient.
- Identificador de la icona: identificador de la icona a mostrar. Per a aconseguir la icona és necessari descarregar-la a través de la *URL* *http://openweathermap.org/img/w/* seguida de l'identificador de la icona.

5. Resultats i avaluació

A continuació, posarem a prova l'aplicació a usuaris reals on hauran de sotmetre's a una sèrie d'instruccions per poder observar si les realitzen amb èxit. Tornarem a contactar amb els usuaris seleccionats de la secció Indagació contextual i d'altres per realitzar proves amb el màxim d'usuaris possibles. Un cop tenim els usuaris, situarem a aquests en un entorn per a donar realisme a les observacions. En el moment de situar als usuaris en l'entorn indicat els demanarem que realitzin una sèrie d'accions i com a moderadors puntuarem la facilitat de realitzar la tasca (0 cap dificultat– 5 molta dificultat) per a poder extreure conclusions. Cal dir que una tasca no assolida la considerarem de dificultat màxima amb temps màxim.

Per dur a terme les observacions, disposarem de dos dispositius mòbils amb l'aplicació ja instal·lada i iniciada la sessió. Un dispositiu l'utilitzarem nosaltres com a observador i un altre per a l'usuari observat. El dispositiu de l'observador té la funció d'activar el mode conducció quan l'usuari observat realitzi el rol de visualitzador. Quan l'usuari es comporti com a conductor, el nostre dispositiu servirà com a usuari per afegir a les tasques que requereixi afegir un usuari a un grup per a compartir dades.

5.1. Entorn 1: trajecte amb transport públic

Utilitzarem el transport públic per a simular un entorn de conducció en l'observador. L'objectiu és avaluar totes les funcionalitats de l'aplicació on l'usuari actuarà com a conductor i visualitzador. Quan l'usuari actuï com a visualitzador, visualitzarà al moderador de l'observació, on prèviament haurem configurat un grup perquè l'usuari observat pugui visualitzar totes les nostres dades.

Tasques

1. Crear un grup anomenat "Família", afegeix com a mínim un usuari i permet compartir totes les dades.
2. Elimina aquest grup i desfés l'acció.
3. Configura l'aplicació perquè comparteixi la ubicació cada 10 segons i de 5 minuts per realitzar imatges.
4. Determina un destí.

5. Activa el mode conducció indicant que acceptes trucades.
6. Desactiva el mode conducció al cap de 5 minuts.
7. Visualitza a un usuari que estigui conduint i un altre que no ho estigui.
8. De l'usuari que està conduint, indica l'hora i la meteorologia de l'última localització.
9. Visualitza l'última imatge.

5.2.Entorn 2: circulació en vehicle privat

Per situar a l'usuari en un context real, contactarem amb usuaris que estiguin disposats a circular i que posseeixin de permís de conduir, per veure com es comporta les funcionalitats de l'aplicació. Per a no posar en perill la integritat del conductor, qui realment interaccionarà amb l'aplicació serem nosaltres, els observadors, durant la conducció. Abans d'iniciar la circulació, l'usuari observat realitzarà les tasques relacionades (tasques 1 a 6) amb el rol de conductor que vàrem realitzar a l'entorn anterior.

L'objectiu d'aquestes observacions és veure com interacciona el conductor amb l'aplicació mitjançant les comandes per veu. Com encara no han sigut desenvolupades, nosaltres com a moderadors, realitzarem tota acció que ens demani l'usuari a fi d'evitar situacions de perill. En el diagrama de la Figura 68 es pot veure quines accions ens pot demanar el conductor mentre esta conduint.

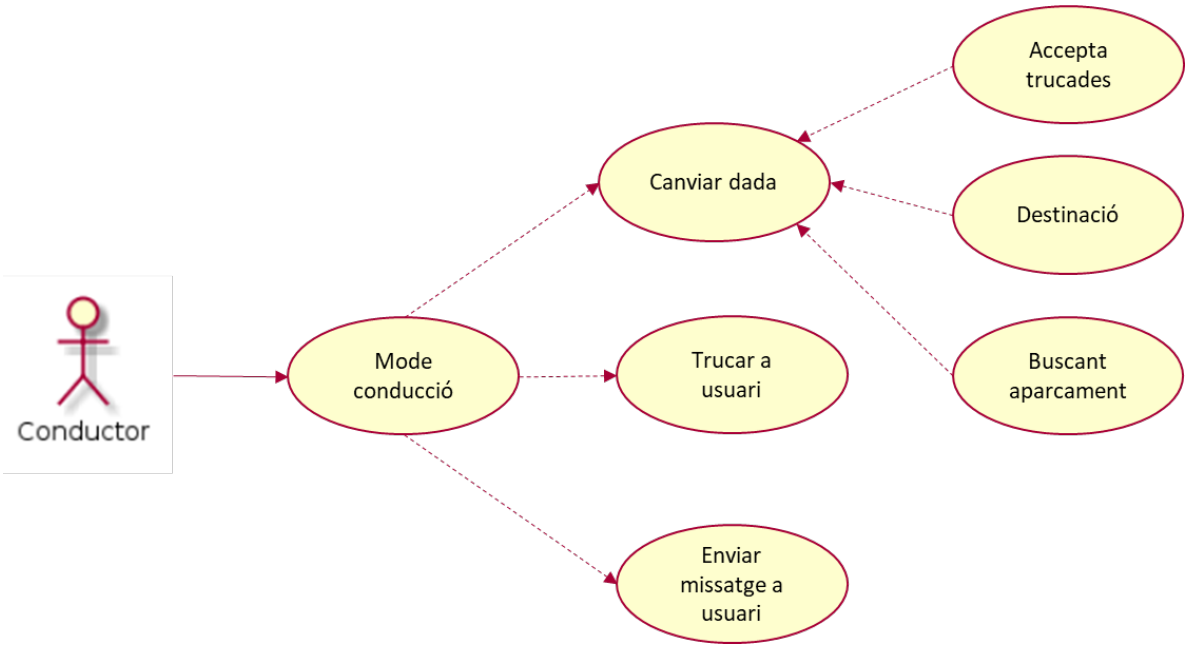

Figura 68: diagrama de les accions que pot realitzar el conductor mentre condueix.

Com a segon objectiu, volem observar com es comporta la funcionalitat de capturar imatges de forma periòdica. Necessitarem col·locar el dispositiu a l'interior del vehicle de forma que es pugui visualitzar aproximadament el que veu el conductor. Per realitzar aquesta prova, dotarem al vehicle d'un suport per a dispositius mòbils en el quadre de comandament.

5.3.Resultats

Primerament, l'aplicació ha estat implementada per a l'API 16, és a dir, compatible per a dispositius amb versió 4.1.2 o superior, actualment disponible al 99.3% dels dispositius Android (Versions de la plataforma Android, 2018). Per aquesta raó, s'han utilitzat dispositius amb versió 4.1.2 o superior. Concretament, els dos dispositius per a realitzar les proves pertanyen a les versions 7.1.2 (API 24) i 8.1 (API 26).

A continuació es presenten els resultats obtinguts de les observacions en els 2 entorns que hem descrit anteriorment. En total s'han realitzat 9 observacions, 6 en el primer entorn i 3 en el segon entorn. Tots els usuaris han donat el seu consentiment verbal després d'explicar els objectius de l'observació i de les tasques a realitzar.

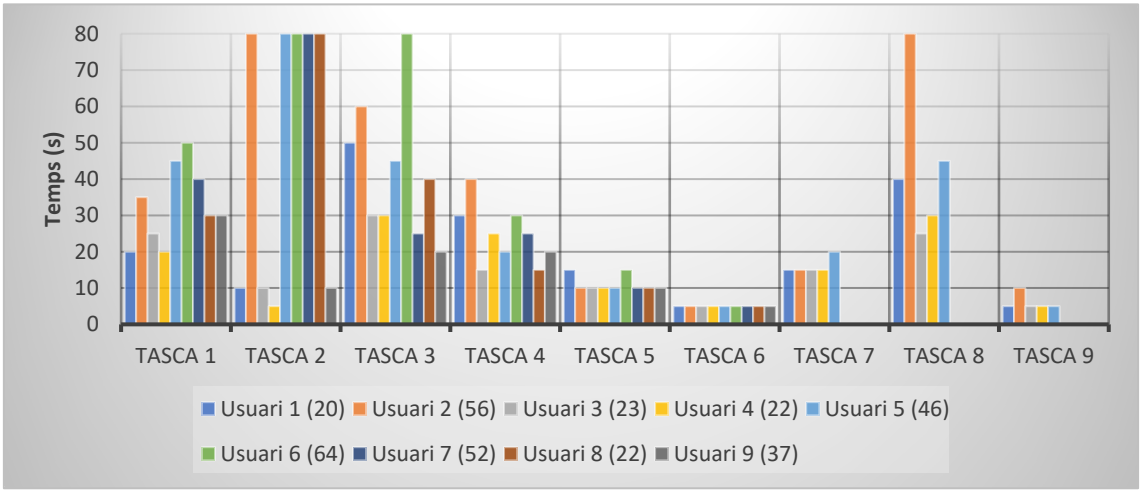

Figura 69: gràfic dels resultats obtinguts classificades per tasca i usuari. Entre parèntesis l'edat de l'usuari observat.

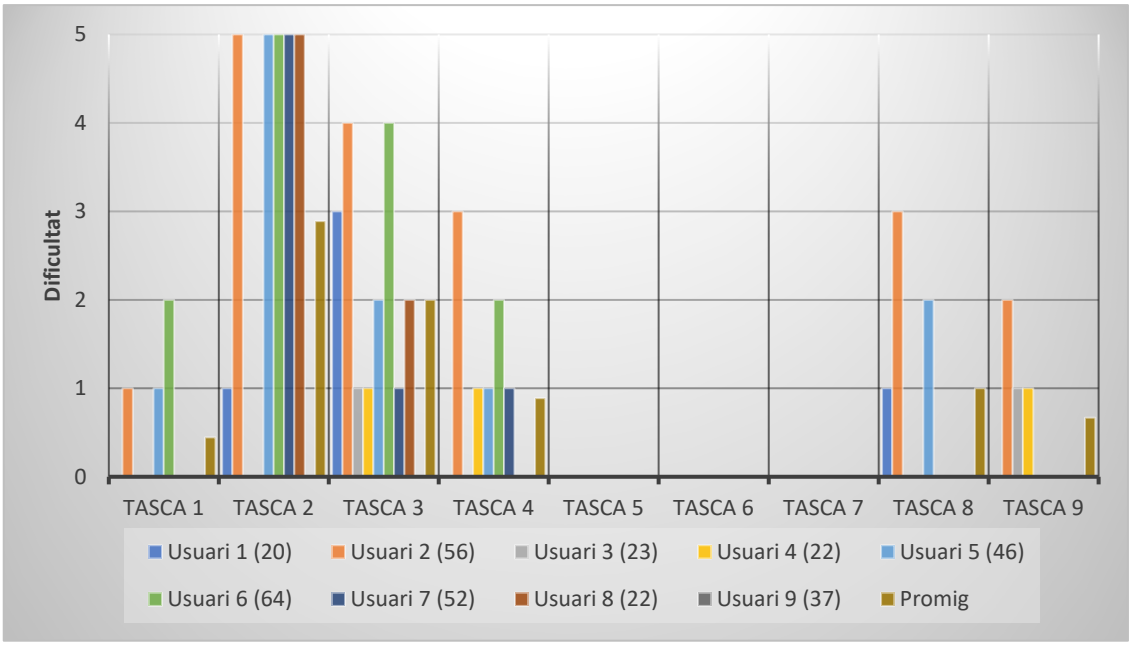

Figura 70: gràfic de les dificultat per realitzar les tasques per part dels usuaris. Entre parèntesis l'edat de l'usuari observat.

Com es pot veure a la Figura 70, la dificultat mitjana de les tasques està al voltant del 1 sobre 5. Majoritàriament, els usuaris han superat les tasques proposades de forma correcta, però altres

han costat més de realitzar o han sorgit comentaris per part dels usuaris desaprovant les funcionalitats.

En primer lloc, tots els usuaris amb més de 50 anys no han pogut completar l'acció número 2. Aquesta tasca demanava esborrar un grup i, com si ens haguéssim equivocat, desfer aquesta acció. Per poder-ho realitzar, en eliminar un grup ens apareix un missatge durant uns segons, com es pot veure a la Figura 71) que ens comunica si volem desfer aquesta acció. Nosaltres com a moderadors, només vam comunicar a l'usuari l'acció, sense comunicar-li que l'hauria de fer amb rapidesa, però conseqüentment no els hi donava temps de llegir el missatge i reaccionar per desfer l'acció. Altres usuaris esperaven que aparegués un diàleg de confirmació al centre de la pantalla (a la mateixa altura del botó d'esborrar) que els hi permetés tornar enrere per desfer l'acció.

En segon lloc, veiem a la Figura 69 com la tasca número 3 ha suposat pels usuaris una inversió de temps moderada. La tasca demanava definir una periodicitat per actualitzar la localització i la captura d'imatges. Els usuaris havien de dirigir-se a la pantalla de configuració i accedir a "Sincronització de dades", és a dir, a una distància de navegació 2 *clicks*. Encara que tots els usuaris han assolit realitzar-la, aquesta distància ha sigut la causa de la demora per realitzar l'acció, on tots els usuaris han trigat més que en la resta de tasques.

En tercer lloc, un parell d'usuaris han trigat més que la resta d'usuaris en indicar la destinació. Han comentat que els ha faltat un indicador per saber introduir la destinació.

En quart lloc, les tasques 8 i 9 han sorgit dificultat a causa del rendiment de l'aplicació. Per part de la tasca 8, no carregava la icona (tal com es pot veure a la Figura 72) que representa el temps meteorològic a causa de la connexió a Internet. Quan un usuari ho tornava a intentar, finalment sortia la icona, ja que havia tingut més temps per descarregar la imatge del proveïdor web. D'altra banda, la tasca 9 generava malestar als usuaris en visualitzar l'última imatge, ja que havien de lliscar les imatges per situar-se a l'última. De vegades, just en aquest moment la pantalla rebia noves dades de l'usuari que estava conduint, i en actualitzar-les, el contenidor de les imatges tornava a la posició de la primera imatge abans de poder visualitzar-la, generant així que l'usuari hagués de tornar a lliscar una altra vegada per visualitzar l'última d'aquestes.

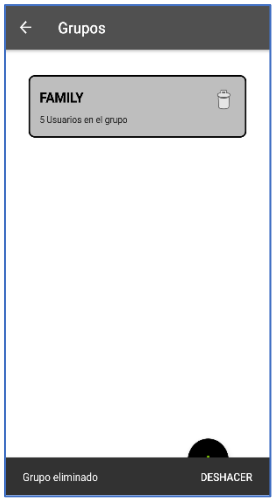

Figura 71: quan s'elimina un grup apareix aquesta barra inferior durant uns segons per desfer l'acció.

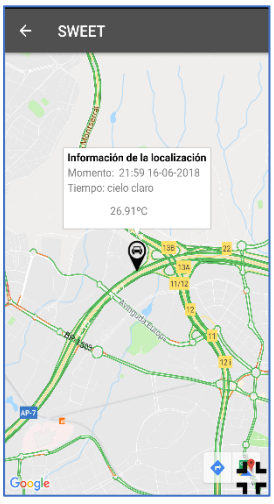

Figura 72: exemple quan no carrega correctament la icona de la meteorologia.

Finalment, les proves al segon entorn ens han confirmat que el control per veu és necessari per actualitzar aquelles dades dinàmiques en la conducció, ja que no és viable, ètic ni segur actualitzar-les interaccionant físicament amb el dispositiu. D'altra banda, els usuaris ens han manifestat que per ells no suposaria un impediment canviar les dades de forma física en moments d'aturada prolongada. També ens ha servit per a veure quins resultats ens donava la realització automàtica d'imatges. Com es pot veure en la Figura 73, les imatges on la il·luminació ve de front es pot observar que hi ha molta claredat i impedeix veure correctament les imatges. En el cas contrari, on la il·luminació es prové de la part posterior del vehicle les imatges es visualitzen correctament (Figura 74). D'altra banda, en els túnels al no tenir tanta il·luminació intensa les imatges es realitzen força nítides, com es pot veure a la Figura 75.

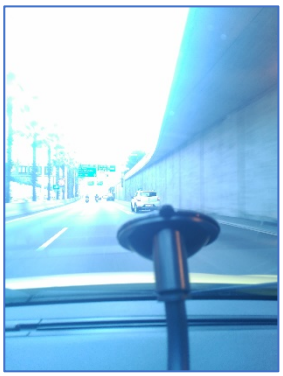

Figura 73: imatge capturada automàticament, amb molta il·luminació.

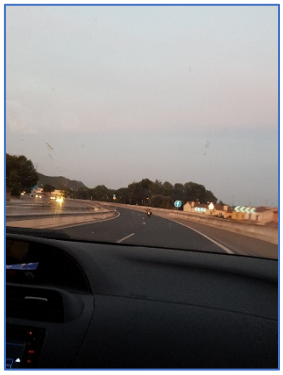

Figura 74: imatge capturada automàticament amb bona qualitat.

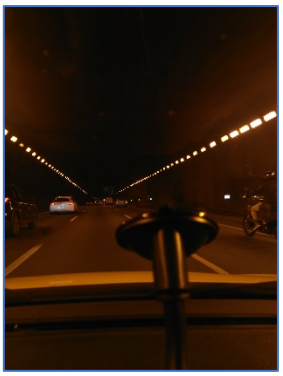

Figura 75: imatge capturada automàticament en un túnel.

6. Discussió

Un cop acabada totes les fases del desenvolupament del software, analitzarem com han sigut les diferents etapes.

En primer lloc, per a realitzar qualsevol projecte de desenvolupament de software hem d'analitzar quins requeriments tenim i si els podem satisfer en el temps establert. L'assignatura Enginyeria del Software ens diu que necessitem una manera de desenvolupar eficientment mitjançant una metodologia. En el nostre cas hem seguit la metodologia incremental, on a cada iteració teníem un producte totalment funcional. Encara que queden funcionalitats pendents l'aplicació realitza l'objectiu que ens vam marcar: compartir dades entre un usuari conductor i un visualitzador. Creiem que és una metodologia encertada en aquest projecte, ja que en cas de no arribar a l'entrega final amb totes les funcionalitats tenim una versió estable de l'aplicació. Cal dir, que aquesta metodologia requereix una planificació inicial prèvia de la feina a realitzar, de la qual no hem pogut complir en la seva totalitat. Per a futurs desenvolupaments hem de pensar una altra manera de planificar la feina de forma més acurada o una alternativa seria fer un canvi de metodologia.

En segon lloc, la fase d'anàlisi ens va permetre pensar molt detingudament com volíem l'aplicació. A partir dels coneixements de l'assignatura Factors Humans i Computació, vam utilitzar les tècniques per a buscar una interfície que sigui usable i adaptada al nombre més gran d'usuaris possible. Les enquestes, on la participació va ser alta, ens van resoldre dubtes, i decidir, sobre les funcionalitats a realitzar, gràcies a les respostes dels usuaris. A més a més, per a definir un primer disseny de l'aplicació les observacions directes van ser encertades, ja que ens van permetre conèixer els defectes i virtuts del prototip conceptual que vàrem dissenyar.

En tercer lloc, com el pla gratuït de Firebase ens donava accés a una base de dades no relacional, creiem que ens ha perjudicat en el moment de desenvolupar. A l'assignatura Bases de Dades ens van ensenyar a treballar amb una base de dades relacional i per aquesta raó hem hagut d'invertir temps a entendre les no relacionals. D'altra banda, en utilitzar a aquest tipus de base de dades tenim coneixements dels avantatges i inconvenients de les bases de dades relacionals i no relacionals. Això ens aporta una presa de decisió més ràpida en un futur, on depenent dels requeriments del projecte tenim els coneixements per saber quina és la millor opció.

Finalment, per realitzar aquesta aplicació requeria una base molt sòlida de programació per fer front a la tecnologia utilitzada. Sense els coneixements de les assignatures Programació 1 i Programació 2, no seria possible entendre i resoldre les funcionalitats definides en la fase d'anàlisi.

7. Conclusions i treball futur

7.1. Conclusions

En primer lloc, cal dir que teníem l'objectiu d'aprendre a programar una aplicació Android i ho hem complert exitosament. Encara que al grau no hem après a programar en Android, hem vist que podem aplicar tots els coneixements de les diferents assignatures del grau en un entorn completament nou per a nosaltres. El més complicat era entendre el flux de funcionament de les classes Android però gràcies a l'aprenentatge inicial i a la documentació molt ben explicada per part de Google hem pogut construir una aplicació robusta i funcional. Encara que amb aquest treball no es pot dir que estem especialitzats en aplicacions Android, podem dir que ens ha aportat molts coneixements que en un futur ens facilitarà realitzar una altra aplicació, ja sigui en el món laboral com en l'àmbit de la investigació.

D'altra banda, hem aconseguit l'objectiu de realitzar una aplicació que permet compartir informació entre un usuari conductor i un usuari visualitzador. Gràcies a les avaluacions que vàrem realitzar a usuaris en entorns reals, vam poder veure el comportament que esperàvem de l'aplicació.

Un aspecte a millorar, és a la planificació. Inicialment teníem la intenció de desenvolupar i documentar al mateix temps, però finalment no hem sigut capaços de complir, tal com es pot veure a la Planificació final.

7.2. Treball futur

Per començar, allò que hauríem d'assolir en un futur és augmentar més quota de mercat per arribar a més usuaris i a més plataformes. Primerament, per completar la quota dels telèfons intel·ligents portaríem aquesta aplicació als dispositius mòbils amb sistemes operatius iOS i Windows. En segon lloc, amb l'auge dels cotxes intel·ligents aquesta aplicació podria ser adaptada i, posteriorment, incorporada de forma nativa en aquests cotxes i no requerir un telèfon intel·ligent per a poder compartir dades.

D'altra banda, per qüestions d'abast no s'han incorporat totes aquelles funcionalitats desitjades. En primer lloc, vam voler que la interacció entre el conductor i l'aplicació fos mitjançant comandes de veu per a no fomentar l'ús del telèfon en la conducció. Aquesta funcionalitat és la més prioritària a realitzar. En segon lloc, tal com vam explicar a la secció 3.5.2, els usuaris van veure amb bons ulls incorporar un sistema d'alerta per comunicar simultàniament a tots els membres dels grups, que el conductor ha patit un accident (d'una magnitud que li permeti realitzar aquesta acció) o una anomalia mentre conduïa. En tercer lloc, un cop l'usuari ha seleccionat el destí, pot ser interessant per l'usuari el temps que trigarà en arribar i anar actualitzant aquesta estimació en funció de la localització i altres factors (tràfic, meteorologia, obres, etc.) que fan canviar el temps estimat.

Una altra millora seria introduir més mètodes d'inici de sessió com Google+, Facebook i Twitter, entre altres. Amb aquesta millora podríem aprofitar per introduir una ajuda a l'usuari per a trobar una manera més còmoda als usuaris a l'hora d'afegir-los a un grup, sense la necessitat de conèixer el correu electrònic.

Finalment, els desenvolupadors hem de considerar sempre les opinions i idees aportades pels usuaris que, al cap i a la fi, són els que utilitzaran l'aplicació i els que necessiten aquest servei, per tant, una vegada l'aplicació estigui disponible per a la comunitat tindrem en compte les valoracions i comentaris dels usuaris que hagin utilitzat l'aplicació.

8. Referencies

Andrew L. Kun, L. N. (2013). AutomotiveUI: Interacting with Technology in Vehicles. *Conferences* (p. 80-82). IEEE CS.

Android. (23 / Juny / 2018). Recollit de https://es.wikipedia.org/wiki/Android

Android Auto. (23 / Juny / 2018). Recollit de Android: https://www.android.com/intl/es_es/auto/

Android Studio. (23 / Juny / 2018). Recollit de Android Studio: https://developer.android.com/studio/

AndroidImagePopup. (23 / Juny / 2018). *AndroidImagePopup*. Recollit de AndroidImagePopup: https://github.com/chathuralakmal/AndroidImagePopup#android-image-popup--

Barcelona, U. d. (23 / Juny / 2018). *Pràctiques en empreses*. Recollit de Pràctiques en empreses: https://mat.ub.edu/matapps/borsa/informacio-practiques-en-empresa/

Bastian Pfleging, S. S. (2013). Exploring User Expectations for Context and Road Video Sharing While Calling and Driving. *AutomotiveUI*, (p. 132-139). Stuttgart, Germany.

Blog, G. O. (23 / Juny / 2018). *Google Open Source Blog*. Recollit de Google Open Source Blog: https://opensource.googleblog.com/2014/09/glide-30-media-management-library-for.html?m=1

El mundo. (23 / Juny / 2018). Recollit de http://www.elmundo.es/motor/2017/03/09/58c1554be5fdea4c708b45e6.html

Firebase. (2018). *Firebase plans*. Recollit de Firebase plans: https://firebase.google.com/pricing/?hl=es-419

Firebase Authentication. (23 / Juny / 2018). Recollit de Firebase Authentication: https://firebase.google.com/docs/auth/?hl=es-419

Gas biker. (23 / Juny / 2018). Recollit de http://gasbiker.com/

Glide. (23 / Juny / 2018). *Glide*. Recollit de Glide: https://github.com/bumptech/glide/wiki

Google Latitude. (23 / Juny / 2018). Recollit de Google Latitude: https://es.wikipedia.org/wiki/Google_Latitude

Google Play. (23 / Juny / 2018). Recollit de Google Play: https://es.wikipedia.org/wiki/Google_Play#Desarrolladores

Heurístiques de Nielsen. (23 / Juny / 2018). Recollit de Heurístiques de Nielsen: https://es.wikipedia.org/wiki/Heurísticas_de_Nielsen

MapFragment. (23 / Juny / 2018). Recollit de MapFragment: https://developers.google.com/android/reference/com/google/android/gms/maps/MapFragment

OpenWeatherMap. (23 / Juny / 2018). Recollit de OpenWeatherMap: https://openweathermap.org/current

Versions de la plataforma Android. (26 / 06 / 2018). Recollit de Versions de la plataforma Android: https://developer.android.com/about/dashboards/#estadsticas-de-instalacin-de-google-play

9. Annexos

9.1.Enquestes

En els següents enllaços es poden consultar les enquestes i els resultats que es varen realitzar en la secció 3.4.

Enquesta en català.

Resultats enquesta en català.

Enquesta en castellà.

Resultats enquesta en castellà.

9.2. Avaluacions

Les dades de la Figura 69 i la Figura 70 estan basades en els resultats extrets de les observacions que es van dur a terme a l'avaluació de l'aplicació. En el següent enllaç es poden consultar els resultats.