UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

Pack for you: repartiments millors I més ràpids

Treball Fi de Grau

Grau en Enginyeria Informàtica

AUTOR/A: Soriano López, Víctor

Tutor/a: Andrés Martínez, David de

Cotutor/a extern: ROSSI, BRUNO

CURS ACADÈMIC: 2021/2022

Resum

Pack4You és una aplicació mòbil enfocada als repartidors. Aquest treball busca facilitar la feina d'aquests oferint un suport per a la gestió dels paquets i l'ordenació dels mateixos mitjançant diversos algorismes. El repartidor iniciarà sessió i automàticament tindrà una llista amb els paquets que ha de repartir eixe dia. Tindrà una vista prèvia a un mapa de la ruta que haurà de seguir, la qual canviarà de manera dinàmica amb la modificació de qualsevol paquet o algorisme d'ordenació.

Una vegada obtesa la ruta desitjada, el repartidor podrà iniciar la ruta. Ací es mostrarà al mapa el pròxim paquet a entregar i informació més detallada sobre aquest, així com una nota informativa si el client ha decidit deixar-ne una.

Per a obtenir aquesta solució s'ha utilitzat el llenguatge Kotlin i el IDE Android Studio, junt a Jetpack Compose per a dissenyar la Interfície d'Usuari i Firebase com a base de dades. En quant als serveis de geolocalització, s'ha usat Directions API i Distance Matrix API per a obtenir les rutes i el temps de viatge entre dues localitzacions, respectivament. Per a mostrar els mapes s'ha fet servir Maps SDK for Android.

El resultat final és una aplicació amb un disseny intuïtiu i satisfactori per a l'usuari. Aquesta experiència m'ha ajudat a veure la importància d'una bona arquitectura i disseny d'un sistema software, ja que gràcies a aquesta base sòlida és molt més fàcil escalar el projecte, tant per a implementar noves característiques com per a acceptar un volum major d'usuaris.

Paraules clau: Aplicació mòbil, Android, geolocalització, repartiment de paquets, Jetpack Compose, disseny centrat en l'usuari, MVVM, estalvi de temps, algorisme

Resumen

Pack4You es una aplicación móvil enfocada a los repartidores. Este trabajo busca facilitar la labor de estos ofreciendo un soporte para la gestión de los paquetes y la ordenación de los mismos mediante diferentes algoritmos. El repartidor iniciará sesión y automáticamente tendrá una lista con los paquetes que ha de repartir ese día. Tendrá una vista previa en un mapa de la ruta que deberá seguir, la cual cambiará de manera dinámica con la modificación de cualquier paquete o algoritmo de ordenación.

Una vez obtenida la ruta deseada, el repartidor podrá iniciar la ruta. Aquí se mostrará en el mapa el próximo paquete a entregar e información más detallada sobre este, así como una nota informativa si el cliente ha decidido dejar una.

Para obtener esta solución, se ha utilizado el lenguaje Kotlin y el IDE Android Studio, junto a Jetpack Compose para diseñar la Interfaz de Usuario y Firebase como base de datos. En cuanto a los servicios de geolocalización, se ha usado Directions API y Distance Matrix API para obtener las rutas y el tiempo de viaje entre dos localizaciones, respectivamente. Para a mostrar los mapas se ha hecho servir Maps SDK for Android.

El resultado final es una aplicación con un diseño intuitivo y satisfactorio para el usuario. Esta experiencia me ha ayudado a ver la importancia de una buena arquitectura y diseño de un sistema software, ya que gracias a esta base sólida es mucho más fácil escalar el proyecto, tanto para implementar nuevas características como para aceptar un mayor volumen de usuarios.

Palabras clave: Aplicación móvil, Android, geolocalización, reparto de paquetes, Jetpack Compose, diseño centrado en el usuario, MVVM, ahorro de tiempo, algoritmo

Abstract

Pack4You is a mobile app focused on delivery men. This project wants to make their life easier by giving them a way to manage and sort packages by using different algorithms. Once the delivery man logs in, the app displays a list of packages he must deliver that day. It will also preview the best route to follow on a map, which will change dynamically after modifying any package or sorting algorithm.

Once the delivery man selects the desired route, the app will display the next package to deliver on a map, and detailed information about it, including any notes posted by the addressee.

This solution uses the Kotlin programming language and the Android Studio IDE for its development; the Jetpack Compose to design the User Interface and Firebase as the database. The Directions API and the Distance Matrix API obtain the routes and the travel time between two locations, respectively, whereas the Maps SDK for Android provides the maps for the app.

The final result is a mobile application with an intuitive and satisfactory design for the user. This experience has raised my awareness of how important a good architecture and design of a software system is. With this solid foundation, it is much easier to scale the project both to implement new features and accept an increasing number of users.

Key words: Mobile app, Android, geolocation, package delivery, Jetpack Compose, user-centered design, MVVM, time saving, algorithm

Índex

Índex de figures

Índex de taules

CAPÍTOL 1
Introducció

En un món tan globalitzat com el d'avui, els repartidors són una peça fonamental. Són ells qui ens fan arribar la majoria de les coses que fem servir i consumim. Aquesta necessitat s'ha incrementat més encara amb els últims anys per tota l'evolució tecnològica que hem experimentat. Internet i la seua popularització va fer possible contactar amb gent d'arreu del món i, per tant, tenir molta més oferta (i, clar està, demanda) de qualsevol cosa que desitjares.

Si el fenòmen d'Internet ja va incrementar significativament la demanda dels serveis de missatgeria, l'aparició d'Amazon la va multiplicar [1]. Tots coneguem aquesta empresa i la majoria l'hem fet servir per demanar qualsevol cosa. Des de la seua creació i posterior popularització, un gran percentatge dels paquets a entregar provenen dels seus magatzems.

Si a aquesta realitat li sumem l'impacte de la pandèmia COVID-19, on aquest servei s'ha utilitzat encara més [2] degut, entre altres coses, a la necessitat de les persones de rebre diferents articles (incloent menjar i productes de primera necessitat) sense poder eixir de casa, obtenim un sector importantíssim on l'eficiència és clau.

Així doncs, si volem tenir la màxima eficiència possible, tenim que optimitzar la ruta que els repartidors han de seguir per a que dure el menor temps possible, així com minimitzar el temps que no estiguen desplaçant-se d'un lloc a un altre (com ara cridar per telèfon als clients o gestionant l'ordre dels paquets).

És ací on entra Pack4You, una aplicació per a ajudar als repartidors en la seua feina. Amb aquesta aplicació, el repartidor només haurà d'iniciar sessió i automàticament obtindrà els paquets que ha de repartir eixe dia. Aquests es mostraran en un mapa amb la ruta corresponent a l'ordre en que estiguen en la llista. El repartidor podrà ordenar la llista amb diferents algorismes, com són *Brute Force*, *Nearest Neighbour*, Directions API o per urgència. D'aquesta manera, el repartidor no tindrà que perdre temps ordenant-los a mà i, a més a més, obtindrà la millor ruta possible en eixe moment, ja que, com utilitzem els serveis de Google en temps real, obtenim la informació més actualitzada sobre l'estat del tràfic. El repartidor podrà també obtenir informació personalitzada del client en cada paquet en forma d'anotacions. Així, no haurà de trucar per telèfon per obtenir indicacions més específiques en cas de ser necessari. A més a més, els algorismes Brute Force i Nearest Neighbour només necessiten de connexió a Internet a l'inici de la ruta. Una vegada obtingudes les dades necessàries per a la seua

execució, el repartidor podrà borrar i marcar com entregats diversos paquets i els dos algorismes funcionaran sense problema de forma offline.

1.1 Motivació

Abans de començar aquest treball, també vaig pensar en altres alternatives:

- Una xarxa social de videojocs on podries veure resums, crítiques i comentaris sobre aquests, posar-los en preferits, tenir una llista de "Jugats", "Per jugar" i "Interessants", veure les llistes dels teus amics, etc.
- Un convertidor de text d'ordinador a Braille. Utilitzaria alguna mena de Hardware on hi hauria una espècie de punts que puguen pujar i baixar per representar les paraules que hi ha a la pantalla. El meu programa faria de *driver* entre aquestes paraules i el hardware mencionat.
- Una app per a juntar persones que volen adquirir un mateix article per, d'aquesta manera, poder comprar-lo al per major i siga més econòmic per a tots.

Finalment, em vaig decantar per aquesta idea de millorar el sector de la missatgeria per divereses raons:

- Volia treballar amb la geolocalització i aprendre més sobre l'ús i les oportunitats que aquesta brinda.
- El meu germà major va treballar com a repartidor, pel que m'era més fàcil obtenir informació del sector i veure els problemes tecnològics d'aquest.
- No hi havia (o, almenys, no tenia constància) una gran varietat d'aplicacions per als repartidors que ordenaren automàticament els paquets. Totes demanaven a l'usuari ordenar-los de manera manual, així com registrar-los un per un.
- Aquest projecte estava dins del meu pressupost i capacitat. Per exemple, el projecte relacionat amb Braille també em feia il·lusió, doncs és un possible treball que vaig pensar quan anava a segon de carrera i sabia que podia ajudar a la gent, però, siguent realista, em vaig adonar que aconseguir o dissenyar un dispositiu hardware així estava fora de les meues competències i temps.

Les altres propostes tenien ja alternatives al mercat molt paregudes o millors que la meva idea (Com la xarxa social de videojocs o el convertidor a Braille) o tècnicament eren complicades de dissenyar o desenvolupar (el convertidor a Braille i l'aplicació per comprar al per major). En aquesta última proposta el problema era la forma d'assegurar el pagament conjunt al venedor i, posteriorment, l'enviament o recollida de cada article al seu respectiu comprador.

És per totes aquestes raons per les que finalment vaig decidir desenvolupar Pack4You: motivació per aprendre sobre la tecnologia utilitzada, oportunitat de millorar un sector tan important com el de missatgeria, facilitat per obtindre informació del propi sector i capacitat per fer producte software de qualitat.

1.2 Objectius

L'objectiu principal d'aquest treball és facilitar la vida dels repartidors, afegint mecanismes de gestió i ordenació dels paquets a repartir. D'aquesta manera, no és el repartidor qui ha d'ordenar els paquets manualment, sinó que s'obtindrà una ruta automàticament i amb un menor temps de repartiment.

El segon objectiu fonamental d'aquest treball és realitzar un projecte des de l'inici: començant per l'anàlisi de requisits, passant pel disseny i implementació del sistema i, finalment, la verificació d'aquest. Es posarà especial atenció a l'arquitectura del sistema i la seua correcta implementació.

El tercer objectiu és aprendre sobre la integració de característiques de geolocalització i mapejat a una aplicació Android.

Finalment, el quart i últim objectiu és investigar i implementar diferents algorismes aplicables a l'àmbit de la missatgeria i geolocalització.

1.3 Estructura de la memòria

- **Introducció:** En aquest apartat s'explica de forma general el context en el que es va a treballar i el problema que es vol tractar, així com la motivació que ha dut a la realització d'aquest projecte i els objectius que es volen complir.
- **Estat de l'art:** En aquest capítol es realitzarà un estudi de les diferents alternatives disponibles al mercat i les seues carcaterístiques per a així poder fer un anàlisi de requisits adequat al nostre context.
- **Anàlisi del problema:** Aquest capítol es centrarà en l'anàlisi de requisits del projecte. Es definirà què és un requeriment i el tipus de requeriments que existeixen. Seguidament es descriuran els requeriments presents en aquest projecte, es prioritzaran utilitzant la tècnica MoSCoW i, finalment, es presentaran els casos d'ús de Pack4You.
- **Disseny de la solució:** Aquest apartat descriurà el procès de disseny de Pack4You. Es començarà per l'explicació de l'arquitectura utilitzada a l'aplicació. Després, es mostrarà el diagrama de classes i es comentaran les distintes entitats presents en aquest. Seguidament, es discutirà sobre els diferents tipus de prototipat que existeixen i es mostraran els prototips de Pack4You. A continuació, es descriurà el disseny de la base de dades, explicant les diferents entitats que la composen. Finalment, les tecnologies utilitzades al projecte seran comentades.
- **Desenvolupament de la solució proposta:** En aquest capítol es descriurà el procés de desenvolupament de Pack4You. Es començarà explicant l'estructura del projecte (basada en l'arquitectura MVVM) per a, després, discutir els patrons de disseny implementats i com s'ha aconseguit una arquitectura *Clean*, indicant els seus beneficis. Seguidament es descriurà l'element

fonamental de Jetpack Compose, les funcions composables. Després es comentarà quina solució s'ha trobat per implementar els mapes a Pack4You i el paper que Directions API ha tingut en aquests. Finalment, els algorismes presents en aquest projecte seran explicats, incloent la seua implementació i complexitat algorítmica.

- **Testing:** En aquest apartat es descriuran les proves que s'han dut a terme per comprovar el funcionament de Pack4You. Es començarà explicant les proves unitàries, seguides de les d'integració. Després es comentaran les proves de regressió executades a la fase final del projecte. Acte seguit es veurà com els *Core app quality tests* han ajudat a comprovar si l'aplicació compleix amb uns criteris de qualitat definits pels desenvolupador Android de Google. Finalment, s'explicaran les proves executades per comprovar el correcte funcionament dels algorismes implementats manualment.

- **Resultats:** Aquest capítol servirà per mostrar el resultat final del projecte. Ací s'explorarà l'aplicació utilitzant com a exemple un dia qualsevol d'un repartidor, recorrent els diferents casos d'ús suportats per Pack4You.

- **Conclusions:** Aquest apartat explicarà les conclusions a les que jo, com a graduat al grau d'Enginyeria Informàtica a la UPV, he arribat durant la consecució d'aquest projecte. També es discutirà si finalment s'han aconseguit els objectius proposats. Després es comentaran quins han sigut els aspectes més complicats d'implementar i com s'han solucionat, junt a les limitacions d'aquesta versió del producte. Finalment, es comentarà el coneixement adquirit durant el desenvolupament de Pack4You i la relació d'aquest projecte amb els estudis cursats.

- **Treball futur:** Per a finalitzar, aquest capítol servirà per descriure els requeriments i refinaments que voldria haver afegit en aquest treball però, per falta de recursos, no ha sigut possible.

CAPÍTOL 2
Estat de l'art

Al mercat d'aplicacions actual hi ha diverses alternatives per ajudar als repartidors a fer la seua feina més còmoda. Algunes se centren més en la part d'optimització de la ruta i no ofereixen la part de gestió de paquets o, si ho fan, és d'una manera un poc farragosa. El que Pack4You busca solucionar és aquesta gestió de paquets, oferint una proposta amb una Interfície d'Usuari clara i satisfactòria.

2.1 Apps similars

En aquesta secció es discutirà sobre les diferents propostes que s'han trobat al mercat d'aplicacions actual. Després d'investigar sobre diverses opcions, s'han elegit tres d'on s'extrauran possibles característiques i idees per a Pack4You: Google Maps [3], ZippyKind [4] i RoadWarrior [5, 6]

2.1.1. Google Maps

És coneguda arreu del món i tots l'hem gastada alguna vegada. Aquesta aplicació va ser creada per Google, i ens permet buscar direccions d'arreu del món (figura 2.1), indicant-nos com anar cap a elles. Com es pot veure a la figura 2.2, també es poden afegir diverses parades i, segons l'ordre que se li ha donat, ens tornarà la ruta que s'ha de seguir per a anar a totes elles.

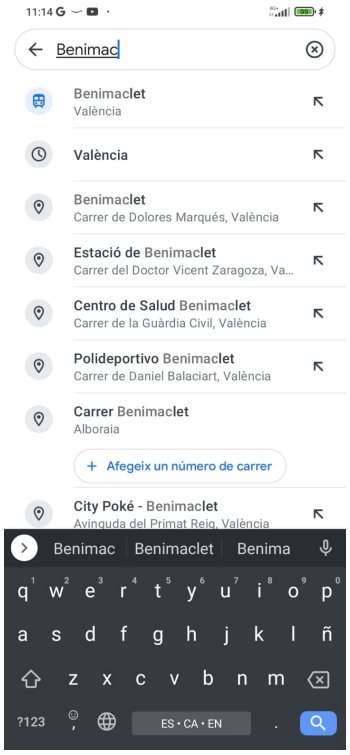

Figura 2.1: Cercant una direcció a Google Maps

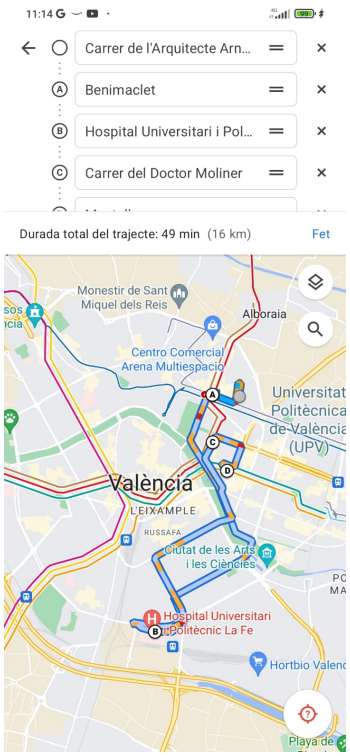

Figura 2.2: Ruta amb diverses parades a Google Maps

2.1.2. Zippykind

Zippykind és una aplicació que et permet gestionar la teua pròpia mercaderia i assignar-la a diferents repartidors en temps real. El producte, que és similar al funcionament d'Uber [7], conté tres rols:

- **Client:** Accedint al link enviat per SMS quan el repartidor està de camí, el client pot localitzar el seu paquet i parlar amb el repartidor per xat. També pot consultar informació sobre aquest (nom i foto), quant de temps tardarà en arribar o el vehicle que condueix. Aquesta vista podem observar-la a la figura 2.3.

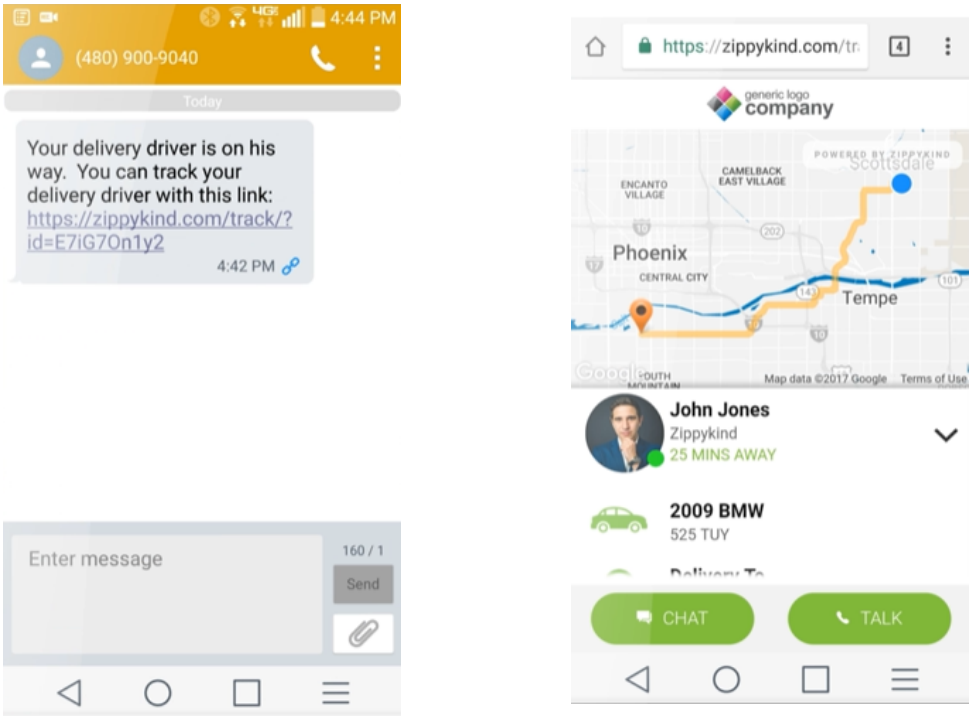

Figura 2.3: Vista client ZippyKind

- **Repartidor:** Accedint a l'aplicació per a smartphones, el repartidor pot observar un mapa amb les parades que ha de realitzar (figura 2.4), així com una llista dels tickets a repartir, com veiem a la figura 2.5. Aquestes parades estan optimitzades per obtenir la millor ruta, i pots elegir entre cotxe, a peu, bicicleta i transport públic. El repartidor també pot veure amb detall la informació sobre un ticket (preu, què conté, alguna nota) i el client (nom, telèfon...) (figura 2.6). Per últim, l'aplicació disposa d'un navegador integrat per arribar al seu pròxim destí, com es pot veure a la figura 2.7.

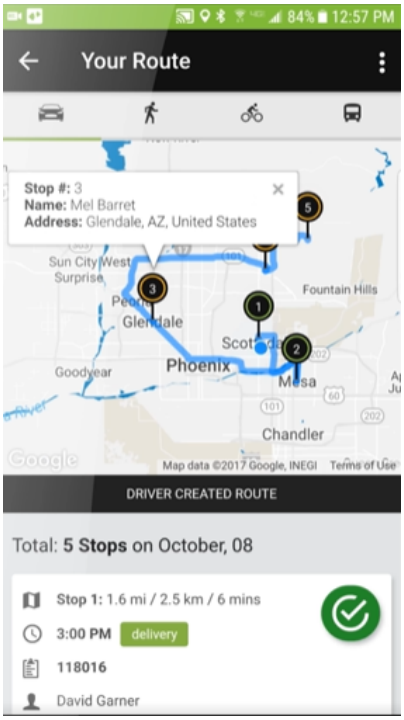

Figura 2.4: Mapa amb les diverses parades a ZippyKind

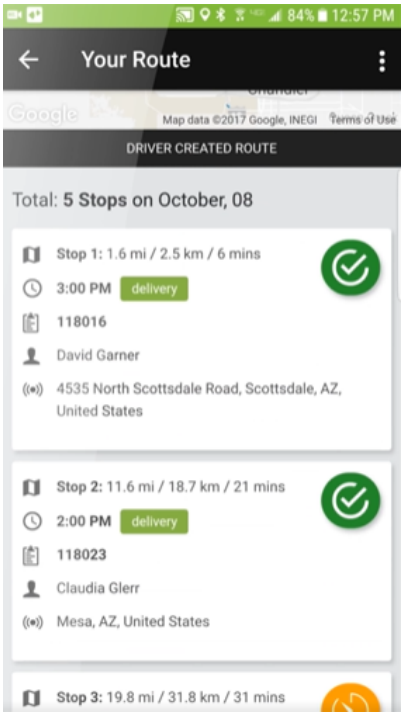

Figura 2.5: Llista de tickets a Zippykind

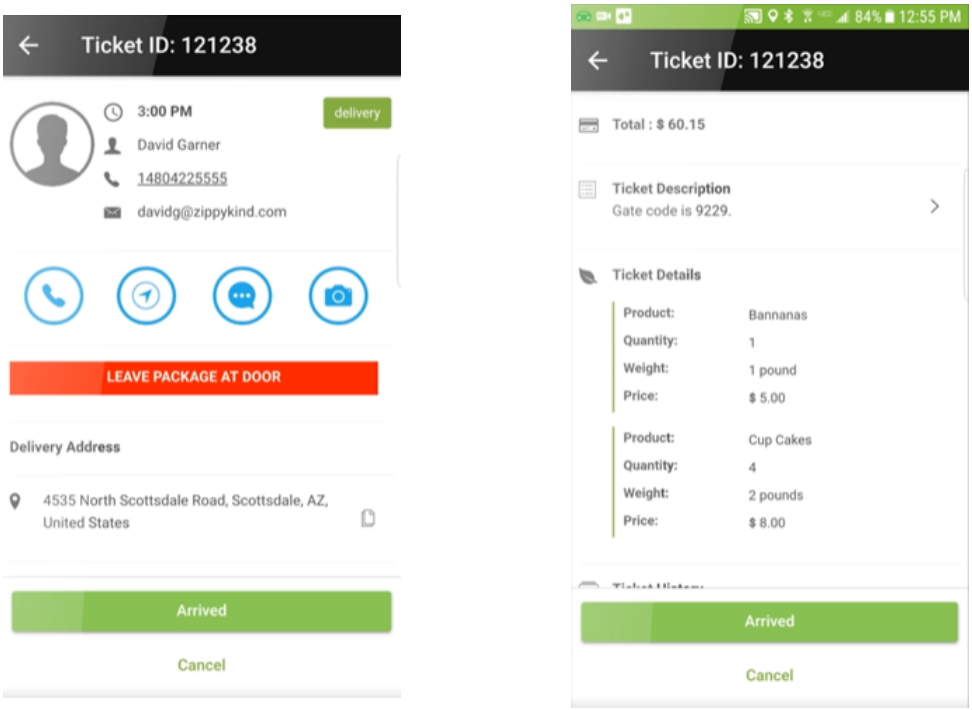

Figura 2.6: Informació sobre el ticket a repartir

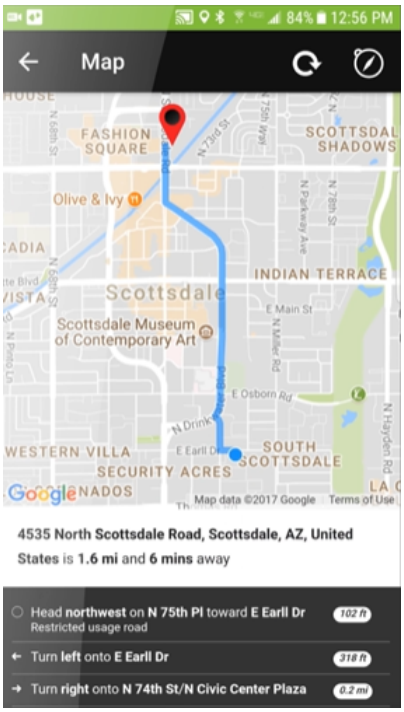

Figura 2.7: Mapa amb les direccions per a arribar al seu pròxim ticket a ZippyKind

- **Administrador:** Accedint al client web de ZippyKind, l'administrador tindrà la possibilitat d'assignar tickets als diferents repartidors registrats. També podrà crear nous tickets (figura 2.8) i modificar-ne els existents, així com consultar informació sobre els repartidors i la seua posició en temps real, com es pot observar a la figura 2.9.

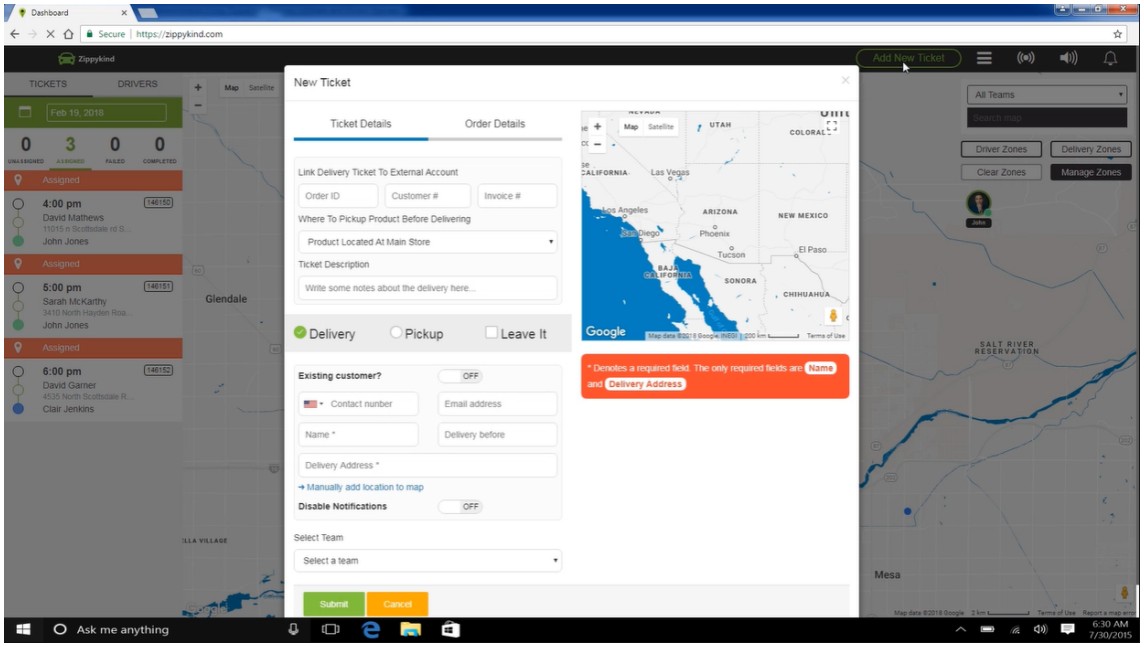

Figura 2.8: Creació d'un nou ticket a ZippyKind

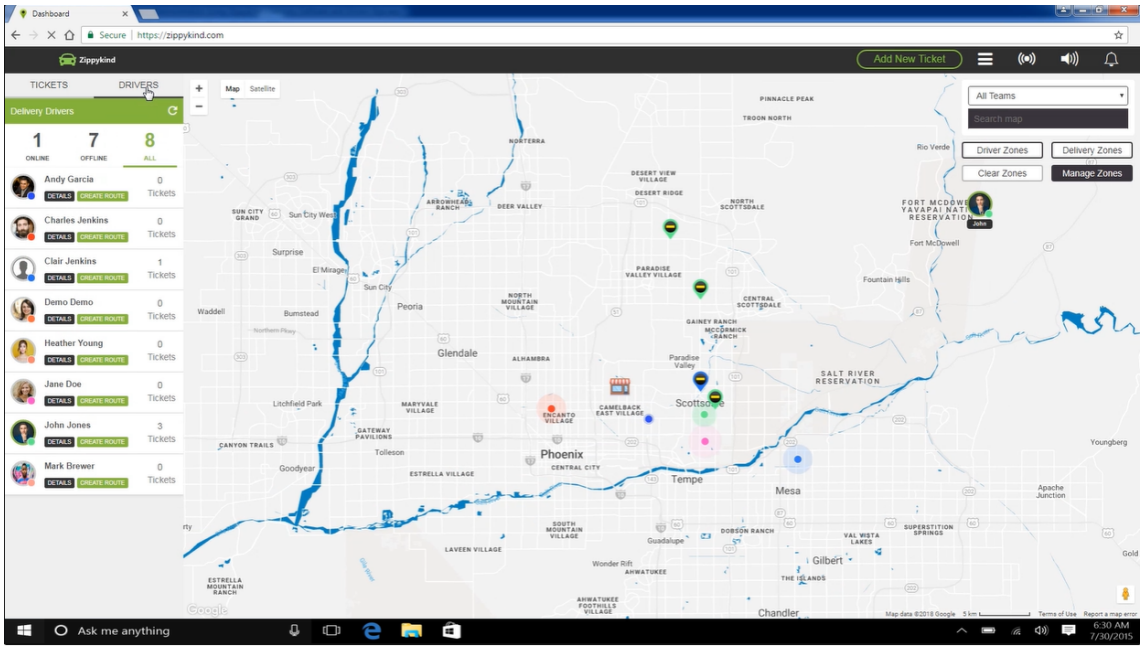

Figura 2.9: Dashboard del client web ZippyKind

2.1.3. RoadWarrior

RoadWarrior és un planificador de rutes que et permet optimizar el camí que vas a realitzar per recórrer-lo en el menor temps possible. També es poden ordenar les parades manualment (figura 2.10) i donar-li prioritat a aquestes, així com visualitzar aquestes parades en un mapa interactiu, com veiem a la figura 2.11. L'aplicació consta de dos rols: repartidor i administrador (figura 2.11), els quals tenen funcions similars als rols del mateix nom de ZippyKind.

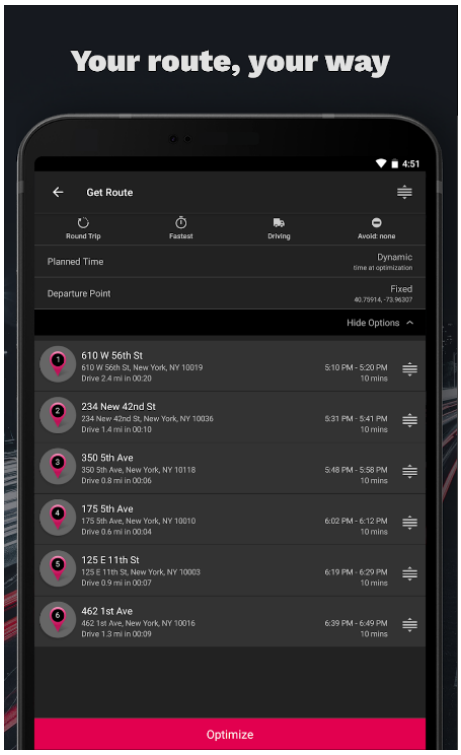

Figura 2.10: Gestió de la ruta a RoadWarrior

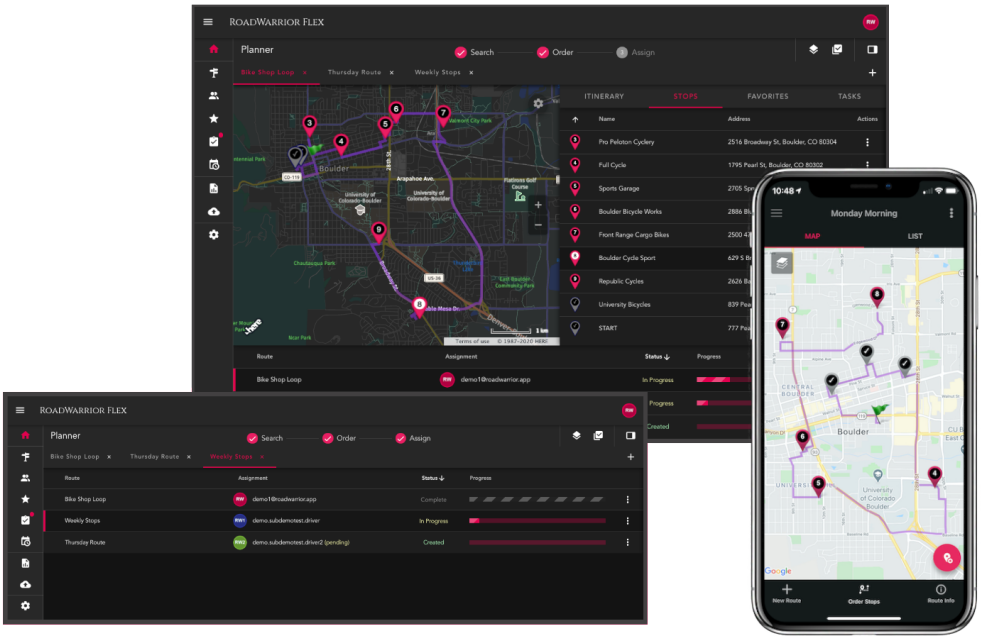

Figura 2.11: Mode administrador i mapa amb parades de RoadWarrior

2.2 Característiques

Després d'haver analitzat algunes de les alternatives disponibles al mercat, anem a extraure diverses característiques presents a aquestes apps i debatre sobre la inclusió o no d'elles a Pack4You. Finalment, afegiré algunes característiques que no he trobat en cap d'aquestes alternatives i poden ser molt útils, així com fer destacar a Pack4You entre les altres opcions.

2.2.1. Característiques d'alternatives al mercat

- **Optimització de la ruta:** És una característica clau. Si un algorisme optimitza la ruta que tenim que recórrer, estalviarem molt de temps. El repartidor no haurà d'ordenar els paquets a mà i, a més a més, s'obtindrà una ruta igual o (probablement) millor que la que el repartidor hauria decidit. A més, si s'obtenen les dades del tràfic en temps real, la solució serà encara millor.
- **Consulta de paquets:** És molt important que el repartidor puga consultar de forma clara i satisfactòria els paquets presents a la ruta, així com informació sobre aquests.
- **Priorització de paquets:** Moltes vegades es dóna el cas que un client necessita un paquet urgentment. És per açò que és important poder-li assignar diferents prioritats als paquets per repartir-los els primers si cal.
- **Visualització de la ruta en el mapa:** S'ha de poder observar a un mapa la ruta que el repartidor realitzarà, així com obtenir informació del paquet corresponent polsant al marcador del mapa.
- **Gestió de paquets:** El repartidor hauria de poder afegir, eliminar i modificar paquets per ell mateix, sense dependre de cap altra persona. A ZippyKind, els tickets només poden ser creats i modificats per l'administrador.
- **Gestió i consulta de paquets, i mapa en una sola pantalla:** Poder eliminar o reordenar els paquets de la ruta, i veure automàticament aquest canvi al mapa sense haver de canviar de pantalla fa a l'aplicació més fluïda i intutïtiva.
- **Vista de client:** Una funcionalitat molt útil és comunicar-li al client quant tardarà el seu paquet i poder parlar amb el repartidor en temps real.
- **Navegador integrat:** Dins de l'aplicació pot ser útil tenir un navegador integrat per poder anar fins el teu pròxim destí seguint les instruccions que aquest et proporciona.
- **Vista d'administrador:** Una característica molt interessant és poder monitoritzar a diferents repartidors alhora i assignar-li paquets en temps real, així com obtenir informació d'aquests i dels paquets que s'han de repartir.

Una vegada llistades les característiques extretes, anem a visualitzar-les (taula 2.1) per saber quines aplicacions les han inclòs i quines no, per a després discutir quines formaran part de Pack4You amb el temps i els recursos disponibles.

Taula 2.1: Taula comparativa de característiques presents a diverses alternatives disponibles al mercat

Característica	Google Maps	ZippyKind	RoadWarrior	Pack4You
Optimització de ruta		✓	✓	✓
Consulta de paquets	✓	✓	✓	✓
Priorització de Paquets			✓	✓
Visualització ruta en mapa	✓	✓	✓	✓
Gestió de paquets	✓	Només administrador	✓	✓
Gestió, consulta i mapa en una sola pantalla	✓	✓		✓
Vista client		✓		
Navegador integrat	✓	✓		
Vista administrador		✓	✓	

Com podem observar, Google Maps és l'única que no permet optimitzar la ruta, tot i siguent una característica molt important. Així i tot, Google ofereix una API (Application Programming Interface), Directions API [8] (que, com es vorà més endavant, s'ha utilitzat al projecte), ferramenta que ens permet, entre altres coses, optimitzar una ruta donades fins 25 parades. També podem veure que tots els productes analitzats permeten gestionar les parades o paquets de la ruta, així com visualitzar aquestes parades a un mapa interactiu i afegir i modificar els paquets de la ruta actual. A més, cap destacar que, com es pot observar a la figura 2.10, si es necessita gestionar la ruta s'ha de canviar de pantalla, acció que fa un poc tosca i no tant intuïtiva la navegació de l'aplicació.

Per altra banda, la priorització de paquets només l'ofereix RoadWarrior. Si Pack4You també inclou aquesta característica, pot sobreeixir per damunt d'altres opcions. A més a més, RoadWarrior també inclou vista client i vista administrador, dos caracterísitques molt útils però que, si s'ajustem al temps i recursos disponibles per a Pack4You, no són del tot viables. Aquesta última característica també està present a ZippyKind.

Finalment tenim el navegador integrat. RoadWarrior és l'únic que no l'inclou. Avui en dia, tenint a tanta gent ja familiaritzada amb Google Maps i disposant aquesta aplicació un funcionament tan òptim, l'implementació pròpia d'aquesta característica no és prioritària.

A mode de resum, Pack4You inclourà les característiques d'optimització de ruta, priorització, consulta i gestió de paquets i visualització de ruta en el mapa

gestió de paquets, i tot açò en una sola pantalla. Les vistes de client i administrador són enfocaments molt interessants, però que escapen de les dimensions d'aquest projecte. Per últim, el navegador integrat, no és una característica prioritària existint la ferramenta Google Maps.

Cap mencionar que és en la part d'administrador on es realitzaria la gestió de clients i repartidors, l'assignació de paquets a cada repartidor, etc. En aquest projecte només estem centrant-nos en la part del repartidor, quedant aquesta vista d'administrador com a part d'altre desenvolupament o inclús TFG.

2.2.2. Característiques innovadores

Com hem vist, les característiques presents a les alternatives del mercat són molt interessants, i podem desenvolupar una aplicació útil però que no aporte res de nou. És per açò que anem a mencionar i incloure al projecte algunes característiques innovadores per fer que Pack4You destaque per damunt d'altres opcions similars. Aquestes són:

- **Ordenació de paquets utilitzant diferents algorismes:** El repartidor podrà ordenar les seues entregues utilitzant Directions API [8], *Brute Force* [9], *Nearest Neighbour* [10] i per Urgència. A més a més, els algorismes de *Brute Force* i *Nearest Neighbour*, una vegada calculats per primera vegada, podran seguir utilitzant-se de forma offline. Així, l'ordre dels paquets canviarà a mida que s'eliminen o es marquen com a entregats.

- **Consulta de paquets entregats:** El repartidor podrà consultar els paquets que ja s'han entregat. D'aquesta manera serà possible saber si un paquet estava assignat al propi repartidor i, en aquest cas, si el primer ja ha arribat al seu destí en cas de dubte.

- **Modificació d'últim destí:** Pot ser que el repartidor acabe alguns dies al magatzem i d'altres puga anar directament a casa. Amb aquesta funcionalitat, aquest podrà elegir la seua última localització de manera predeterminada i canviar-la quan vulga.

- **Navegació prèvia de la ruta:** El repartidor podrà iniciar la ruta i observarà el següent paquet a entregar. Podrà desplaçar-se per aquests i observar la seua posició a la ruta, així com marcar-los com entregats. Al polsar sobre el marcador del paquet al mapa, el repartidor podrà elegir entre dues opcions: començar una navegació des de la posició actual a aquesta localització (el qual el redirigeix a Google Maps) o només situar-la a Google Maps.

CAPÍTOL 3
Anàlisi del problema

Aquest capítol de la memòria se centrarà en l'anàlisi funcional de Pack4You: primerament, es llistaran els requeriments funcionals i no funcionals. Després, es descriuran les funcionalitats amb les que el producte final Pack4You comptarà, i es classificaran seguint la priorització MoSCoW [11]. Finalment, es mostrarà i explicaran els actors i casos d'ús del diagrama de casos d'ús d'aquest projecte.

3.1 Requeriments

La *IEEE Standard Glossary of Software Engineering Terminology* [12] defineix un requeriment (o requisit) com a *"Una condició o capacitat necessitada per un usuari per a resoldre un problema o assolir un objectiu"*. Així doncs, podem dir que tots els requeriments d'un sistema software han de complir-se quan el producte estiga finalitzat ja que, si no és així, l'usuari no podrà resoldre o aconseguir els objectius que el software anava a permetre-li.

A continuació, es descriuran els requisits funcionals i no funcionals de Pack4You.

3.1.1. Requeriments funcionals

Els requisits funcionals [12] són aquells que descriuen una funcionalitat del sistema, és a dir, diuen *què* ha de fer el producte software. Per a Pack4You, s'han considerat aquests requisits funcionals:

- El sistema ha de permetre a l'usuari consultar els paquets de la ruta.
- El sistema ha de permetre a l'usuari afegir un nou paquet.
- El sistema ha de permetre a l'usuari eliminar un paquet.
- El sistema ha de permetre a l'usuari posposar un paquet.
- El sistema ha de permetre a l'usuari editar un paquet.
- El sistema ha de permetre a l'usuari marcar un paquet com a entregat.
- El sistema ha de permetre a l'usuari consultar els paquets ja entregats.

- El sistema ha de permetre a l'usuari ordenar els paquets utilitzant l'algorisme Directions API.
- El sistema ha de permetre a l'usuari ordenar els paquets utilitzant l'algorisme *Brute Force*.
- El sistema ha de permetre a l'usuari ordenar els paquets utilitzant l'algorisme *Nearest Neighbour*.
- El sistema ha de permetre a l'usuari ordenar els paquets per urgència.
- El sistema ha de recalcular la ruta quan un paquet és borrat, afegit o modificat.
- El sistema ha de posicionar la localització dels paquets a un mapa.
- El sistema ha de permetre a l'usuari iniciar sessió mitjançant el seu correu i contrassenya.
- El sistema ha de permetre a l'usuari tancar sessió.
- El sistema ha de permetre a l'usuari iniciar la ruta i accedir a una vista prèvia de la realització d'aquesta.
- El sistema ha de permetre a l'usuari consultar informació del paquet polsant el marcador corresponent al mapa.

3.1.2. Requeriments no funcionals

Els requisits no funcionals [12] són aquells que descriuen els criteris que s'hauran de seguir per evaluar si un sistema compleix amb els estàndards de seguretat, disponibilitat, rendiment... Per tant, podem dir que aquests requeriments ens diuen *com* ha de funcionar el nostre sistema.

- Quan el sistema està recalculant una ruta, no hauria de tardar més de 7 segons en retornar la nova ruta.
- Quan el repartidor inicia sessió, la llista amb els paquets que s'han d'entregar hauria de carregar-se en menys de 3 segons.
- Un repartidor només ha de poder accedir als paquets assignats al mateix.
- El sistema hauria de dibuixar la nova ruta en menys de 3 segons.
- El sistema hauria de funcionar en dispositius amb sistema operatiu Android 10.0 o superior.
- L'aplicació hauria de tenir un tamany menor a 70MB.

3.2 Priorització MoSCoW

La priorització MoSCoW [11], també anomenada mètode MoSCoW, és una popular tècnica de priorització de requeriments d'un sistema software. Aquesta consisteix en classificar, junt a l'equip de desenvolupament, les diferents característiques que es volen incloure en un producte segons la seua importància en aquest. Per fer-ho, aquestes funcionalitats es separaran en els següents grups:

- **Must have:** Requeriments fonamentals. El producte, passe el que passe, haurà de comptar amb aquestes característiques.
- **Should have:** Requeriments importants. Aporten un gran valor si finalment s'implementen, però sense elles el producte seguirà tenint utilitat.
- **Could have:** Requeriments no necessaris. Si finalment són implementats al producte, aportaran valor, però aquest no és una prioritat. La diferència amb "Should have" és que, de finalment no ser implementades aquestes característiques,l'impacte en el producte és menor.
- **Won't have:** Requeriments no importants. Donat el temps i els recursos actuals, aquestes funcionalitats no seran implementades. Pot ser que en altre moment algunes d'aquestes característiques siguen prioritàries, però per a aquesta entrega ningú invertirà temps en elles.

Una vegada explicat el mètode MoSCoW, es presentarà la classificació dels requeriments de Pack4You utilitzant aquesta tècnica (figura 3.1) [1].

Els requeriments presents a "Must Have" s'han considerat prioritaris perquè són indispensables per a la mínima funcionalitat de l'aplicació. El repartidor realitza aquestes accions a sovint i, sense alguna d'aquestes, no podria completar satisfactòriament un dia normal.

Els requisits que es troben a "Should have" són característiques també molt importants, però que, de no estar, el repartidor pot seguir treballant, tot i que aquestes aportarien funcionalitats molt útils per a ell.

A "Could have" trobem requisits útils que, de tenir finalment recursos disponibles, s'implementaran a l'aplicació. Hi ha alguns ítems que són molt útils (com "Gestionar resposta del client per canviar en temps real la informació dels paquets" o "Integració del missatge enviar al client amb Telegram") però que, com necessiten una gran quantitat de recursos per a la seua realització, els he hagut d'incloure a aquesta secció.

Finalment, els requeriments presents a "Won't have" no són factibles per a aquesta entrega. Són ítems amb una demanda de recursos molt gran i que no són prioritaris per a Pack4You. Així i tot, és possible que en futures entregues estiguen presents a "Must have".

[1] Plantilla disponible a https://www.figma.com/community/file/1121894777164334454

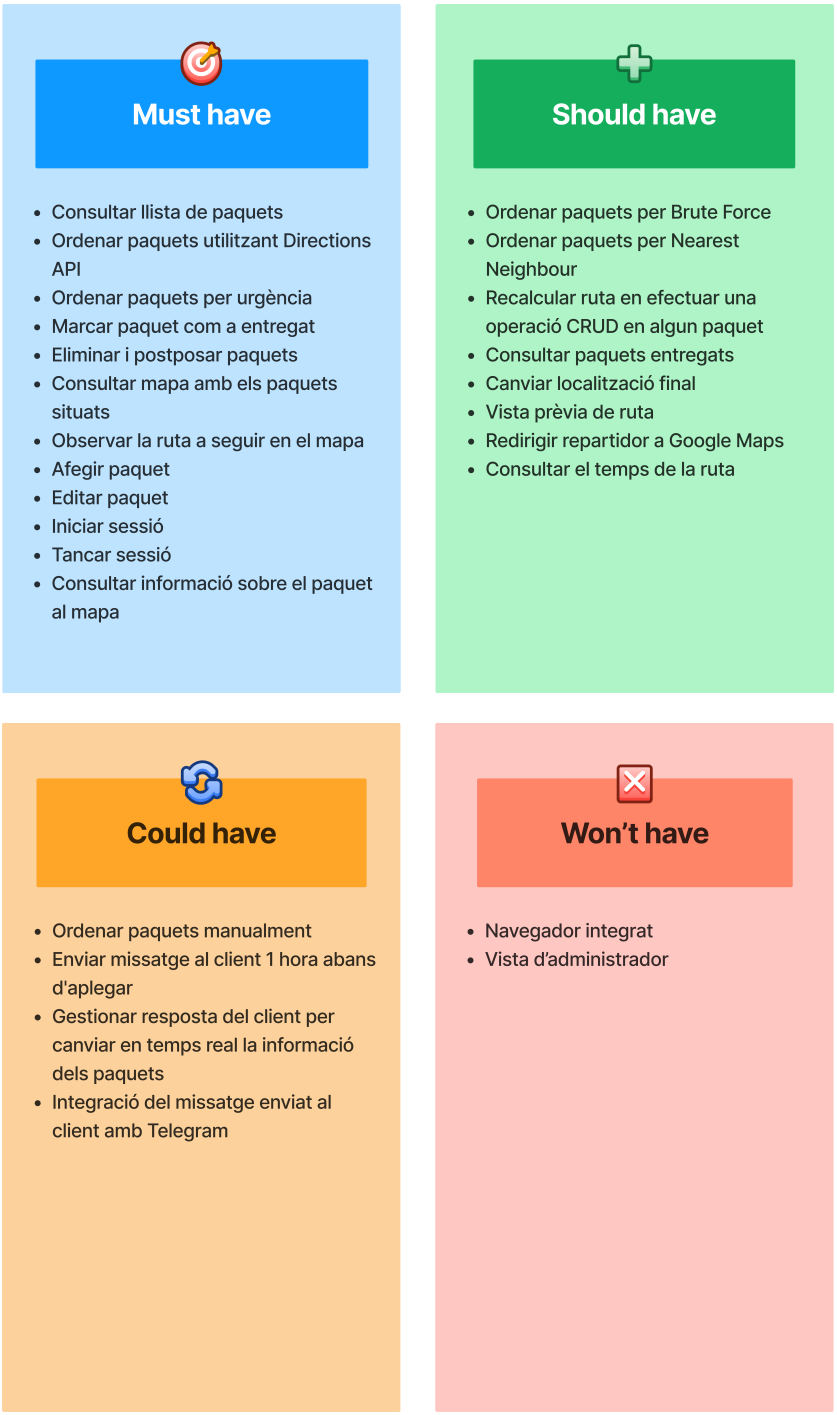

Figura 3.1: Requeriments classificats utilitzant la priorització MoSCoW

3.3 Casos d'ús

Segons Grady Booch, James Rumbaugh i Ivar Jacobson [13], un cas d'ús es pot definir com a *"una seqüència d'accions, incloent variacions, que el sistema pot executar i que produeix un resultat de valor observable per a un actor que interactua amb el sistema"*. En altres paraules, és una acció que, ja siga per ella mateixa o junt a altres, el sistema pot executar per mostrar-li una resposta a un actor (que poden

ser els propis usuaris o altres sistemes que interactuen amb el nostre). Com els casos d'ús són accions que el sistema executarà, el diagrama que conté aquests és molt útil per explicar i debatre amb el client les característiques amb les que comptarà l'aplicació i mostrar-li de forma general quines seran les funcionalitats del sistema final.

Una vegada explicat el significat i la importància dels casos d'ús en un projecte d'enginyeria del software, es mostrarà i descriurà el diagrama de casos d'ús de Pack4You (figura 3.2).

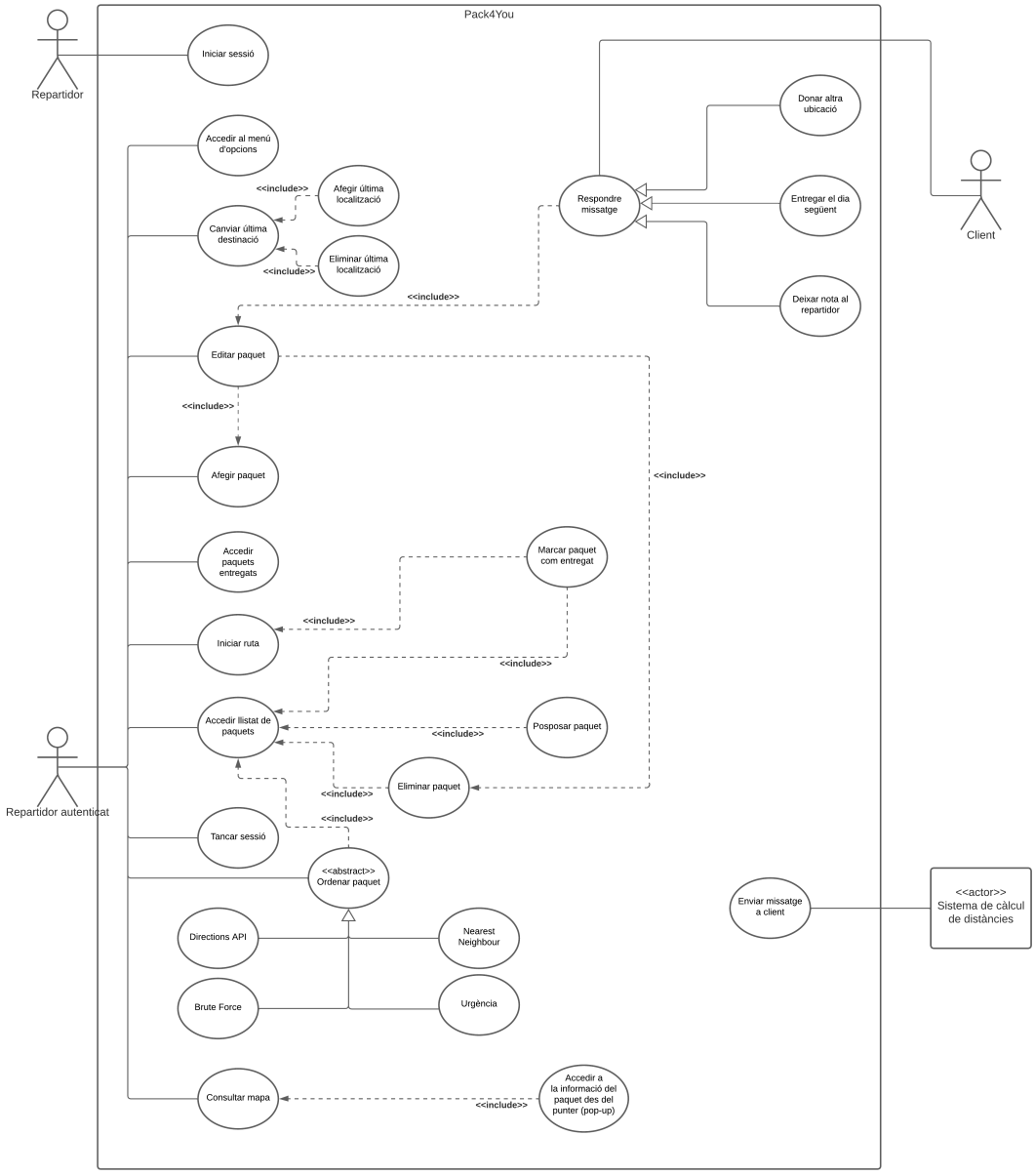

Figura 3.2: Diagrama de casos d'ús de Pack4You

Com es pot observar, en Pack4You tenim quatre actors diferents:

- **Repartidor:** Només pot iniciar sessió.
- **Repartidor autenticat:** És l'actor principal. Pot realitzar la majoria d'accions que el sistema ofereix, com són operacions CRUD (*Create*, *Read*, *Update*, *Delete*) sobre els paquets, ordenar els paquets mitjançant diversos algorismes o consultar el mapa.
- **Sistema de càlcul de distàncies:** S'encarrega de calcular la distància entre la ubicació del repartidor i els diferents paquets per saber quant tardarà aquest en arribar a les diferents destinacions.
- **Client:** Rep missatges a través de Telegram i, en donar una resposta sobre el seu estat (si està en casa o no, si vol canviar l'ubicació del paquet, etc.), aquesta arriba al repartidor i el seu entorn canvia d'acord a la mateixa.

En quant a la descripció detallada de cada cas d'ús, es pot trobar a l'Apèndix A. Casos d'ús en format taula.

CAPÍTOL 4
Disseny de la solució

Una vegada identificats els diferents requeriments de l'aplicació i haver-los classificat per prioritat, es presentarà el disseny de l'aplicació. Es començarà explicant l'arquitectura del sistema: les diferents capes de les que consta i com interactuen entre elles. A continuació es descriurà el diagrama de classes i, amb açò, les diferents classes amb les que comptarà el producte final, junt a les seues relacions. Després, es discutirà sobre la importància del prototipat a la enginyeria de software i es mostraran els wireframes i prototips d'alta fidelitat de l'aplicació [20]. Es continuarà explicant el disseny de la base de dades i, finalment, es comentarà la tecnologia utilitzada per realitzar aquest projecte.

4.1 Arquitectura del sistema

L'arquitectura d'un sistema, segons la pàgina oficial de desenvolupament d'Android [18], *" defineix els limits entre les parts de l'aplicació i les responsabilitats que cada part hauria de tenir"*. És molt important elegir i implementar una bona arquitectura des del principi, ja que aquesta permetrà fer un sistema escalable i facilitarà la implementació de noves característiques i la modificació de les ja existents, així com la simplificació de la verificació del propi sistema. Si no hem elegit una bona arquitectura, canviar-la requereix d'un esforç molt gran.

Tenint açò en compte, per a Pack4You s'ha decidit utilitzar una arquitectura MVVM (Model-View-ViewModel) [14], que és una de les més utilitzades i recomanades per al desenvolupament Android. Aquesta arquitectura presenta, al menys, dues capes:

- **Capa de presentació:** El seu rol és mostrar les dades de l'aplicació per pantalla. Quan aquestes dades canvien (ja siga per una interacció de l'usuari o un input extern), la Interfície d'Usuari (IU) ha de reflectir aquests canvis. En PackForYou, aquesta capa es divideix en dos components:
  - **Vista:** Ací estaran tots els components de la IU (els objectes que es veuen per pantalla) i la comunicació amb el ViewModel. Només està present la lògica de la Interfície d'Usuari (canviar el color d'un botó quan es polsa, etc.), no la lògica empresarial. Aquest component mostrarà les dades per pantalla. També informarà el ViewModel sobre totes les

accions de l'usuari. En el nostre cas, estem utilitzant Jetpack Compose [15] per crear la IU de l'aplicació.

- **State holders:** En Pack4You, aquest rol el tenen els ViewModels. Té les connexions entre la Vista i el Model (d'ací ve el seu nom). La seua funció principal és mantenir les dades que s'han de mostrar a la vista. En cas de no existir capa de domini (com a Pack4You), aquest component serà el que es comunique amb la capa de dades, informant-li sobre les sol·licituds o canvis en aquestes que l'usuari haja realitzat.

- **Capa de dades:** El seu rol és contenir la lògica de negoci i gestionar totes les dades que l'aplicació necessita (ja siga les que estan de forma local o de forma remota). A Pack4You, aquesta capa consta de tres components:

  - **Repositori:** Aquest component conté una o diverses fonts de dades d'on obtindran les dades necessàries. També servirà per preparar i filtrar aquesta informació per al ViewModel. És, per tant, qui alberga la lògica de negoci. Per exemple, si el ViewModel necessita només el nom i correu d'un usuari present a la base de dades, el Repositori, en fer la sol·licitud a la font de dades corresponent, obtindrà l'objecte "Usuari" amb tots els seus atributs, no només amb el seu nom i correu. És, com s'ha dit, el Repositori l'encarregat de retornar-li al ViewModel només el nom i el correu d'aquest usuari.
  - **Model:** Aquest component conté els objectes propis de l'aplicació. És ací on es defineixen les classes de dades.
  - **Font de dades:** Aquest component té les connexions amb els llocs d'on s'obtenen les dades que necessita l'aplicació (arxius locals en cas d'una font de dades local, connexions amb bases de dades remotes en cas d'una font de dades remota). A més, serà l'encarregada de convertir la informació rebuda per aquestes fonts de dades (per exemple, en format JSON [16]) als models de l'aplicació (per exemple, al teu objecte "Usuari"), i viceversa. A Pack4You estem utilitzant Firestore [22], que està suportat per Firebase [17], com a base de dades remota.

Com s'ha comentat, en alguns casos també existeix la capa de domini, que es situa entre la capa de presentació i la de dades. La capa de domini seria l'encarregada de connectar aquestes dues capes i encapsular tota la lògica empresarial complexa. Aquesta capa és opcional perquè no totes les apps tenen aquests requisits.

Amb una arquitectura com aquesta, els components externs (Font de dades) no tenen coneixement sobre els interns (Vista), així que poden ser utilitzats per a diversos propòsits sense cap problema. Si aquests components interns canvien, no cal modificar el codi dels components externs. Són els primers els que utilitzen les funcionalitats dels segons, mentre aquests únicament envien la resposta o realitzen l'acció corresponent sense saber per a què va a utilitzar-se. Açò permet, per exemple, que diverses Vistes tinguen un mateix ViewModel, o que una mateixa Font de dades puga proporcionar informació a diversos Repositoris sense problema.

Finalment, a la figura 4.1 es pot observar un diagrama d'aquesta arquitectura MVVM utilitzada a Pack4You.

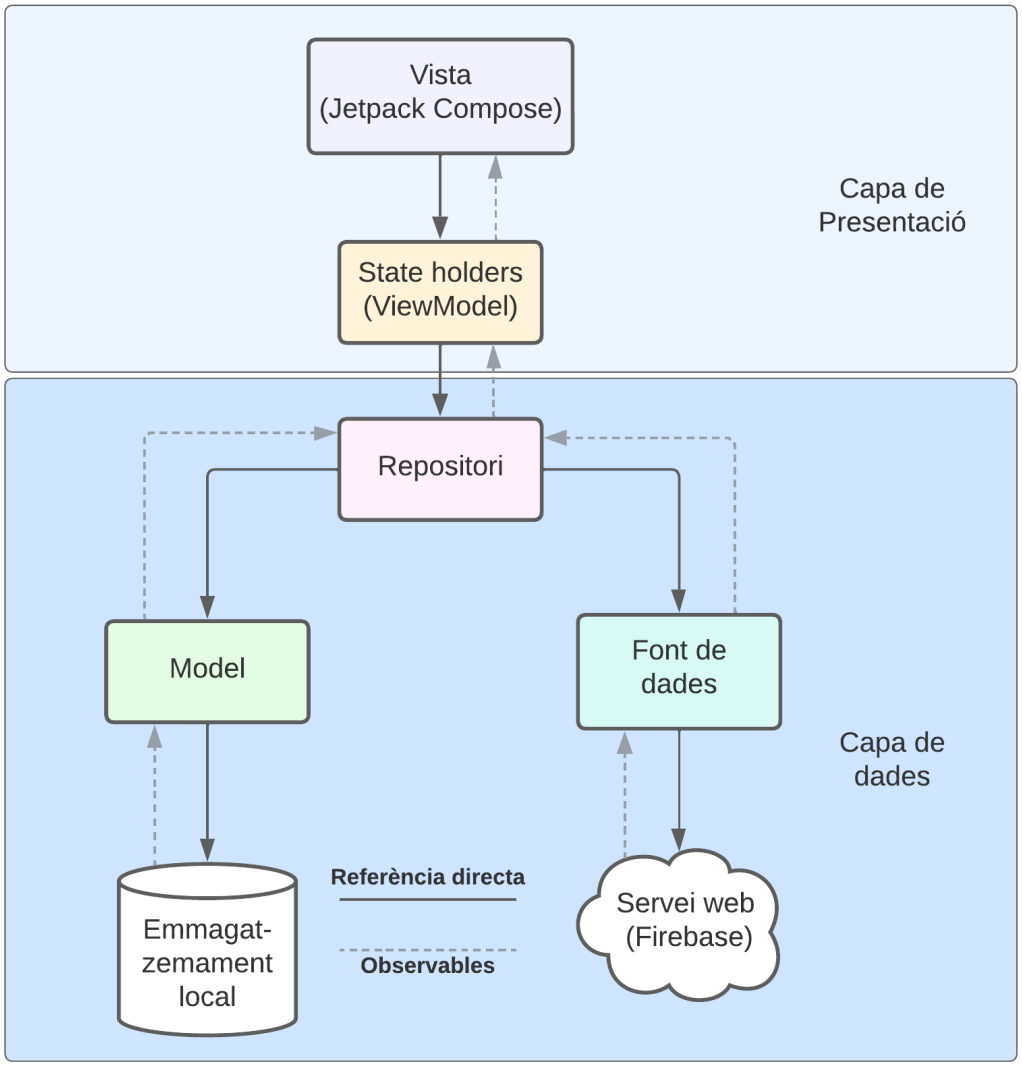

Figura 4.1: Diagrama de l'Arquitectura de Pack4You

4.2 Diagrama de classes

Una vegada definides les relacions entre les capes i components de Pack4You, es procedirà a explicar les relacions entre els diferents objetes (models) que composen l'aplicació. Açò ho farem mitjançant el llenguatge UML [13], desenvolupant amb ell un Diagrama de Classes UML [19].

UML (Unified Modeling Language) és el llenguatge de modelatge estàndard utilitzat a l'enginyeria de software per a visualitzar i dissenyar un sistema software **abans** d'escriure cap línia de codi. Amb ell poden desenvolupar-se diversos tipus de diagrames molt útils en la fase de disseny d'un sistema, però en aquest cas anem a centrar-se en el Diagrama de classes.

Segons Gene Zeiniss [19], un diagrama de classes "*descriu l'estructura d'un sistema mostrant les seues classes, atributs, mètodes i les relacions entre ells. És un vocabulari del sistema, una llengua comuna entre tots els membres de l'equip.*". En altres paraules, un diagrama de classes ens mostra de forma visual els atributs de totes les classes (models, entitats) que composen el nostre sistema i les interaccions entre elles. És molt útil treballar amb un diagrama de classes, doncs es pot entendre de forma molt visual com funcionen les relacions entre entitats i tenir-les clares abans de programar.

A continuació podem observar el Diagrama de classes de Pack4You (figura 4.2):

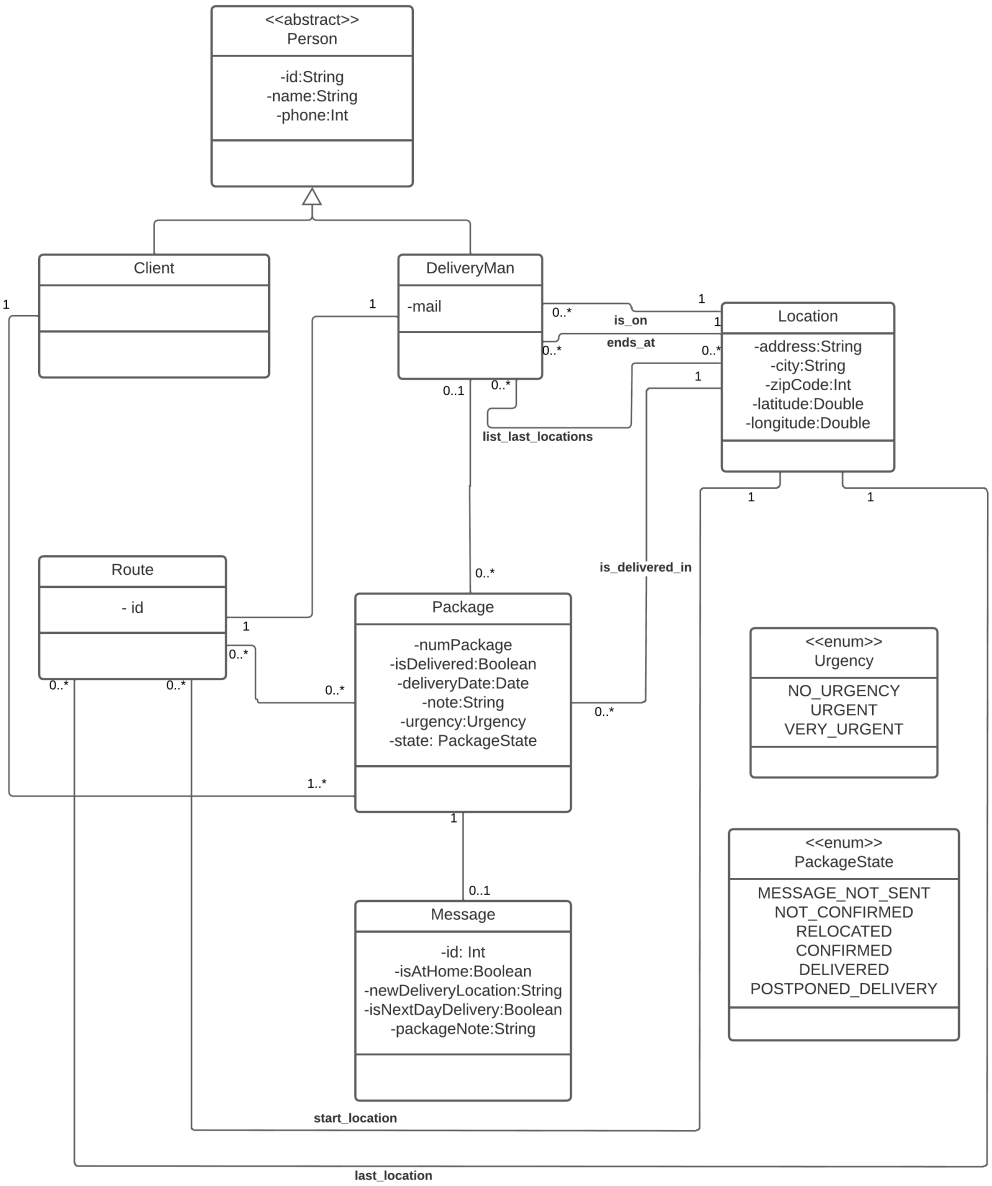

Figura 4.2: Diagrama de classes de Pack4You

Com es pot veure, tenim nou classes diferents, de les quals dos són *Enum* (classes que consisteixen en una llista de constants que defineixen un nou tipus de dades) i una és abstracta (no es poden crear objectes d'aquesta classe).

Cal mencionar que hi ha tres classes de les quals no es comentarà la seua funció i relacions en aquest apartat de la memòria. Es farà a l'apartat 4.4. Aquestes classes són *DeliveryMan*, *Package* i *Client*.

Les altres classes de dades que composen Pack4You són [1]:

- **Person:** Classe abstracta d'on hereten *DeliveryMan* i *Client*.
- **Location:** Aquesta classe ens permet descriure una localització. Els seus atributs són:
  - address -> Cadena de text que representa la localització (per exemple, carrer Arquitecte Arnau 25, València).
  - city -> Ciutat on se situa la localització.
  - zipCode -> Codi postal de la localització.
  - latitude -> Latitud de la localització.
  - longitude -> Longitud de la localització.

  Aquesta classe té dues relacions simples amb *DeliveryMan* per indicar on es troba i on finalitza aquest i una relació molts a molts per representar la llista d'últimes localitzacions del repartidor. També té una relació simple amb *Package* que representa on ha de ser entregat aquest, així com dues relacions simples més amb *Route* per indicar on comença i on acaba aquesta ruta. Aquesta última localització és la que l'administrador haurà decidit, però el repartidor la podrà canviar elegint una última localització diferent.
- **Route:** Aquesta classe és la que descriu el recorregut a seguir pel repartidor. A banda de les relacions simples amb *Location* ja explicades, també té una relació simple amb *DeliveryMan*, que defineix el repartidor al que aquesta ruta està assignada, i una relació molts a molts amb *Package*. Aquesta indica els paquets que conformen la ruta.
- **Message:** Classe que representa el missatge que un client enviaria des de la seua part del sistema. Conté aquests atributs:
  - id -> Identificador del missatge.
  - isAtHome -> Valor booleà que indica si el client està a casa o no.
  - newDeliveryLocation -> Cadena de text que representa una nova adreça on entregar el paquet. Aquest atribut no serà buit quan el client comunique que no és possible entregar el paquet en l'adreça original i en done una nova.
  - isNextDayDelivery -> Valor booleà que descriu si el paquet haurà de ser posposat al dia següent o no.
  - packageNote -> Cadena de text que permet al client escriure una nota addicional el mateix dia de l'entrega.

[1] S'ha omès l'explicació d'alguns atributs per l'obvietat de la funció d'aquests.

- **Urgency:** Aquesta classe *Enum* representa els diferents nivells d'urgència que un paquet pot tenir. Aquesta classe pot adoptar els valors *NO_URGENCY* per a les entregues estàndar, *URGENT* per a les entregues ràpides i *VERY_URGENT* per a les entregues urgents.
- **PackageState:** Classe *Enum* que indica els diferents estats en els que un paquet es pot trobar. Aquests són:
  - MESSAGE_NOT_SEND -> El repartidor encara no està suficientment prop per a que el missatge s'haja enviat.
  - NOT_CONFIRMED -> El missatge ha sigut enviat al client, però aquest encara no ha respòs.
  - RELOCATED -> El client ha respòs el missatge indicant que no és possible l'entrega en la direcció original del paquet, i n'ha donat una alternativa.
  - CONFIRMED -> El client ha indicat que és possible l'entrega en la direcció original del paquet.
  - DELIVERED -> El paquet ja ha sigut entregat.
  - POSTPONED_DELIVERY -> No és possible l'entrega del paquet en el dia corresponent i aquesta ha sigut posposada al següent dia laboral.

4.3 Prototipat

Abans de començar aquesta secció, vull agraïr enormement a Esther Frasquet Ricarte el desenvolupament dels prototips de Pack4You, tant els *wireframes* com els prototips d'alta fidelitat. Treballant en equip i juntant les meues idees i les seues propostes com a dissenyadora, s'ha aconseguit aquesta Interfície d'Usuari.

Ara sí, primerament cal definir què és el prototipat. Segons Denise Sánchez i Carmen Gerea [20], el prototipat *"és una tècnica que permet realitzar i materialitzar diverses idees de solucions propostes en un projecte de disseny o redisseny de productes i serveis."*. Així doncs, podem entendre que en el disseny d'una aplicació, el prototipat és fonamental. Aquest ajuda, tant a usuaris com a desenvolupadors, a fer-se una idea de com serà l'aplicació final i debatre sobre les funcionalitats que finalment aquesta tindrà.

Per suposat, tots els prototips no són iguals i no representen l'aplicació final de la mateixa manera. Podem distinguir entre tres tipus de prototips:

- **Wireframe:** Un wireframe no és més que un esbós de les principals funcionalitats de l'aplicació. Serveix per a posicionar els elements clau i fer-se una idea de la distribució d'aquests. No ha d'invertir-se molt de temps en el desenvolupament d'aquest tipus de prototips.
- **Prototips de baixa fidelitat:** Els prototips de baixa fidelitat, també anomenats lo-fi (Low-Fidelity) són aquells utilitzats per representar conceptes de

disseny ràpidament. També han de desenvolupar-se ràpidament, sense invertir molts recursos en ells. La diferència amb els Wireframes és que els prototips de baixa fidelitat sí utilitzen elements que estaran a l'aplicació final, com les formes definitives dels diversos components.

- **Prototips d'alta fidelitat:** Els prototips d'alta fidelitat, o hi-fi (High-Fidelitity) són, com el seu nom indica, els més fidels a l'aparença final de l'aplicació. Són creats en les etapes finals del disseny i quan es té un gran enteniment del producte. Els recursos que s'han d'emprar per a desenvolupar els prototips d'alta fidelitat són majors als emprats per a desenvolupar els prototips lo-fi o els wireframes. Els usuaris poden fer-se una idea molt més realista de com serà l'aplicació final.

  A la figura 4.3 podem observar un exemple de com una mateixa pantalla ha evolucionat des de Wireframe (esquerra) a prototip de baixa fidelitat (centre) i, finalment, a prototip d'alta fidelitat (dreta) [2].

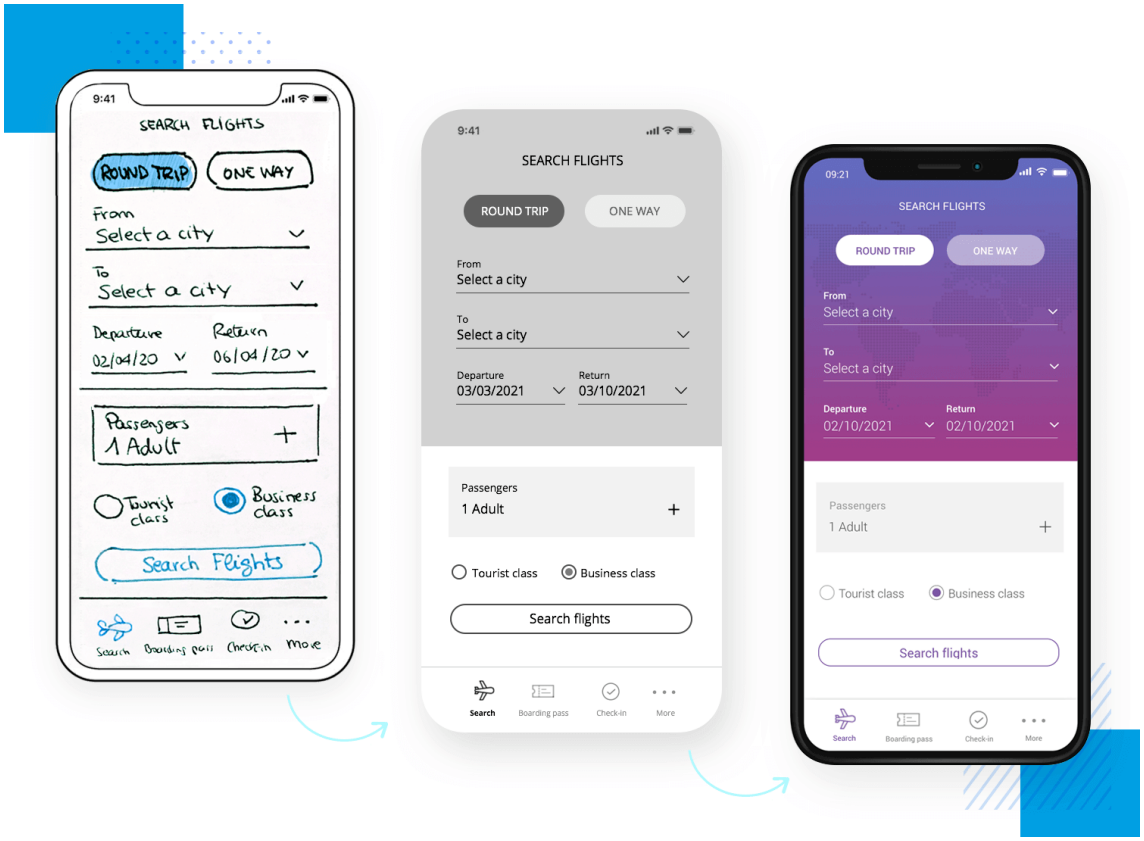

Figura 4.3: Diferència entre Wireframe, prototip de baixa fidelitat i prototip d'alta fidelitat

Una vegada explicada la importància del prototipat a un projecte d'enginyeria del software i els diferents tipus d'aquest que existeixen, mostrarem els prototips desenvolupats per a Pack4You.

Al principi del projecte es van realitzar ràpidament uns wireframes i, en una etapa més avançada, es van realitzar prototips d'alta fidelitat. Cal mencionar que

[2]Imatge disponible a https://assets.justinmind.com/wp-content/webp-express/webp-images/uploads/2021/01/low-to-high-fidelity-prototype.png.webp

els *wireframes* només van ser realitzats per representar les pantalles principals de l'aplicació.

Per a realitzar aquests prototips, s'ha utilitzat Figma [21], una ferramenta de disseny molt potent que també permet la col·laboració amb diferents dissenyadors.

4.3.1. Pantalla d'inici de sessió

Aquesta pantalla es va dissenyar per a que l'usuari emplenara ràpidament el seu correu i contrassenya per a accedir al seu compte. Com podem observar (figura 4.4), en aquesta pantalla està present el logo de l'aplicació i icones en cada camp a emplenar per ajudar a l'usuari a entendre més ràpidament la informació que hauria d'haver en cadascun d'ells. També tenim un botó gran d'Inici de sessió per a que el repartidor sàpiga fàcilment on polsar per poder accedir al seu compte.

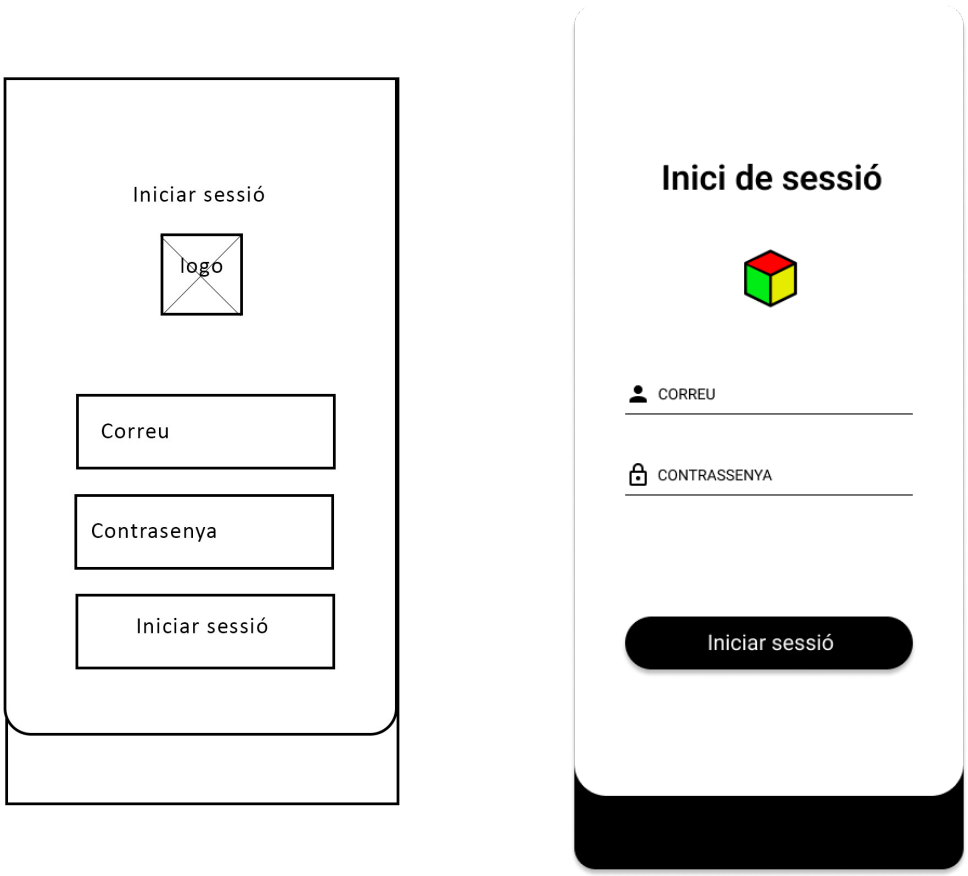

Figura 4.4: Pantalla d'inici de sessió

4.3.2. Pantalla principal

Aquesta és la pantalla principal de l'aplicació (figura 4.5). En ella podem observar dalt del tot la Top Bar de l'aplicació. Aquesta conté un botó per desplegar un menú d'opcions i altre per consultar els paquets ja entregats.

En la part inferior de la Top Bar tenim un mapa amb els diferents paquets ubicats en aquest. Parlarem més a fons d'aquesta part de l'aplicació més endavant, en l'apartat 4.3.3.

Més avall comença la llista arrossegable on es realitzarà la gestió dels paquets. Primerament veiem el botó que, en polsar-lo, ens permet elegir l'algorisme d'ordenació. Al costat d'aquest botó podem observar l'algorisme usat actualment.

Després tenim tots els paquets que composen la ruta a recórrer. Cada targeta de paquet conté la seua referència, la seua direcció, el nom del client i el tipus d'entrega (estàndar, ràpida o urgent).

En la part inferior de la targeta de cada paquet trobem l'estat d'aquest. Encara que en la versió actual de l'aplicació aquesta funcionalitat no tinga ús, en pròximes versions d'aquesta sí en tindrà. Ja es té la Interfície d'Usuari preparada. L'estat d'un paquet canvia quan el client envia una resposta (si està a casa, l'estat canviarà a *confirmat*. Si no està, a *cancel·lat* i, si demana entregar-lo a altra direcció, l'estat passarà a *canviat*).

També podem observar com les targetes es poden arrossegar cap a l'esquerra o la dreta per eliminar un paquet o marcar-lo com a entregat. Alhora d'eliminar-lo, un diàleg apareixerà per fer-nos decidir si eliminar el paquet definitivament o només posposar-lo.

Finalment, tenim al cap davall el temps que es tarda en recórrer la ruta i el botó d'Iniciar ruta.

Figura 4.5: Pantalla principal

4.3.3. Mapa

Ací (figura 4.6) es troba el mapa on es pot, com ja s'ha dit, veure la localització dels diferents paquets que composen la ruta a recórrer, així com el camí a seguir segons l'algorisme seleccionat. Aquest camí canviarà dinàmicament depenent de l'algorisme i les operacions CRUD realitzades sobre els paquets.

Si polsem sobre un punter, obtindrem més informació sobre el paquet corresponent. En la part inferior dreta podem observar el temps que es tardarà a recórrer la ruta i el botó per iniciar-la.

Finalment, a la part inferior veiem com sobresurt una fracció de la llista de paquets, indicant que podem arrossegar-la cap amunt per accedir a aquesta llista.

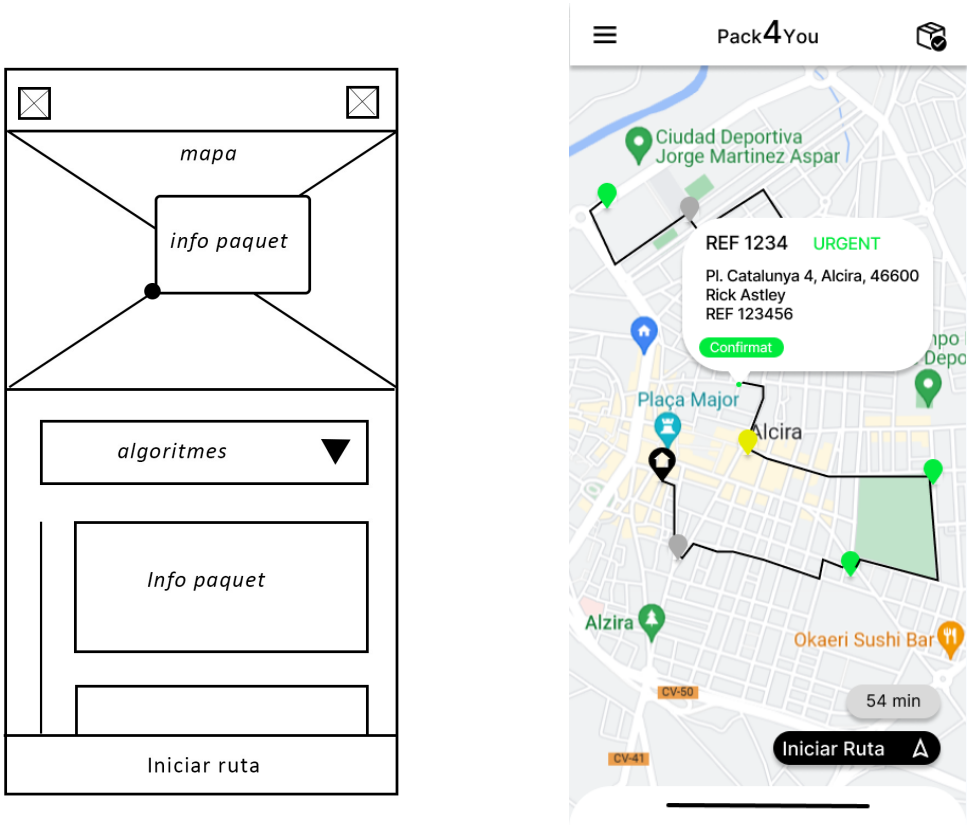

Figura 4.6: Mapa amb els paquets de la ruta posicionats

4.3.4. Menú d'opcions

Aquest menú d'opcions (figura 4.7) ens permet accedir a diverses funcionalitats de l'aplicació, així com tancar la sessió si cal. A la part superior tenim una icona amb una "X" per poder tancar aquest menú desplegable.

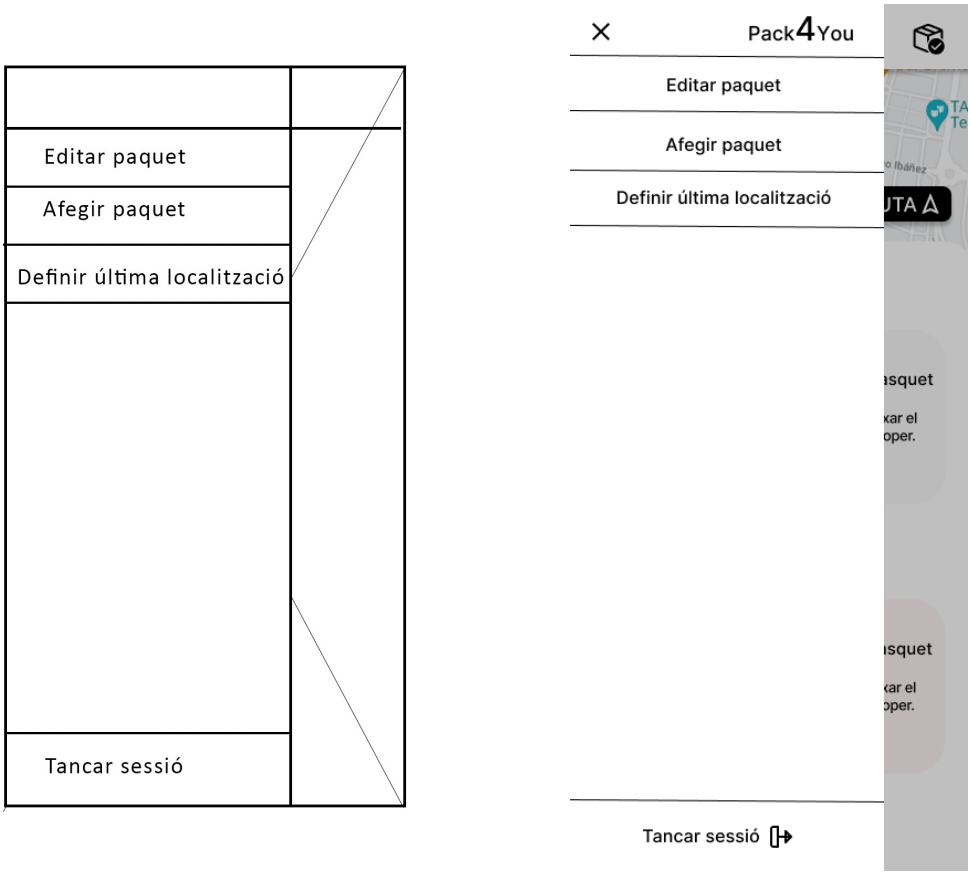

Figura 4.7: Menú d'opcions

4.3.5. Editar paquet

En aquesta pantalla (figura 4.8) podrem elegir un paquet per editar la seua informació. Es pot observar com, en cada targeta de paquet, tenim la seua referència, el client i la direcció d'aquest. Així, es pot elegir el paquet desitjat sense equivocar-se.

En polsar el botó "Editar paquet xxxx", anirem a una pantalla que és idèntica a la que vorem a l'apartat 4.3.6, però amb les dades ja emplenades.

4.3.6. Afegir paquet

Aquest prototip suporta dos casos d'ús: Afegir paquet i Editar paquet.

En la figura 4.9 podem observar com tenim un mapa amb un marcador, el qual estarà posicionat a l'adreça escrita al camp "Direcció". També podem veure el camp "Destinatari" (on posarem el nom d'aquest) i el camp "Entrega" (on, amb un desplegable, decidirem si el paquet té entrega estàndard, ràpida o urgent). En polsar al botó "Afegir", el paquet formarà part de la nostra ruta (o editarem la informació del paquet desitjat).

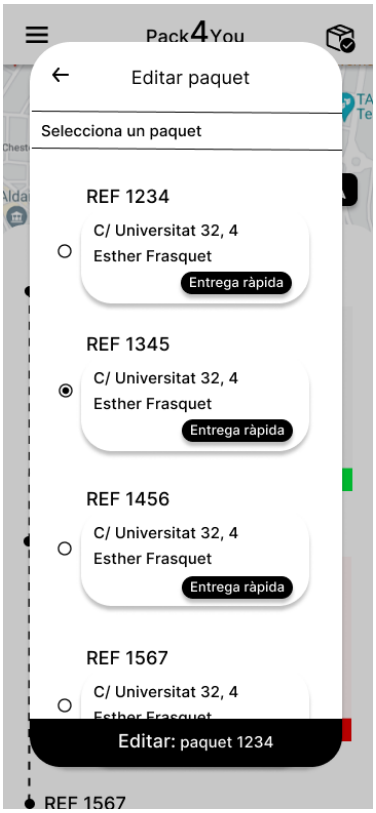

Figura 4.8: Elegint un paquet per editar-lo

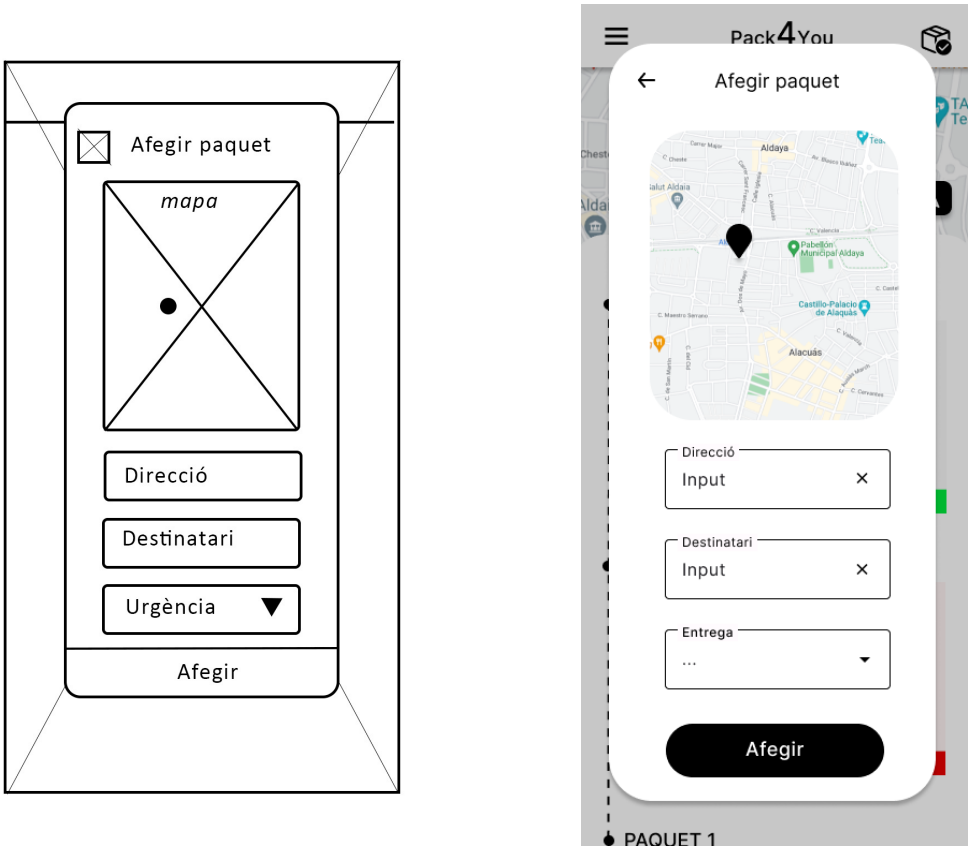

Figura 4.9: Afegir/Editar paquet

4.3.7. Definir última localització

Aquesta funcionalitat permet al repartidor elegir el punt on acabarà el repartiment. Com podem observar (figura 4.10), tenim diversos punts que ja hauran sigut definits pel repartidor i es poden seleccionar per ser l'última localització. Al costat de cadascun d'aquests ítems veiem una icona que permetrà a l'usuari eliminar aquest punt. L'últim camp de la llista és un camp editable on es pot escriure una direcció nova. Si aquesta és reconeguda pel sistema, es convertirà en un nou punt definit i serà elegible com a punt per acabar la ruta.

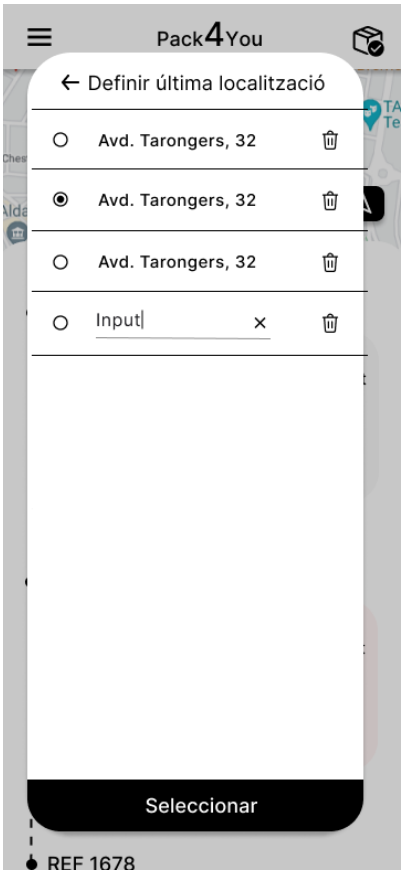

Figura 4.10: Definir última localització

4.3.8. Ruta iniciada

Finalment, aquesta pantalla (figura 4.11) representa el cas d'ús d'iniciar ruta. Com ja s'ha dit, aquesta característica ens permet recórrer una vista prèvia de com seria la ruta a seguir.

Podem observar com la Top Bar es manté com a la pantalla principal, amb la diferència que ara tenim una fletxa per poder fer enrere i tornar a aquesta.

Més avall trobem un mapa interactiu amb un marcador negre que indica la posició del paquet actual i una porció de la ruta a seguir.

Després trobem informació sobre el paquet en forma d'una targeta similar a l'emprada en la llista de paquets de la pantalla principal.

Finalment, dues fletxes permeten a l'usuari anar al paquet anterior (esquerra) i al següent (dreta). En la part inferior està el botó de marcar com entregat aquest paquet. Una vegada polsat, l'aplicació ens mostrarà la informació del següent paquet.

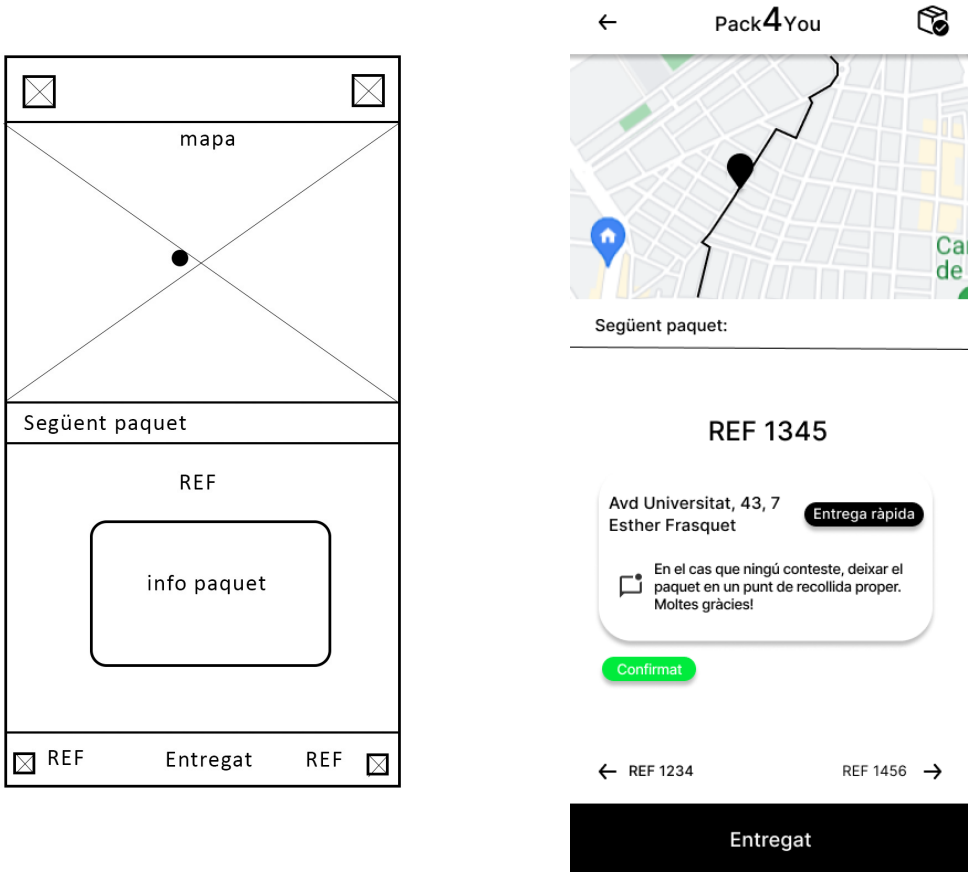

Figura 4.11: Ruta iniciada

Per concluir aquesta secció, a la figura 4.12 podem trobar un diagrama de navegació que ens proporciona informació sobre, com el seu nom indica, la navegació entre les diferents figures mostrades. S'han suprimit les relacions que indicaven navegació cap enrere (com les fletxes enrere per tornar a la pantalla principal) per a simplificar aquest diagrama.

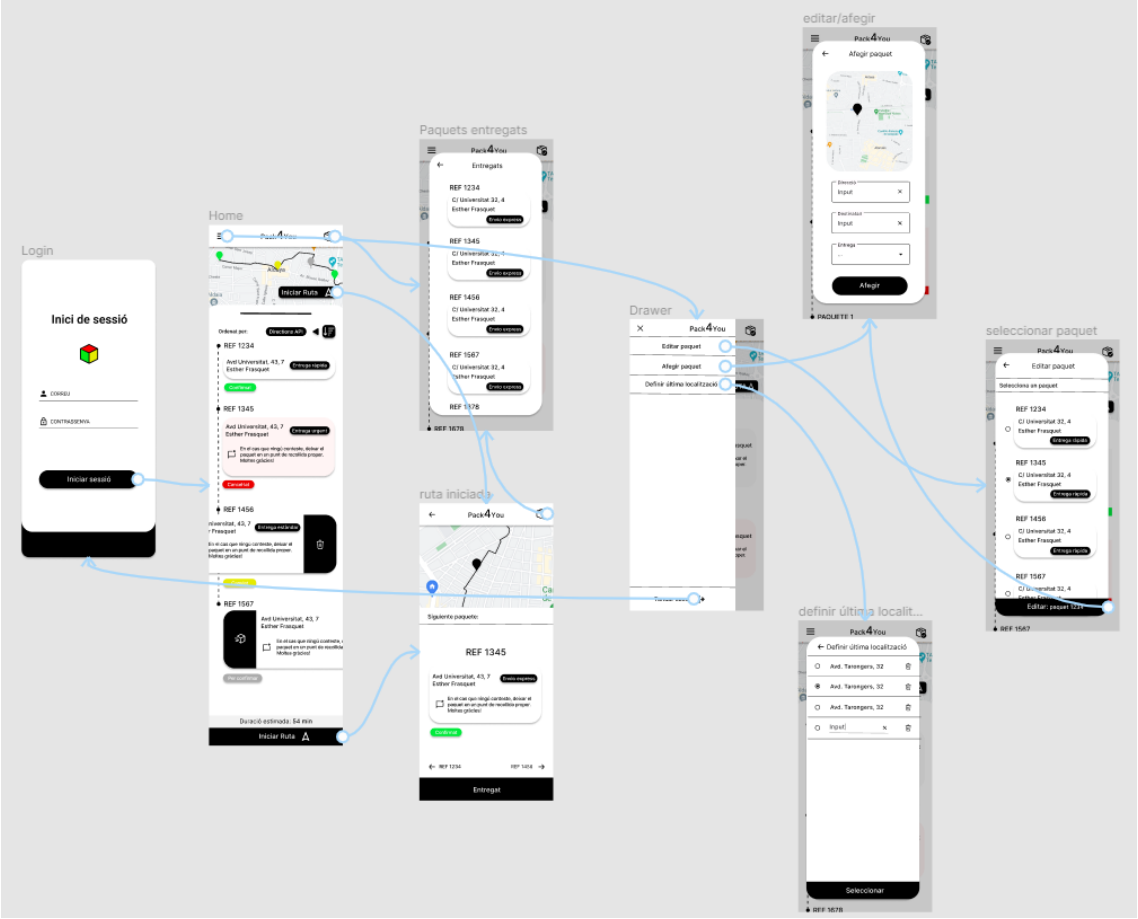

Figura 4.12: Diagrama de navegació de Pack4You

4.4 Disseny de la base de dades

La base de dades en temps real (Firestore [22]) està suportada per Firebase [17]. Aquesta és una base de dades NoSQL [23], el que ens permet guardar grans quantitats d'informació i, al no ser una base de dades relacional (és a dir, no s'ha de crear un arbre de referències per poder obtenir tota la informació desitjada), guanyar en eficiència i escalabilitat a canvi d'ocupar molt més espai d'emmagatzematge (ja que estem duplicant la informació).

Firestore utilitza un array d'objectes JSON, pel que alhora d'executar operacions CRUD sobre la base de dades, es crida a funcions de la API de Firebase i, automàticament, transforma un objecte local (com *Package*) a un objecte JSON. Com podem observar en la figura 4.13, a Pack4You només tenim tres objectes JSON.

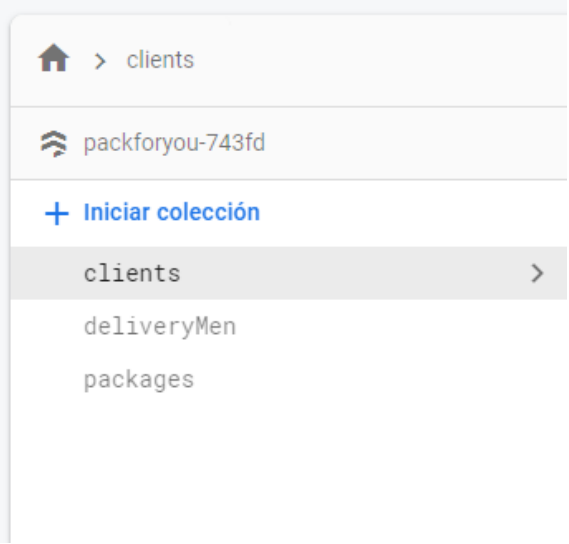

Figura 4.13: Base de dades a Firestore. Cal destacar que el nom dels objectes és en plural perquè són col·leccions

A continuació, va a descriure's cadascun dels objectes presents a la base de dades. Cal tenir en compte que alguns atributs d'aquests objectes estan composats pels objectes presents al diagrama de classes, el qual podem trobar a la figura 4.2.

- **Client:** Aquest objecte emmagatzema la informació d'un client present a Pack4You. El composen aquests atributs:
  - id -> Identificador del client.
  - name -> Nom del client.
  - phone -> Número de telèfon del client.
- **DeliveryMan:** Aquest objecte conté la informació d'un repartidor registrat a Pack4You. És l'objecte d'on obtenim tota la informació quan s'inicia la sessió. Els seus atributs són:
  - id -> Identificador del repartidor.
  - currentLocation -> Atribut de tipus Location que indica la localització actual del repartidor.
  - lastLocation -> Atribut de tipus Location que ens indica l'última localització elegida pel repartidor.
  - lastLocationList -> Col·lecció d'objectes Location que ens diu les últimes localitzacions registrades pel repartidor.
  - mail -> Correu del repartidor.
  - name -> Nom del repartidor.
  - packages -> Col·lecció d'objectes de tipus Package que ens dóna informació sobre tots els paquets assignats a aquest repartidor.
  - phone -> Telèfon del repartidor.

  - route -> Objecte de tipus Route que ens indica la ruta que el repartidor seguirà.

- **Package:** Aquest objecte conté informació sobre un paquet registrat a l'aplicació Pack4You. Les seues propietats són:

  - numPackage -> Identificador del paquet.
  - client -> Objecte tipus Client a qui el paquet ha de ser entregat.
  - isDelivered -> Indica si el paquet ha sigut entregat o no.
  - deliveryDate -> Data d'entrega del paquet.
  - location -> Objecte de tipus Location que indica l'adreça on el paquet ha de ser entregat.
  - message -> Un objecte de tipus Message. Aquest tindrà informació sobre la resposta del client al missatge que el sistema li envia per saber si estarà o no a casa.
  - note -> Nota que el client ha deixat junt al paquet.
  - state -> Estat del paquet.
  - urgency -> Urgència del paquet.

4.5 Tecnologies utilitzades

Per finalitzar aquest capítol, es descriuran les tecnologies utilitzades al projecte.

4.5.1. Android Studio

Android Studio [24] és el IDE (Integrated Development Environment) oficial per a desenvolupament en Android. Va ser desenvolupat i és mantès per JetBrains [26]. Algunes de les característiques més destacables d'aquesta plataforma són:

- **Emulador integrat:** És possible provar el teu codi a l'instant gràcies al ràpid emulador que Android Studio té integrat.
- **Perfilador en temps real:** Pots obtenir estadístiques en temps real sobre la teua aplicació com, per exemple, ús de CPU i memòria o activitat de la xarxa.
- **Editor de codi intel·ligent:** Android Studio compta amb un editor de codi amb suport per a diversos llenguatges com Java, Kotlin o C/C++. A més, també suporta completació de codi, que t'ajuda a ser més ràpid i eficient escrivint codi, ja que et suggereix noms de classes o mètodes per completar la línia que estàs escrivint.

A la figura 4.14 es pot observar aquest projecte executant-se a Android Studio.

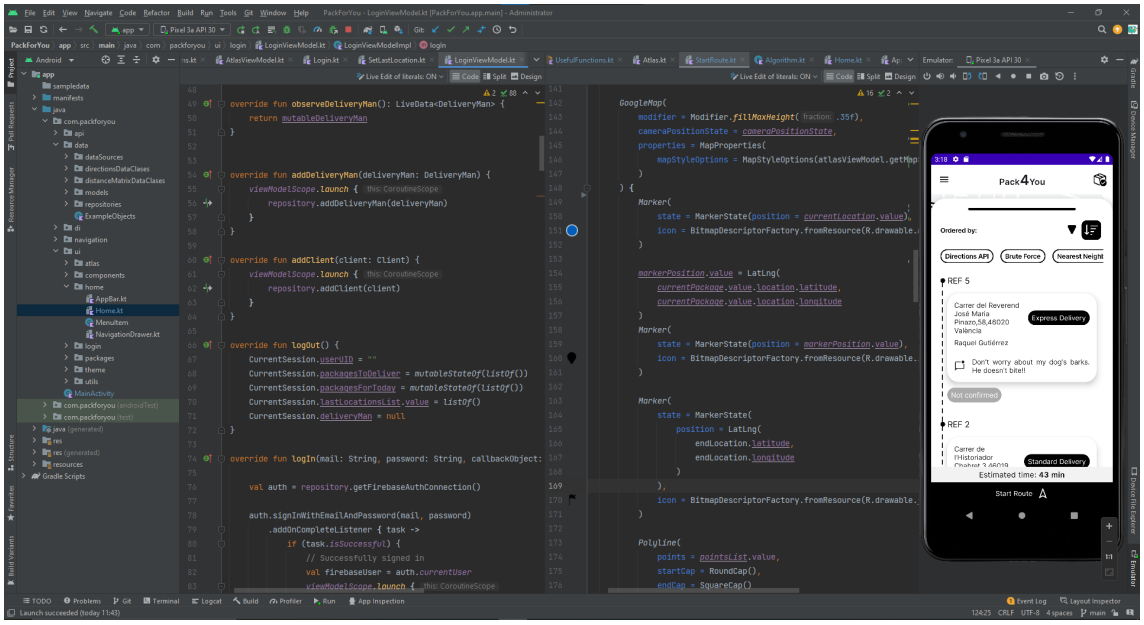

Figura 4.14: Pack4You executant-se a Android Studio

4.5.2. Kotlin

Kotlin [25] és un llenguatge de programació orientat a objectes creat per JetBrains que està dirigit, principalment, al mercat mòbil. És multiplataforma i completament interoperable amb Java. Açò vol dir que un sistema pot funcionar perfectament tenint codi Kotlin i Java alhora. És molt útil per migrar codi entre aquests llenguatges, ja que el sistema pot seguir funcionant sense problema durant la seua migració. A la figura 4.15 tenim el logo de Kotlin.

Figura 4.15: Logo de Kotlin

4.5.3. Jetpack Compose

Segons la web oficial de desenvolupadors d'Android [15], *"Jetpack Compose és un conjunt d'eines d'Android que et permet construïr una Interfície d'Usuari nativa. Simplifica i accelera el desenvolupament de les Interfícies d'Usuari en Android."*. La versió estable 1.0 va ser llançada el 28 de juliol de 2021, i ha suposat un gran canvi per al desenvolupament mòbil. Abans, les Interfícies d'Usuari en Android es desenvolupaven utilitzant XML [27]. Amb Jetpack Compose (figura 4.16) es

necessiten menys línies de codi i es té accés a ferramentes més potents i APIs de Kotlin per fer aquestes Interfícies d'Usuari.

Figura 4.16: Logo de Jetpack Compose

4.5.4. Firebase

Firebase [17] és una plataforma en el núvol que et permet la creació i sincronització de projectes al núvol. La seua propietària és Google, i ofereix múltiples ferramentes, com una base de dades en temps real, serveis d'autenticació o consulta d'estadístiques. En el cas de Pack4You, aquesta plataforma s'ha utilitzat com a base de dades i servei d'autenticació. A la figura 4.17 es pot observar el logo de Firebase.

Figura 4.17: Logo de Firebase

4.5.5. Git

Git [28] és un software de control de versions open source [29]. Va ser desenvolupat per Linus Torvalds (creador de Linux) i actualment és el controlador de versions més utilitzat arreu del món. Gràcies a aquest software, és possible desenvolupar software amb gent de qualsevol part del planeta molt fàcilment. El logo de Git podem observar-lo a la figura 4.18.

Figura 4.18: Logo de Git

CAPÍTOL 5
Desenvolupament de la solució proposada

En aquesta secció es parlarà del propi desenvolupament de Pack4You. Es passarà del disseny de la solució descrit abans a la implementació real d'aquesta. Per a açò, primerament es tractarà l'estructura pròpia del projecte, inspirada, com ja s'ha explicat, en l'arquitectura MVVM. Seguidament, es parlarà dels patrons de disseny utilitzats a Pack4You, i de com s'ha seguit una arquitectura Clean [30] per al desenvolupament d'aquest projecte. Més tard, es profunditzarà en les funcions composables de Jetpack Compose, element indispensable per poder crear la Interfície d'Usuari. Després, es descriurà el desenvolupament de les característiques relacionades amb els mapes i el paper de Directions API en aquests. Finalment, es discutirà sobre la implementació dels diferents algorismes d'ordenació de paquets presents en aquesta versió del projecte.

5.1 Estructura del projecte

L'estructura d'aquest projecte, com s'ha mencionat abans, està dissenyada per seguir l'arquitectura MVVM. La figura 5.1 descriu com és aquesta estructura en Android Studio.

Com es pot observar, el projecte compta amb diversos directoris. Cadascun compleix un propòsit diferent:

- **api:** Aquesta carpeta conté les classes relacionades amb la comunicació amb les diferents APIs que Pack4You utilitza.
- **data:** Directori que representa la capa de dades de l'arquitectura MVVM. Com es pot observar, aquest conté les fonts de dades i els repositoris (carpetes *dataSources* i *repositories*, respectivament), els models descrits al punt 4.2 (directori *models*) i les classes de dades necessàries per a la utilització de Directions API i Distance Matrix API (directoris *directionsDataClasses* i *distanceMatrixDatalClasses*, respectivament. Parlarem d'aquesta última interfície a la secció 5.5).

- **di:** Carpeta que conté dues classes relacionades amb la Injecció de Dependències [31] (*Dependency Injection* en anglès). En la secció 5.2 es parlarà més a fons d'aquest patró de disseny.

- **ui:** Directori que representa la capa de presentació de l'arquitectura MVVM. Conté diverses carpetes (*atlas*, *home*, *login* i *packages*) que corresponen a les diferents parts en les que s'enfoca el desenvolupament de la Interfície d'Usuari. Cadascuna d'aquestes carpetes compta amb una o diverses vistes (View) i un ViewModel. La carpeta *ui* també inclou els directoris *components* i *utils*, que contenen components visuals que s'utilitzen en diverses pantalles i funcions que utilitzen diverses parts de la Interfície d'Usuari, respectivament. El directori *navigation* compta amb les classes necessàries per permetre la navegació de l'aplicació. Finalment, la carpeta *theme* defineix alguns estàndards per a que l'aplicació seguisca una mateixa línia visual (per exemple, la forma de certs botons, el tamany i estil dels títols, etc).

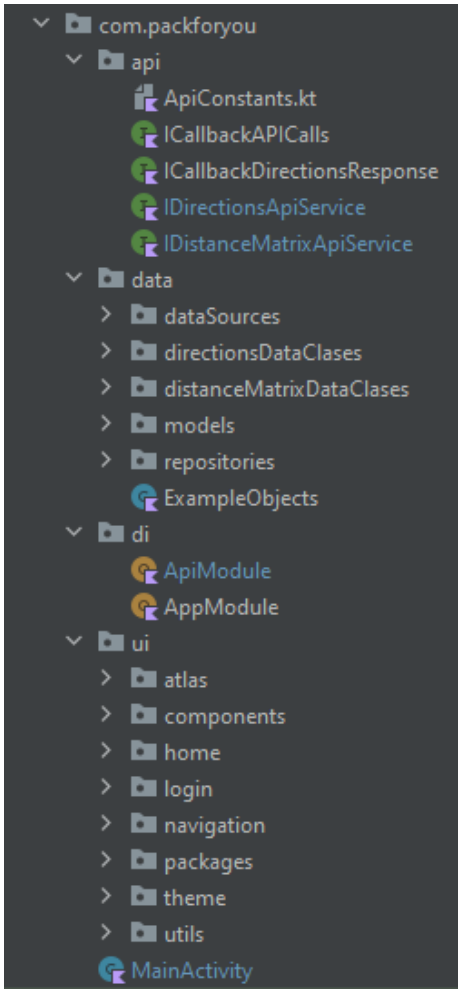

Figura 5.1: Estructura del projecte

5.2 Patrons de disseny i arquitectura Clean

Un patró de disseny és una tècnica utilitzada per resoldre problemes comuns en el desenvolupament de software. Aquests han sigut ja provats diverses vegades i s'ha comprovat que són efectius. És per açò que és molt útil aplicar-los als problemes que puguen sorgir ja que, d'aplicar-se correctament, és segur que el problema està resolvent-se de manera adequada i escalable.

A Pack4You s'han aplicat quatre patrons de disseny: arquitectura MVVM [14], injecció de dependències [31], Singleton [32] i el patró observador [33]. Cal dir que el patró de disseny MVVM ja s'ha explicat al punt 4.1, pel que s'ha omès en aquest apartat.

5.2.1. Injecció de dependències

La injecció de dependències és un patró de disseny que consisteix a passar directament com a arguments d'una classe les dependències que aquesta puga tenir. Aquests arguments venen definits per una interfície, pel que la classe depenent no coneix la implementació de la dependència. Només sap que existeix un mètode que, en cridar-lo, li retorna la resposta desitjada.

Per exemple, si una classe necessita una API concreta, en lloc de crear la referència a aquesta en la pròpia classe (i, per tant, cada vegada que s'instancie un objecte d'aquesta classe, es cree una nova referència/connexió a aquesta API), podem únicament passar-la com a paràmetre i així evitar la creació massiva de referències/connexions a aquesta API, a banda de desacoblar aquesta classe i la seua dependència, ja que la primera no sap d'on procedeix la segona. A més, si en algun moment volem canviar la implementació d'aquesta dependència, només hem de canviar el codi en el lloc on s'instancia aquesta, no en totes les vegades que caldria crear l'objecte en cada classe depenent d'ella.

A més a més, aquest patró de disseny també és molt útil per a provar el codi. Per exemple, si es vol provar el funcionament d'una classe que conté una dependència amb una API, però no volem fer connexió amb aquesta cada vegada (perquè porta molt de temps), podem injectar-li una API feta a mà que retorne el valor que nosaltres vulguem. El test tardarà molt menys i seguirà provant-se el que nosaltres desitgem, ja que en ningun moment volem provar aquesta API.

En la figura 5.2 es pot observar com s'utilitza aquest patró de disseny per proveir a "PackagesAndAtlasRepositoryImpl" de les dependències necessàries.

```
class PackagesAndAtlasRepositoryImpl(
    private val dataSource: IFirebaseRemoteDatabase,
    private val directionsApiService: IDirectionsApiService,
    private val distanceMatrixApiService: IDistanceMatrixApiService
) : IPackagesAndAtlasRepository {
```

Figura 5.2: Injecció de dependències a Pack4You

A Pack4You, aquest patró s'ha implementat amb l'ajuda de Hilt [34]. Aquesta ferramenta permet automatitzar la creació de les classes necessàries per a la injecció de dependències. Com es pot observar a la figura 5.3, utilitzant l'anotació "*@Provides*" estem indicant que, cada vegada que es necessite la interfície "IPackagesAndAtlasRepository", crearem una instància de l'objecte "providePackagesAndAtlasRepository" amb els paràmetres donats. D'aquesta manera, estem independitzant la pròpia creació de la dependència, fent que les demés classes puguen reutilitzar-se sense haver de dur amb elles la instanciació de les dependències.

```
@Provides
fun providePackagesAndAtlasRepository(retrofit: Retrofit): IPackagesAndAtlasRepository {
    return PackagesAndAtlasRepositoryImpl(
        AppModule.provideFirebaseDataSource(),
        provideDirectionsApiService(retrofit),
        provideMatrixApiService(retrofit)
    )
}
```

Figura 5.3: Automatització de la injecció de dependències amb Hilt

5.2.2. Singleton

El patró de disseny Singleton permet obtenir sempre la mateixa instància d'un objecte. Limita la creació d'objectes de la classe a la que s'aplique aquest patró a un sol objecte.

Aquest mètode és molt útil per a, per exemple, controlar l'accés a un recurs compartit, com pot ser una base de dades. També és útil per a evitar el malbaratament de memòria creant nous objectes innecessaris. Aquest últim cas pot donar-se, per exemple, creant diverses instàncies d'un repositori. Amb la mateixa instància d'aquest pots realitzar les mateixes accions que amb diverses instàncies d'aquesta classe. No és útil fer-ne més d'una.

Com es pot observar a la figura 5.4, a Kotlin és molt fàcil implementar aquest patró. Simplement s'ha de substituir la paraula "*class*" per "*object*" en la capçalera de la definició de la classe. D'aquesta manera, en tot el sistema només hi haurà una instància de la classe *CurrentSession*. D'haver-ne diverses, és possible que en algun moment es produïra alguna inconsistència i la informació mostrada al repartidor no siguera fiable. Altra manera d'implementar Singleton a Kotlin és simplement afegir l'anotació "*@Singleton*" sobre la classe que desitges.

```
object CurrentSession {
    var userUID = ""
    var deliveryMan: DeliveryMan? = null
    var algorithm = Algorithm.NOT_ALGORITHM
    lateinit var route: MutableState<Route>
    lateinit var packagesForToday: MutableState<List<Package>>
    lateinit var packagesToDeliver: MutableState<List<Package>>
    lateinit var lastLocationsList: MutableState<List<Location>>
    lateinit var travelTime: MutableState<Int>
}
```

Figura 5.4: Patró de disseny Singleton aplicat a l'objecte "*CurrentSession*"

5.2.3. Patró observador

El patró observador permet a una classe (subscriptor) observar els canvis en un objecte (publicador). D'aquesta manera, cada vegada que el publicador tinga algun canvi, aquest ho notificarà al subscriptor (o subscriptors) i, en rebre l'avís, aquest últim podrà actuar en conseqüència.

Aquest patró de disseny és molt útil en una gran quantitat de sistemes, ja que et permet reaccionar a diferents events del sistema (ja siguen provocats per l'usuari o per una entrada externa) d'una manera eficient i escalable. A més, el codi executat pel subscriptor estarà al component de l'arquitectura corresponent.

A Pack4You, aquest patró s'ha utilitzat per canviar dinàmicament la ruta dibuixada al mapa. S'aprofundirà més sobre la implementació d'aquest patró, utilitzant com exemple aquesta característica, a la secció 5.4.

5.2.4. Arquitectura Clean

A banda dels patrons de disseny, a Pack4You també s'ha aplicat el principi d'Inversió de Dependències present als principis SOLID [35]. Aquest consisteix a utilitzar interfícies entre cada component de l'arquitectura (entre el *ViewModel* i un repositori, per exemple. D'aquesta manera, un component no depèn de la implementació de l'altre. Només de la interfície que aquest implementa. Si en algun moment vol canviar-se la implementació de qualsevol capa, només s'ha de fer. No caldrà canviar res dels components que utilitzen aquesta nova implementació. A més a més, ara les capes internes no depenen de les externes, doncs, al no conéixer la procedència d'aquestes dependències (només saben que existeix cert mètode que els retorna el valor esperat), podem canviar tota la capa externa sense cap problema.

És per totes aquestes característiques que es pot dir que a Pack4You s'ha implementat una arquitectura Clean [30], aconseguint un producte amb components desacoblats, testable i molt escalable.

5.3 Jetpack Compose i les funcions composables

Com ja s'ha explicat a la subsecció 4.5.3, Jetpack Compose és un conjunt de ferramentes novedós per a desenvolupar les Interfícies d'Usuari en Android. Aquest es basa, fonamentalment, en les funcions anotades com a "*Composable*"

Les funcions composables són els elements que li donen sentit a Jetpack Compose. Totes les Interfícies d'Usuari són i estan composades per funcions *Composable*.

Aquestes funcions permeten al desenvolupador definir la IU de forma programàtica. Només es descriu com hauria de veure's la IU i es proveeixen les dependències necessàries i arguments desitjats, sense fer èmfasi en la inicialització d'aquest element o lligar-lo a un pare, com sí succeeix amb el desenvolupament d'Interfícies d'Usuari utilitzant XML.

A la figura 5.5 podem observar com seria el codi necessari per crear el botó que mostra o amaga els diferents algorismes disponibles a Pack4You, junt a la seua visualització en pantalla. Noteu que al codi tenim tant la part corresponent a la pròpia IU (*modifier, clip, padding...*) com la pròpia lògica del botó (*onClick*).

```
@Composable
fun SortButton() {
    Row { this: RowScope
        Box(
            modifier = Modifier
                .clip(RoundedCornerShape(10.dp))
                .background(Black)
                .padding(5.dp)
                .shadow(5.dp)
        ) { this: BoxScope
            IconButton(
                onClick = {
                    expanded.value = !expanded.value
                },
                modifier = Modifier.size(32.dp)
            ) {
                Icon(
                    painter = painterResource(id = R.drawable.ic_sort),
                    contentDescription = stringResource(
                        id = R.string.sort_button_description
                    ),
                    tint = White,
                    modifier = Modifier.fillMaxSize()
                )
            }
        }
    }
}
```

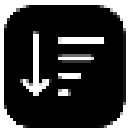

Figura 5.5: Exemple d'una funció composable i la seua visualització

D'aquesta manera és com es creen tots els elements que conformen la IU. Per a utilitzar aquest botó, com que és una funció, simplement cal cridar-la (figura 5.6).

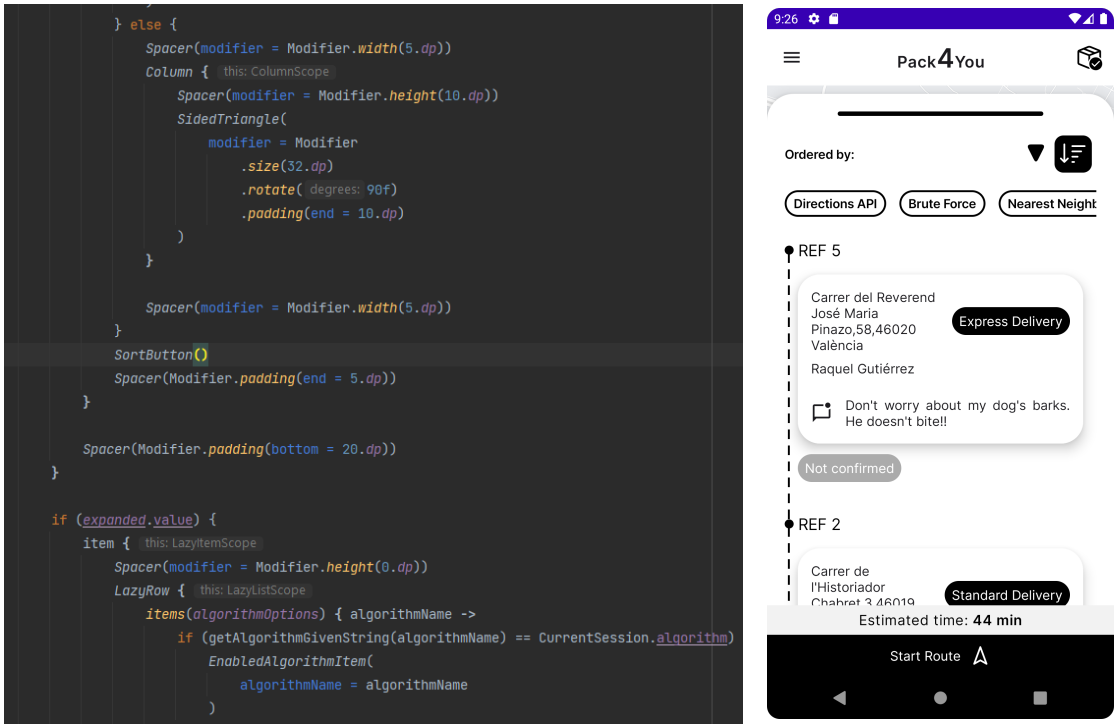

Figura 5.6: Funció composable siguent cridada. Aquesta és la pantalla final on aquest botó està present

5.4 Desenvolupament dels mapes i paper de Directions API

Una característica fonamental per a facilitar la vida d'un repartidor és poder visualitzar els paquets a repartir en un mapa i veure la ruta a seguir.

Per a implementar el posicionament de paquets i el mapa, a Pack4You s'ha utilitzat la funció composable "GoogleMap" proporcionada per Maps SDK for Android. Aquesta ens permet mostrar un mapa de la zona desitjada, amb el zoom que vulguem i posicionar diversos marcadors a les coordenades requerides. Aquests poden tenir la icona requerida i un missatge informatiu (també personalitzable) en polsar-la.

En quant a la visualització de la ruta al mapa i el canvi d'aquesta de forma dinàmica, he utilizat Directions API. Aquesta interfície (pertanyent a Google) et retorna un objecte JSON amb, entre altres dades, els diversos punts que composen la ruta a dibuixar. Per a obtindre aquesta resposta, s'ha de crear una petició amb un punt d'origen, un punt de destí i els diversos *waypoints* requerits (açò és, parades intermèdies. En aquest cas, els paquets). Aquests punts s'han de descodificar amb una utilitat anomenada "*PolyUtil*". Els punts, ja descodificats, es presenten en format [latitud, longitud] (*LatLng*). Aquests s'inclouen en una llista de tipus *LatLng* i, en passar aquesta llista per la funció composable "*Polyline*", aquesta ruta és dibuixada al mapa.

Clar, aquesta solució només dibuixaria la ruta cada vegada que es carrega el propi mapa, però Pack4You necessita que la ruta es dibuixe cada vegada que

aquesta canvia. Per a aconseguir-ho, s'ha fet servir un MutableLiveData [36] que conté la llista de de tipus *LatLng*, i un objecte *callback*. El procés, que es pot entendre millor observant la figura 5.7 junt al diagrama de la figura 5.8, és el següent:

1. En crear-se la vista, aquesta observa el mutableLiveData que conté la llista de de tipus *LatLng* (present al viewModel).

2. Quan la ruta canvia (ja siga perquè el repartidor ha borrat o afegit un paquet, o perquè s'ha utilitzat un altre algorisme per ordenar-la), el sistema crida a la funció *computeDirectionsAPIResponse*, passant-li com a paràmetre aquesta nova ruta i un objecte *callback*. Aquest *callback* només té una funció, anomenada "*onSuccessResponseDirectionsAPI*", que accepta com a argument la resposta de Directions API.

3. Aquesta funció envia una petició a Directions API amb el punt d'origen, destí i *waypoints* de la nova ruta.

4. Aquesta petició es processa i, en rebre el resultat, es crida a la funció *onSuccessResponseDirectionsAPI* del *callback* que s'ha passat com a argument.

5. Com que aquest objecte s'ha inicialitzat al *ViewModel*, és ara quan s'executa la funció *onSuccessResponseDirectionsAPI*, la qual s'encarrega de descodificar els punts necessaris per dibuixar la ruta presents en la resposta de Directions API i canviar el valor del *mutableLiveData* que conté els punts en format *LatLng*.

6. Com que la vista estava observant aquest valor, quan aquest canvia, també és ella la que es recomposa (es torna a dibuixar) i, per tant, dibuixa la nova ruta.

```
class AtlasViewModelImpl @Inject constructor(
    private val repository: IPackagesAndAtlasRepository
) : ViewModel(), IAtlasViewModel {

    var pointsList = MutableLiveData<List<LatLng>>()

    private val callbackObject = object : ICallbackDirectionsResponse {
        override fun onSuccessResponseDirectionsAPI(response: DirectionsResponse) {
            val encodedPoints = response.routes[0].overview_polyline.points
            var travelTime = 0

            pointsList.postValue(PolyUtil.decode(encodedPoints).toList())
            response.routes[0].legs.forEach { it: Leg
                travelTime += it.duration.value
            }

            if(CurrentSession.algorithm != Algorithm.URGENCY) {
                CurrentSession.travelTime.value = travelTime
            }
        }
    }
```

Figura 5.7: computeDirectionsAPIResponse definida al *callbackObject*

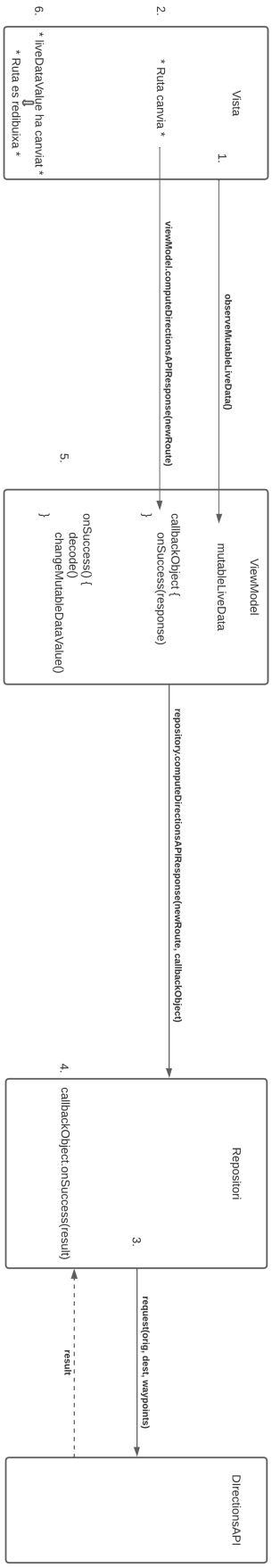

Figura 5.8: Diagrama que mostra el funcionament del canvi dinàmic de la ruta als mapes

5.5 Algorismes implementats

Altra característica molt important per a un repartidor és l'ordenació automàtica dels paquets. Per a açò, a Pack4You s'han implementat diversos algorismes que s'ajustaran a les necessitats que el repartidor requerisca en cada moment. En aquesta secció es descriurà la implementació de cadascun dels algorismes presents en aquest projecte. Cal mencionar que en les notacions matemàtiques que s'incloguen a continuació, n sempre farà referència al nombre de paquets presents a la ruta.

Abans de començar, és necessari explicar què és el que permet als algorismes *Brute Force* i *Nearest Neighbour* funcionar.

Per a executar l'algorisme *Brute Force*, s'han de calcular totes les permutacions possibles. No es considera com a part de l'algorisme perquè aquestes permutacions també podrien utilitzar-se per a executar altres versions de l'algorisme (com, per exemple, una basada en la distància en lloc de en el temps de viatge). En total hi haurà $n!$ permutacions diferents.

Per utilitzar els algoritmes *Brute Force* i *Nearest Neighbour*, abans de començar a executar-los, cal obtenir tres llistes diferents, les quals seran guardades en memòria. Les dades que contenen aquestes llistes s'obtenen de Distance Matrix API [37], interfície que permet obtenir el temps de viatge entre dos punts donats. Les llistes necessàries són:

- Una que descriga el temps de viatge entre tots i cadascun dels paquets. Açò són un total de $(n-1)$ crides. Cal calcular tant el temps d'A a B com el de B a A, doncs pot ser que, per exemple, el camí d'A a B siga de sentit únic i es tarde un sol minut en recórrer-lo però, per a anar de B a A, calga anar per altres carrers i es tarde 3 minuts. També cal tenir en compte que no es fa cap crida per saber el temps de viatge entre un paquet i ell mateix.
- Una que descriga el temps de viatge entre la posició d'origen i cadascun dels paquets (un total de n crides).
- Una que descriga el temps de viatge entre cadascun dels paquets i la localització final (un total de n crides).

Així doncs, en total caldran $(n-1)^2 + 2n$ crides a Distance Matrix API. És clar que poden ser moltes crides, pel que en quantitats grans de paquets aquestes dades tardaran molt de temps en obtindre's. Encara així, si es planifica bé la ruta i no cal afegir cap paquet enmig d'aquesta, aquest càlcul només serà necessari una sola vegada al principi del dia, pel que si s'executa amb el suficient temps d'antelació, el problema es pot mitigar.

A més a més, com que aquestes llistes estan guardades en memòria, el repartidor no depèn d'una connexió constant a Internet per obtenir una ruta optimitzada, doncs ja compta amb la informació necessària per poder executar de nou els algorismes *Brute Force* i *Nearest Neighbour* (tot açò, clar, mentre no s'afegisca altre paquet. En aquest cas, les llistes han de recalcular-se). Cal mencionar que, de no disposar d'aquesta connexió a Internet, només s'ordenaran els paquets en

la llista de paquets, doncs per dibuixar la ruta cal fer ús dels serveis de Directions API, com ja s'ha explicat al punt 5.4.

Una vegada explicats aquests punts, es procedirà a descriure cadascun dels algorismes presents a Pack4You, junt a la seua implementació i complexitat algorísmica:

5.5.1. Directions API

Aquest és l'algorisme utilitzat per Google. Quan es crea una petició a Directions API, es té la possibilitat d'afegir un argument anomenat "*optimize*". Si s'afegeix com a *true*, Directions API ordenarà els *waypoints* afegits a la petició de la millor manera considerada. El nombre màxim de parades per petició és de 25.

Com que és un algorisme que no està implementat manualment ni és públic, no pot saber-se la seua complexitat.

5.5.2. Brute Force

Aquest algorisme té una premissa molt simple: provar totes les possibles rutes i retornar la que menor temps es tarde en recórrer.

Com que ha de provar totes les combinacions disponibles (per exemple, en una ruta on s'ha de passar pels paquets A, B i C, aquest algorisme provarà les rutes A-B-C, A-C-B, B-A-C, B-C-A, C-A-B i C-B-A), la complexitat d'aquest algorisme és $O(n!)$. És per açò que aquest algorisme només es recomana utilitzar quan tenim un nombre xicotet de paquets. A Pack4You aquest nombre ha sigut limitat a 10 paquets.

5.5.3. Nearest Neighbour

Aquest algorisme consisteix a dirigir-se al paquet més proper, sense tenir en compte res més.

L'algorisme, primerament, buscarà entre tots els paquets quin és el més proper al punt d'origen. Aquest serà marcat a una array anomenada *visitedArray*, la qual dóna informació sobre quins paquets estan ja presents en la ruta optimitzada. Aquests paquets no seran comptats per a les futures iteracions de l'algorisme. Seguidament, recorrerà tots els paquets buscant quin és el més proper al paquet actual i, en trobar-lo, també serà marcat en *visitedArray*. Açò ho farà per a cada paquet. Finalment, s'afegirà el temps des de l'últim paquet de la ruta a la localització final.

La complexitat d'aquest algorisme és $O(n^2 + n)$.

5.5.4. Urgency

Aquest algorisme ordena, com el seu nom indica, els paquets per urgència.

Com que volem que l'últim paquet dels paquets amb entrega urgent siga el més proper al primer dels paquets amb entrega ràpida, i l'últim d'aquests el més proper al primer dels paquets amb entrega estàndar, en l'execució d'aquest algorisme també s'utilitzen *callbackObjects*.

Es creen tres rutes diferents: una per a cada tipus de paquet (basat, clar, en la seua urgència). Es comença per la ruta amb els paquets amb entrega estàndar. Fem una crida a DirectionsAPI (que és asíncrona) per a que ordene els paquets d'aquesta ruta i, en obtenir la resposta, cridem a la funció corresponent del *callbackObject* per a que realitze altra crida a DirectionsAPI passant-li, aquesta vegada, la ruta amb els paquets amb entrega ràpida. La diferència és que l'última localització d'aquesta ruta serà la localització del primer paquet de la ruta optimitzada amb entrega estàndar. Una vegada acabada la computació i rebuda la resposta, fem el mateix: cridem a la funció corresponent del *callbackObject* per a que cride a DirectionsAPI passant-li com a paràmetre la ruta amb els paquets amb entrega urgent. La localització final d'aquesta ruta serà la localització del primer paquet de la ruta optimitzada amb entrega ràpida. Una vegada obtesa la resposta d'aquesta última crida, només cal cridar de nou a la funció corresponent del *callbackObject*, on es juntaran les rutes i es mostrarà la ruta conjunta a l'usuari.

Com que s'utilitza DirectionsAPI per ordenar cadascuna de les rutes individuals, la complexitat d'aquest algorisme tampoc pot saber-se.

CAPÍTOL 6
Testing

El testing és un apartat important en qualsevol procés de desevolupament de software. És gràcies a aquestes proves que es pot comprovar que el funcionament del producte és l'esperat i, de no ser-ho, corregir-ho abans que siga molt més complicat fer-ho.

A Pack4You s'han executat diverses proves durant tot el procés de desenvolupament, encara que és en l'última fase d'aquest on les proves han pres més protagonisme.

Cal mencionar que l'automatització d'aquestes proves en un projecte a major escala és molt important, doncs redueix molt el temps que cal per passar-les i, a més, elimina el factor humà, fent que no hi haja possibilitat d'un error d'aquesta naturalesa. A Pack4You, aquesta automatització no ha estat possible pel gran esforç que aquesta requereix; esforç que no era compatible amb els recursos disponibles per a aquest projecte.

6.1 Proves unitàries

Les proves unitàries són aquelles que et permeten provar una secció de codi menuda, com per exemple una funció.

Tots els mètodes de cada classe de Pack4You han sigut provats en el moment d'escriure's, utilitzant diversos valors d'entrada i comprovant que els valors d'eixida eren els adequats.

6.2 Proves d'integració

Una vegada passades les proves unitàries, les proves d'integració són les que segueixen. S'encarreguen de provar que tots els elements provats en les proves unitàries funcionen de la manera esperada en conjunt.

En el cas de Pack4You, aquestes proves es realitzaven cada vegada que un requeriment era implementat. S'ha obtès un resultat satisfactori en totes aquestes proves.

6.3 Proves de regressió

Aquestes són les proves que s'executen als casos d'ús que ja van ser provats a les proves d'integració i van obtindre un resultat satisfactori. El seu objectiu és comprovar que aquest resultat segueix siguent l'esperat.

A Pack4You, les proves de regressió s'han executat en aquesta fase final del projecte, obtenint-se un resultat satisfactori en totes elles.

6.4 Core app quality test

Aquests tests [38] són una manera d'evaluar si l'aplicació a provar compta amb un conjunt de criteris de qualitat definits pels propis desenvolupadors d'Android de Google. La millor manera de comprovar si aquests criteris es compleixen és fer un recorregut per l'aplicació, passant pels principals casos d'ús.

Els criteris de qualitat [1] que Pack4You compleix són els següents:

- **Experiència visual:** L'aplicació compleix amb els requeriments amb ID *VX-N1, VX-N2, VX-N3, VX-V1, VX-V2, VX-A1, VX-A2* i *VX-A3*.
- **Rendiment i estabilitat:** Pack4You compleix amb els requeriments amb ID *PS-S1, PS-P1, PS-P2, PS-T1, PS-T2, PS-T3* i *PS-B1*.
- **Privacitat i seguretat:** L'aplicació compleix amb els següents requisits: *SC-P1, SC-DF1, SC-DF2, SC-DF3* i *SC-E1*.

Els apartats no mencionats no són aplicables a Pack4You.

6.5 Proves amb els algorismes

Per provar si els algorismes implementats per mi funcionen correctament, s'han comprovat manualment els resultats obtinguts després de posicionar paquets en certes localitzacions on pot observar-se si els algorismes mostren els resultats esperats. Cal mencionar que aquest és només un exemple senzill que facilita la comprensió del treball realitzat, però s'han realitzat proves més exhaustives en diversos tipus de rutes (diversos paquets en la mateixa ciutat, repartits per diferents pobles, etc).

Començarem mostrant el mapa sense optimitzar, junt a la llista de paquets corresponent (figura 6.1).

Com es pot observar, és una ruta molt ineficient, on primer es va a La Font de la Figuera (localització final), després es torna a Catarroja, etc.

[1] Referència dels codis disponible a https://developer.android.com/docs/quality-guidelines/core-app-quality

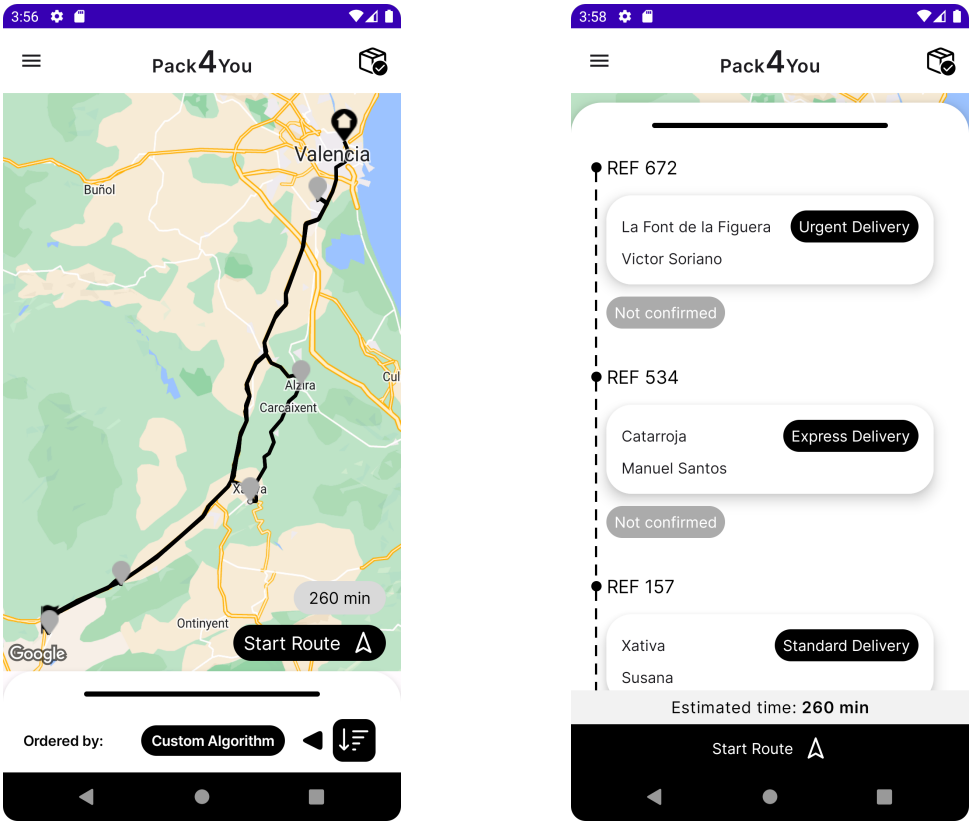

Figura 6.1: Ruta sense optimitzar

6.5.1. Brute Force

En aquest cas utilitzarem l'algorisme *Brute Force* per fer aquesta mateixa ruta en molt menys de temps. El resultat es mostra a la figura 6.2.

Com es pot observar, ara tenim una ruta amb molt més sentit. Per exemple, La Font de la Figuera és l'últim paquet que es reparteix, ja que aquest paquet està al mateix poble que la localització final. El primer és Catarroja, el més proper al punt d'origen.

6.5.2. Nearest Neighbour

Per a la comprovació efectiva d'aquest algorisme, s'ha afegit una nova parada en Castelló de la Plana. Segons el funcionament d'aquest algorisme, el primer paquet en repartir-se hauria de ser el de Catarroja. Després Alzira, Xàtiva, Moixent i la Font de la Figuera. Finalment, hauria d'anar a Castelló de la Plana (doncs en cap moment ha sigut el paquet més proper) i, per a acabar, tornar a La Font de la Figuera, la localització final.

A la figura 6.3 es mostra el resultat de l'execució d'aquest algorisme.

Com podem observar, l'algorisme funciona correctament, doncs segueix l'ordre descrit anteriorment.

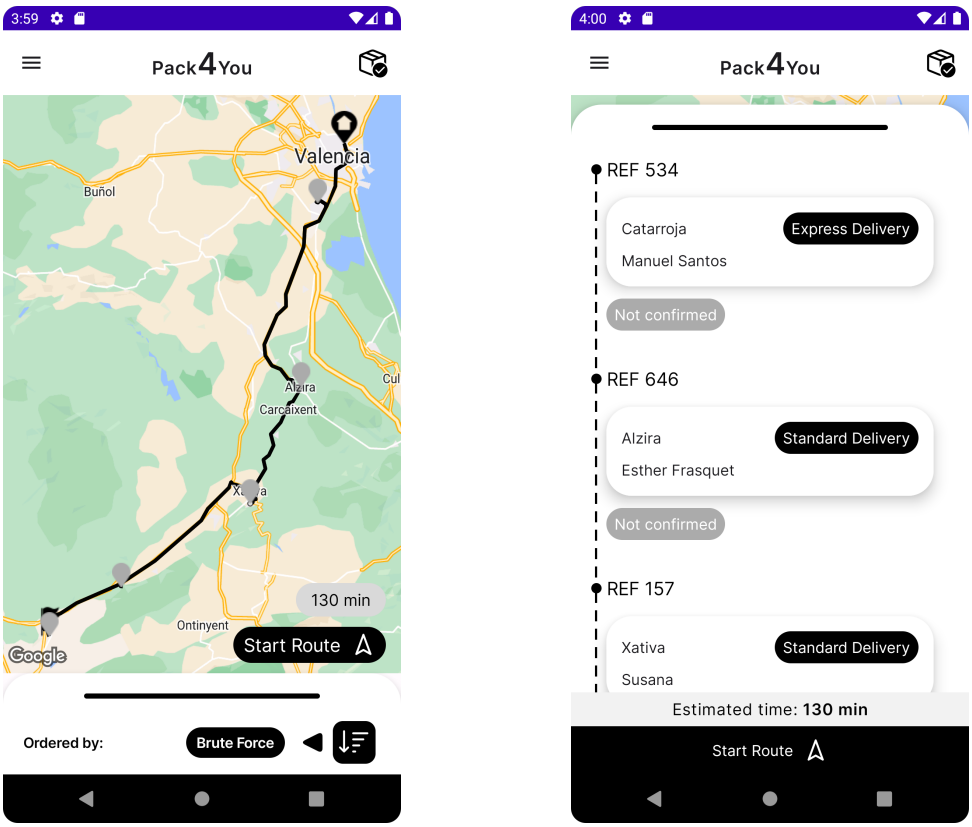

Figura 6.2: Ruta optimitzada per *Brute Force*

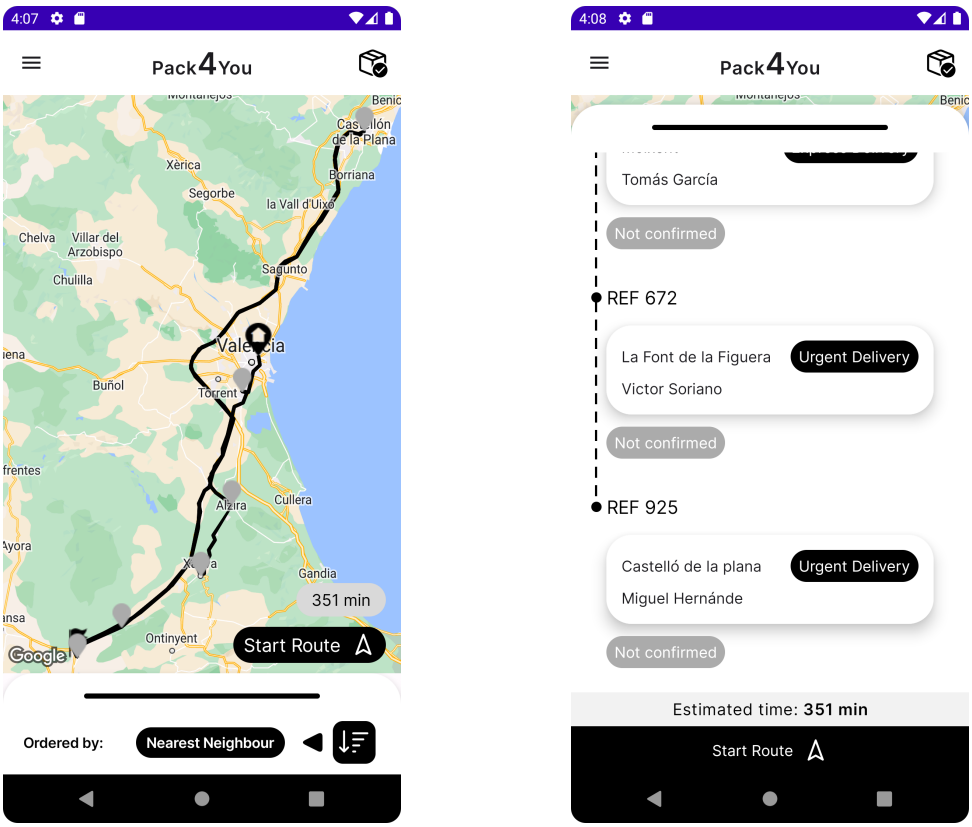

Figura 6.3: Ruta optimitzada per *Nearest Neighbour*

6.5.3. Urgència

Finalment, per comprovar si l'ordenació per urgència funciona correctament, només cal observar si els primers paquets que apareixen a la llista són els que tenen entrega urgent, els segons són els que la tenen ràpida i els últims són els que compten amb entrega estàndar.

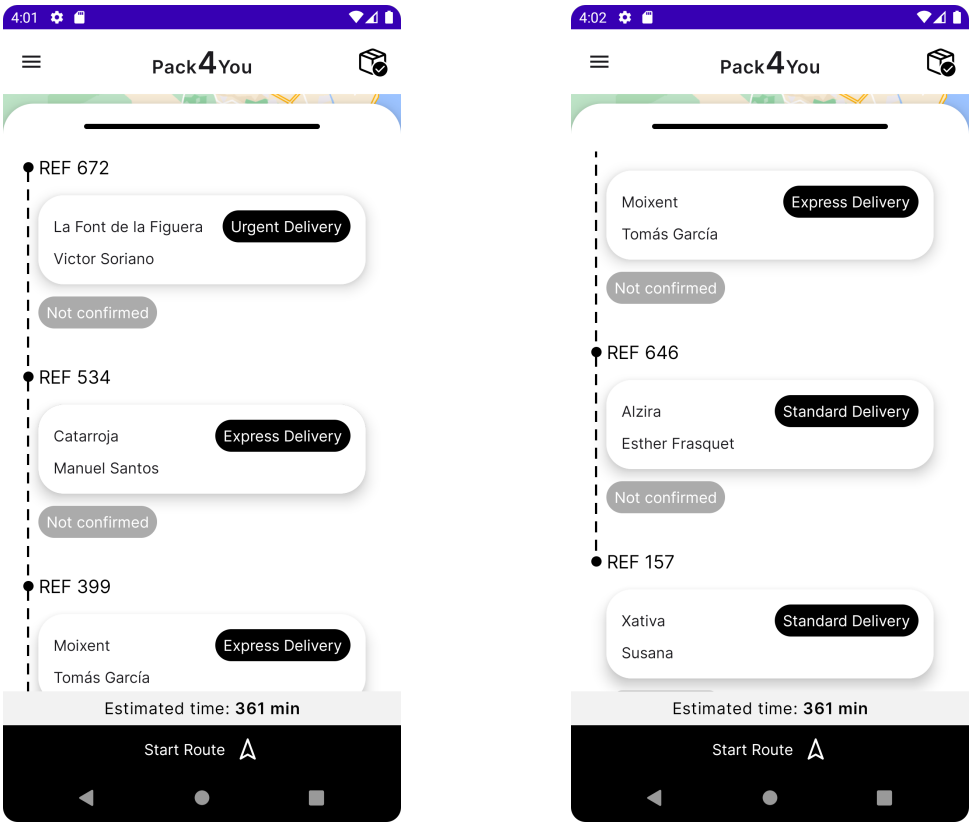

Figura 6.4: Ruta optimitzada per urgència

Efectivament, com podem comprovar a la figura 6.4, l'ordre de paquets és l'esperat.

Gràcies a l'execució d'aquestes proves durant i al final del desenvolupament de Pack4You ha sigut possible detectar diversos errors i corregir-los al codi, fent que l'usuari tinga una millor experiència d'ús i millorant la fiabilitat del sistema. Tots els tests descrits en aquest capítols han sigut passats satisfactòriament.

CAPÍTOL 7
Resultats

En aquest capítol es mostraran els resultats finals aconseguits en el projecte. Per a açò, farem un recorregut per l'aplicació final, exposant els principals casos d'ús del producte.

Quan el repartidor obri l'aplicació, el primer que veu és la pantalla d'inici de sessió (figura 7.1).

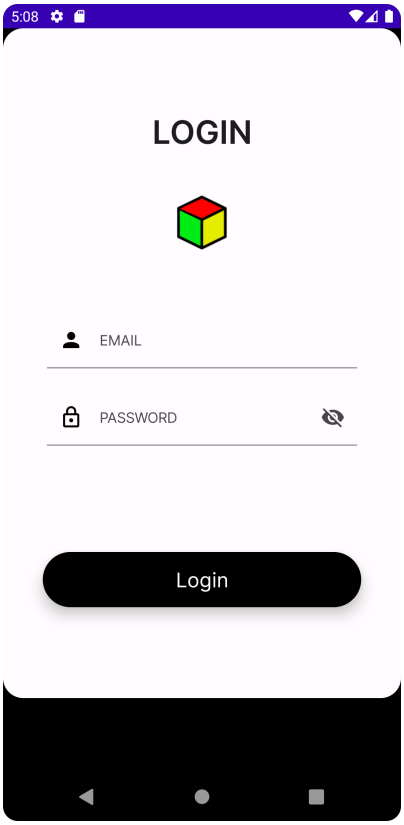

Figura 7.1: Versió final de la pantalla d'Inici de sessió

Una vegada ha emplenat les credencials correctament, el sistema porta al repartidor a la pantalla principal (figura 7.2). Ací, el repartidor pot observar on estan situats els paquets que ha de repartir eixe dia, així com la ruta que s'ha de seguir. Si polsa damunt d'algun dels marcadors presents al mapa, podrà veure informació sobre el paquet corresponent.

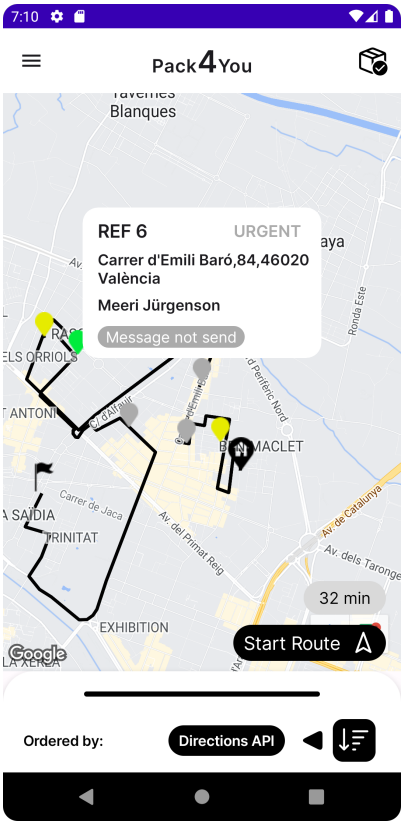

Figura 7.2: Versió final de la pantalla principal

Com que el repartidor vol veure informació més detallada dels paquets i l'ordre d'aquests en una llista, arrossegarà cap amunt el menú inferior i trobarà la llista de paquets a repartir.

L'usuari s'adona que els paquets no estan ordenats, pel que polsa el botó d'ordenació i apareixen els diferents algorismes amb els que Pack4You compta. En aquest cas (figura 7.3) el repartidor ha decidit ordenar el paquets mitjançant l'algorisme Directions API. En polsar sobre aquest algorisme, una alerta apareixerà indicant que s'està executant aquest algorisme i, en acabar, la llista de paquets s'haurà reordenat.

Després, el repartidor veu que hi ha un paquet al furgó que no està ben empaquetat, pel que no pot repartir-lo. Com que vol posposar-lo, arrossegarà la targeta corresponent cap a l'esquerra i una alerta apareixerà preguntant-li si vol eliminar-lo o simplement posposar-lo. El repartidor elegirà posposar, i un missatge de confirmació apareixerà per pantalla.

Abans de començar la ruta, el repartidor s'adona que un paquet no està registrat. Per tant, el que farà serà obrir el menú d'opcions (figura 7.4) i polsarà sobre el menú que li permet afegir un nou paquet. En aquesta pantalla (figura 7.5) podrà definir la direcció de destí, el nom del client i l'urgència del paquet. El sistema només deixarà al repartidor afegir el nou paquet si la direcció seleccionada és vàlida i el nom del client no és buit.

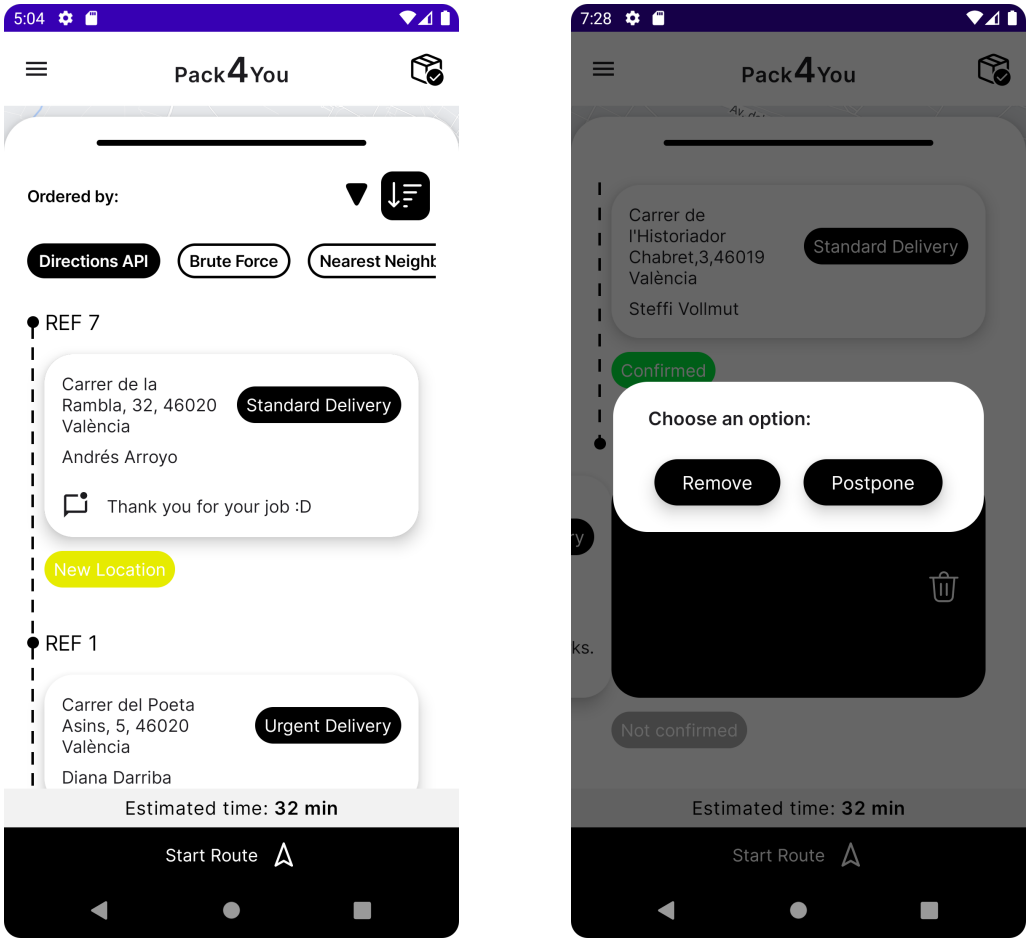

Figura 7.3: Versió final de la pantalla on es mostra la llista de paquets

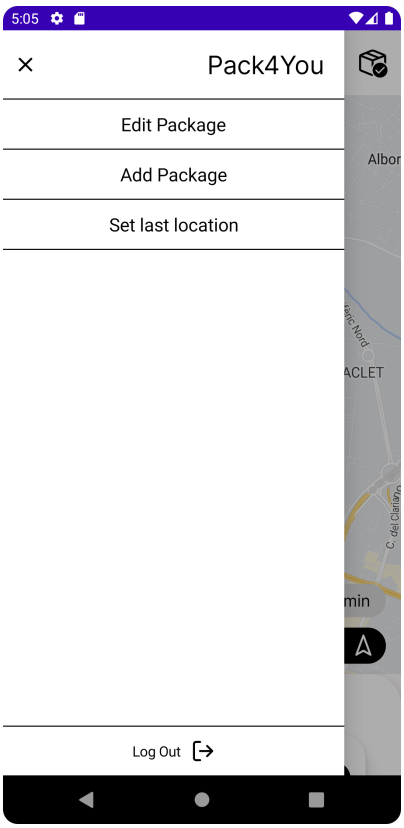

Figura 7.4: Versió final del menú lateral de selecció

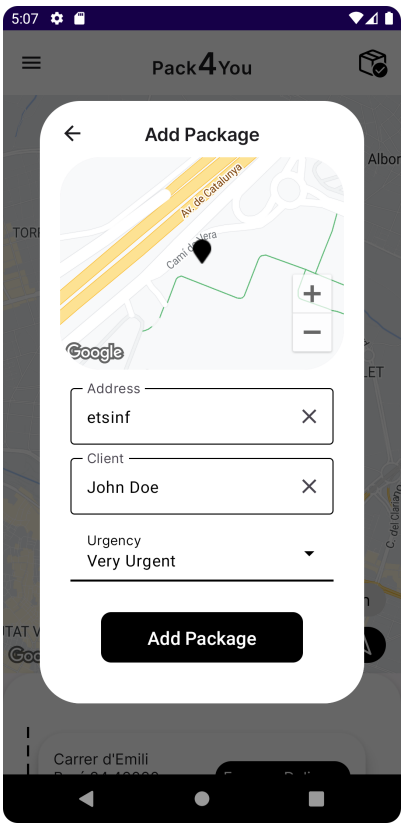

Figura 7.5: Versió final de la pantalla d'afegir paquet

Una vegada afegit el paquet, el repartidor veu que no ha assignat l'urgència adequada al paquet que acaba d'afegir. Per tant, torna a obrir el menú d'opcions i polsa en editar paquet. Apareix una finestra on ha d'elegir el paquet a editar (figura 7.6). Una vegada seleccionat, una finestra similar a la d'afegir paquet es mostrarà, amb les dades d'aquest paquet seleccionat ja emplenades. Una vegada canviada l'urgència del paquet, el repartidor confirma el canvi i aquest es fa efectiu.

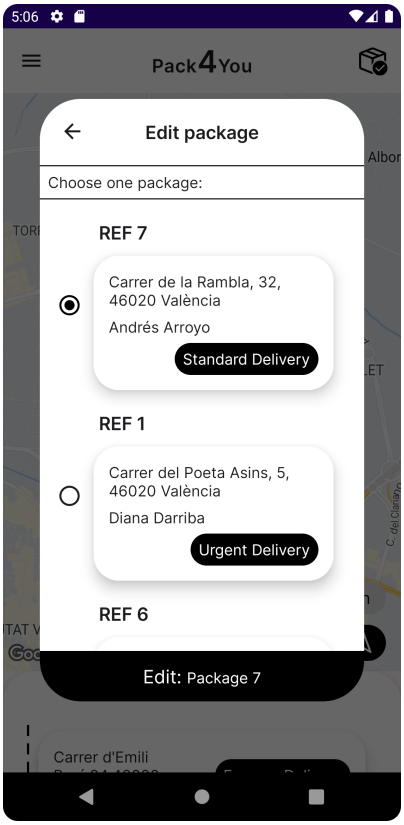

Figura 7.6: Versió final de la pantalla de seleccionar un paquet per a editar

Després, el repartidor recorda que el gerent li ha dit que avui podria acabar a casa, doncs no necessiten el furgó fins el pròxim dia. Així doncs, el repartidor torna a obrir el menú i aquesta vegada elegeix l'opció que li permet canviar l'última localització. En fer-ho, apareix una finestra (figura 7.7) on pot elegir entre les seues últimes destinacions o, de ser necessari, afegir-ne una nova. En aquest últim cas, el sistema només l'afegiria si és una adreça vàlida. En elegir l'última localització desitjada, el repartidor confirmarà la selecció i la ruta canviarà tenint aquesta última localització en compte.

Una vegada configurada la ruta al seu gust, el repartidor la iniciarà (figura 7.8), accedint a una vista prèvia de la realització de la ruta a seguir. Ací podrà marcar com a entregat els paquets que ja estiguen al seu destí.

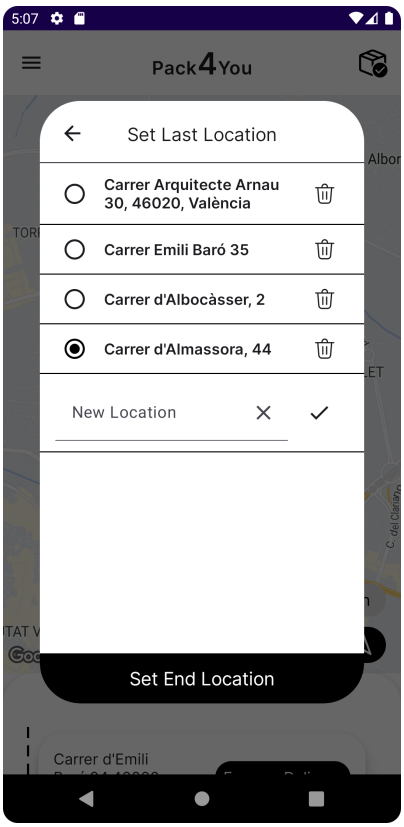

Figura 7.7: Versió final de la pantalla d'elegir última localització

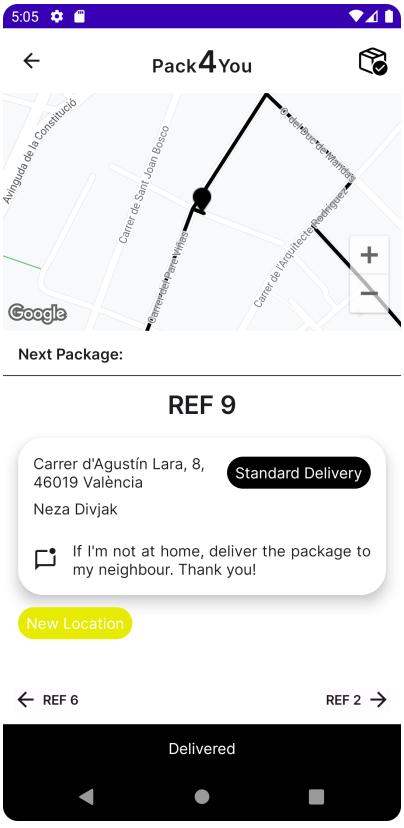

Figura 7.8: Versió final de la pantalla de començar ruta

Finalment, enmig de la ruta el repartidor vol comprovar si el paquet amb número de referència 9 ha sigut entregat. Per a açò, polsarà sobre el botó de paquets entregats i apareixerà una finestra amb tots els paquets que el repartidor ha marcat com a entregats.

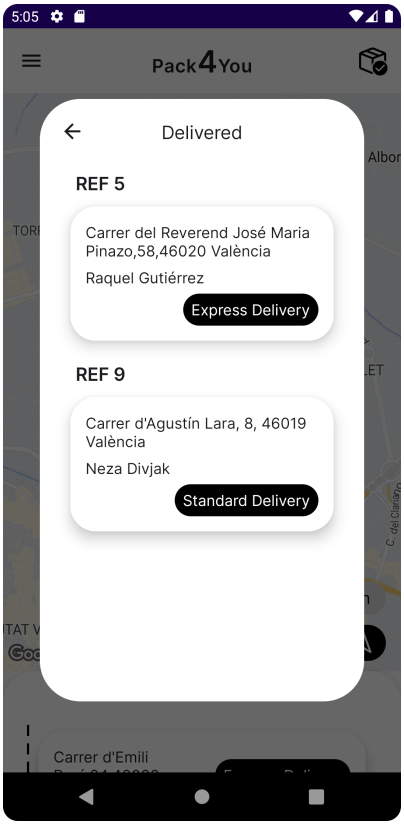

Figura 7.9: Versió final de la pantalla de paquets entregats

CAPÍTOL 8
Conclusions

Per poder donar com a finalitzat aquest Treball de Fi de Grau, es comprovarà si els objectius definits a l'apartat 1.2 han sigut aconseguits. També es discutirà quin efecte ha tingut el desenvolupament de Pack4You sobre la meua formació com a Enginyer de Software i creixement personal. Seguidament, es comentaran quins han sigut els aspectes més difícils d'implementar i com s'han solucionat, així com les limitacions de la versió actual de Pack4You. Finalment, es reflexionarà sobre la relació d'aquest projecte amb els estudis cursats.

8.1 Ha tingut èxit el projecte?

El primer objectiu definit al principi d'aquest projecte era facilitar la vida dels repartidors mitjançant una aplicació amb mecanismes de gestió i ordenació automàtica de paquets. Aquest objectiu s'ha aconseguit, doncs el resultat final d'aquest projecte és un producte amb una Interfície d'Usuari satisfactòria i mecanismes de gestió i ordenació automàtica de paquets implementats. Aquest sistema evitarà que el repartidor ordene manualment els paquets i li estalviarà molt de temps en el seu dia a dia. A més a més, a diferència de les altres opcions disponibles al mercat, Pack4You permet la utilització de diferents algorismes i l'ús de *Brute Force* i *Nearest neighbour* de manera offline, així com definir l'última destinació del repartidor o consultar els paquets ja entregats.

El segon objectiu era realitzar un projecte de principi a fi, passant per totes les fases de desenvolupament que aquest comporta. Aquest objectiu també s'ha complit doncs, com s'ha observat al llarg de la memòria, durant aquests mesos s'ha realitzat un treball que ha comportat un anàlisi de requisits, un disseny del sistema a desenvolupar, una posterior implementació d'aquest i, finalment, una verificació del mateix. A més, s'ha implementat de manera satisfactòria l'arquitectura MVVM, característica en la que s'havia posat especial interès.

El següent objectiu era aprendre sobre la integració de característiques de geolocalització i un sistema de mapes a una aplicació Android. Com s'ha comprovat en l'aplicació final, aquest objectiu s'ha aconseguit, doncs Pack4You compta amb aquests requeriments, i els mateixos funcionen d'una manera satisfactòria.

L'últim objectiu era investigar i implementar diferents algorismes aplicables a l'àmbit del projecte. Aquest objectiu també s'ha complit, ja que el producte

final compta amb diversos algorismes implementats manualment amb els que el repartidor pot confiar per fer la seua feina més eficientment.

Com que tots els objectius han estat aconseguits, podem afirmar que el projecte ha sigut un èxit. Però no només per complir aquests objectius, sinó per tota l'experiència, tant professional com personal, que el desenvolupament complet de Pack4You m'ha donat:

- A nivel professional, aquest projecte m'ha obligat a aprendre i investigar sobre diverses tecnologies. Entre els nous coneixements dels que dispose, es troba la Injecció de Dependències utilitzant *Hilt*, el principi d'Inversió de Dependències, la implementació d'un sistema de mapejat en una aplicació Android o l'ús de les interfícies Directions API i Distance Matrix API. Així doncs, m'he convertit en una millor opció per al mercat laboral i en un enginyer més qualificat.
- A nivel personal, Pack4You ha sigut el projecte al que més hores he invertit mai. En ser un projecte propi, he estat molt implicat i m'ha fet reafirmar que el desenvolupament de software és allò al que vull dedicar-me.

8.2 Dificultats encontrades i limitacions del projecte

Per suposat, el desenvolupament d'aquest projecte ha comportat dificultats i, per superar-les, s'ha hagut d'investigar i aprendre sobre diverses tecnologies i tècniques.

La primera dificultat amb la que em vaig trobar va ser la correcta implementació de l'estructura MVVM. Abans de començar aquest projecte, ja tenia un noció del funcionament d'aquesta, però ràpidament em vaig adonar que era més complexa de l'esperat. A més, per aconseguir un projecte escalable i més robust, també vaig haver d'implementar l'inversió de dependències i l'injecció de dependències mitjançant Hilt. Després d'investigar i aprendre d'altres projectes i, sobretot, amb l'ajuda de David de Andrés, el meu tutor, vaig poder entendre completament el funcionament d'aquesta arquitectura i l'aplicació d'aquestes tècniques. Com a conseqüència, s'ha obtès un producte preparat per poder afegir-li noves característiques sense un gran esforç, fàcilment verificable i molt desacoblat.

Altra dificultat a la que vaig enfrontar-me va ser el tractament de les crides asíncrones. Al principi sabia quin era el problema, doncs el que passava era que feia una crida però no entenia com podia esperar el resultat i seguir amb l'execució. De nou, investigant i amb l'ajuda de David, vaig poder utilitzar el patró observador i els *callback* per poder implementar correctament les característiques que necessitaven d'aquestes crides. Gràcies a superar aquesta dificultat, Pack4You compta amb una de les seues característiques fonamentals: l'ordenació automàtica de paquets. Depenent de l'algorisme utilitzat, l'execució d'aquest implicarà bé cridar a Directions API (algorisme Directions API i ordenació per urgència) o bé a Distance Matrix API (*Brute Force* i *Nearest Neighbour*).

Sobre les limitacions d'aquesta versió de Pack4You, de moment encara no és comercialitzable. Encara que ja puga ser útil per als repartidors, aquesta cal que

s'integre amb els sistemes de les empreses de missatgeria (per exportar la informació dels paquets als models de Pack4You). A més, per a que el valor aportat siga major, caldria desenvolupar la part d'administrador (per poder gestionar els diferents repartidors i assignar-los els paquets corresponents) i la part client (per a que el repartidor no haja de preocupar-se sobre la situació del client enfront l'entrega. El primer no haurà de cridar per saber què fer si el segon no està a casa, ja que aquest ja haurà donat tota la informació des de la seua part).

8.3 Reflexió sobre la relació amb els estudis cursats

Finalment, com a reflexió sobre la relació d'aquest treball amb els estudis cursats, puc afirmar que aquests m'han donat totes les ferramentes necessàries per poder aconseguir acabar satisfactòriament aquest projecte. Gràcies al grau en Enginyeria Informàtica i l'especialització en Enginyeria del Software, puc afirmar que sóc capaç de desenvolupar productes de qualitat de principi a fi, així com resoldre els problemes que puguen sorgir durant el desenvolupament d'un sistema, aportant solucions efectives basades en l'anàlisi del propi problema i en l'experiència i coneixement adquirits durant aquests anys.

CAPÍTOL 9
Treball futur

En aquest capítol es llistaran les característiques i perfeccionaments que no ha sigut possible implementar amb els recursos assignats a aquest projecte. Alguns punts que es tractaran ja han sigut mencionats al llarg del projecte.

- **Vista client:** Una de les característiques que més valor aportarien a Pack4You és la possibilitat d'interactuar amb el client. A l'inici del projecte, aquesta característica va ser contemplada i, com pot observar-se al diagrama de casos d'ús i de classes, el disseny de Pack4You està preparat per suportar-la. Aquesta característica seria la pròxima en invertir els recursos.
- **Vista administrador:** Per suposat, la vista administrador també seria una característica molt important. Aquest seria l'encarregat, entre altres responsabilitats, de registrar nous usuaris, assignar els paquets als diferents repartidors i poder comunicar-se amb ells de ser necessari.
- **Comentaris del tutor:** Una vegada finalitzat el desenvolupament de Pack4You, el meu tutor va fer diversos comentaris sobre la versió final. Entre els més destacats, tenim la possibilitat de borrar/posposar un paquet des de la pantalla de ruta iniciada, mantenir la sessió iniciada quan es tanca l'aplicació i afegir un botó a la pantalla de ruta iniciada per a poder tornar a la posició original.
- **Traducció:** Per suposat, si es vol que Pack4You siga accessible per al major nombre de persones possible, l'aplicació hauria de suportar més d'un idioma, com ara el català o l'espanyol.
- **Inclusió d'animacions:** Avui en dia, tota aplicació que vulga tenir una Interfície d'Usuari moderna ha de comptar amb animacions. Aquestes proporcionen informació a l'usuari d'una manera clara i estètica. Actualment, Pack4You només compta amb l'animació de càrrega als algorismes.
- **Ordenació manual de paquets:** Encara que l'ordenació automàtica de paquets siga la majoria de vegades la millor opció, pot ser el repartidor vulga ajustar aquesta ruta optimitzada per alguna raó.
- **Automatització dels tests:** Per suposat, és important automatitzar la verificació de l'aplicació. Com ja s'ha explicat, per falta de recursos no ha sigut possible fer-ho fins ara, però qualsevol aplicació que comporte canvis i la

introducció de noves característiques ha de comptar amb proves automatitzades. Amb elles, el temps utilitzat en verificar el sistema es redueix enormement i la fiabilitat augmenta.

Bibliografia

[1] J. Salvatierra. La paquetería salta a otra dimensión. *El País*, 26 de setembre de 2016, [Online] Consultat a https://elpais.com/economia/2016/09/23/actualidad/1474643851_381534.html [Data de consulta] 10 d'agost de 2022

[2] Los ingresos del sector postal aumentaron en 2020. Nota de prensa del CNMC datada el 29 de desembre del 2021 Consultat a https://www.cnmc.es/prensa/informe-sector-postal-2020-20211229 [Data de consulta] 10 d'agost de 2022

[3] Pàgina web Google Maps. Disponible a https://www.google.com/maps [Data de consulta] 12 d'agost de 2022

[4] Pàgina web oficial de ZippyKind. Disponible a https://zippykind.com/ [Data de consulta] 12 d'agost de 2022

[5] Pàgina a la PlayStore de RoadWarrior. [Data de creació de la pàgina] 4 d'agost de 2013. Disponible a https://play.google.com/store/apps/details?id=com.roadwarrior.android [Data de consulta] 12 d'agost de 2022.

[6] Pàgina web oficial per a equips de RoadWarrior. Disponible a https://roadwarrior.app/teams [Data de consulta] 12 d'agost de 2022

[7] Pàgina web oficial d'Uber Disponible a https://www.uber.com/es/en/ [Data de consulta] 12 d'agost de 2022

[8] Secció de DirectionsAPI a la pàgina web oficial de desenvolupadors de Google Disponible a https://developers.google.com/maps/documentation/directions/overview [Data de consulta] 14 d'agost de 2022

[9] Equip de Free Code Camp. Brute Force Algorithms Explained *Free Code Camp*, 6 de gener de 2020, [Online] Disponible a https://www.freecodecamp.org/news/brute-force-algorithms-explained/ [Data de consulta] 14 d'agost de 2022

[10] Dhilip Subramanian. A Simple Introduction to K-Nearest Neighbors Algorithm *Towards Data Science*, 8 de juny de 2019, [Online] Disponible a https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e [Data de consulta] 14 d'agost de 2022

[11] MoSCoW Prioritization. Disponible a https://www.productplan.com/glossary/moscow-prioritization/ [Data de consulta] 14 d'agost de 2022

[12] Institute of Electrical and Electronics Engineers (IEEE). IEEE Standard Glossary of Software Engineering Terminology *IEEE*, 31 de desembre de 1990

[13] Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide- *Addison-Wesley*, 20 d'octubre de 1998

[14] Priyank Kumar Understanding MVVM Architecture in Android *Medium*, 11 d'abril de 2020, [Online]. Disponible a https://medium.com/bbc-design-engineering/modernising-a-legacy-android-app-architecture-part-two-mvvm-ish-1b0372678005 [Data de consulta] 16 d'agost de 2022

[15] Leland Richardson Understanding Jetpack Compose. *Medium*, 28 d'agost de 2020, [Online]. Disponible a https://medium.com/bbc-design-engineering/modernising-a-legacy-android-app-architecture-part-two-mvvm-ish-1b0372678005 [Data de consulta] 16 d'agost de 2022

[16] Pàgina web oficial de JSON. Disponible a https://www.json.org/json-en.html [Data de consulta] 16 d'agost de 2022

[17] Pàgina web oficial de Firebase. Disponible a https://firebase.google.com/ [Data de consulta] 16 d'agost de 2022

[18] Pàgina web oficial de desenvolupadors Android, secció d'Arquitectura de la app Disponible a https://developer.android.com/topic/architecture [Data de consulta] 17 d'agost de 2022

[19] Gene Zeiniss. UML Diagrams: The Common Tongue. *Medium*, 21 de desembre de 2020, [Online]. Disponible a https://medium.com/swlh/uml-diagrams-the-common-tongue-f19366e5564 [Data de consulta] 17 d'agost de 2022

[20] Denise Sánchez i Carmen Gerea. Prototipo: qué es y para qué sirve. *Freed*, 15 de març de 2021, [Online]. Disponible a https://freed.tools/blogs/ux-cx/prototipo [Data de consulta] 17 d'agost de 2022

[21] Pàgina web oficial de Figma. Disponible a https://www.figma.com/ [Data de consulta] 17 d'agost de 2022

[22] Pàgina web oficial de Firebase, secció de Firestore. Disponible a https://firebase.google.com/docs/firestore [Data de consulta] 21 d'agost de 2022

[23] Christof Strauch NoSQL Databases *Stuttgart University* Disponible a https://bigb.es/lectures/2014/15.5.pdf [Data de consulta] 21 d'agost de 2022

[24] Pàgina web oficial de desenvolupadors Android, secció d'Android Studio Disponible a https://developer.android.com/studio [Data de consulta] 21 d'agost de 2022

[25] Pàgina web oficial de Kotlin Disponible a https://kotlinlang.org/ [Data de consulta] 21 d'agost de 2022

[26] Pàgina web oficial de JetBrains Disponible a https://www.jetbrains.com/ [Data de consulta] 21 d'agost de 2022

[27] Pàgina web oficial de desenvolupadors Android, secció de Layouts Disponible a https://developer.android.com/guide/topics/ui/declaring-layout [Data de consulta] 21 d'agost de 2022

[28] Pàgina web oficial de Git Disponible a https://git-scm.com/ [Data de consulta] 21 d'agost de 2022

[29] Què és open source? Pàgina web que conté projectes i iniciatives open source Disponible a https://opensource.com/resources/what-open-source [Data de consulta] 21 d'agost de 2022

[30] Robert C. Martin (Uncle Bob) The Clean Architecture *Clean Coder*, 13 d'agost de 2012, [Online] Disponible a https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html [Data de consulta] 23 d'agost de 2022

[31] Álvaro Blázquez ¿Qué son las inyecciones de dependencias? *Medium*, 22 de setembre de 2017, [Online] Disponible a https://medium.com/@alvarob25/qu%C3%A9-son-las-inyecciones-de-dependencias-f4b97a9e644a [Data de consulta] 23 d'agost de 2022

[32] Singleton design pattern on Refactoring Guru website Disponible a https://refactoring.guru/design-patterns/singleton [Data de consulta] 24 d'agost de 2022

[33] Observer design pattern on Refactoring Guru website Disponible a https://refactoring.guru/design-patterns/observer [Data de consulta] 24 d'agost de 2022

[34] Pàgina web oficial de desenvolupadors Android, secció d'injecció de dependències amb Hilt Disponible a https://developer.android.com/training/dependency-injection/hilt-android [Data de consulta] 3 de setembre de 2022

[35] Samuel Oloruntoba SOLID: The First 5 Principles of Object Oriented Design *Digital Ocean*, 30 de novembre de 2021, [Online] Disponible a https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design [Data de consulta] 3 de setembre de 2022

[36] Anupam Chugh. Android LiveData. *Digital Ocean*, 3 d'agost de 2022, [Online] Disponible a https://www.digitalocean.com/community/tutorials/android-livedata [Data de consulta] 3 de setembre de 2022

[37] Pàgina web oficial de desenvolupadors Android, secció Distance Matrix API Disponible a https://developers.google.com/maps/documentation/distance-matrix/overview [Data de consulta] 3 de setembre de 2022

[38] Pàgina web oficial de desenvolupadors Android, secció de Core app quality. Disponible a https://developer.android.com/docs/quality-guidelines/core-app-quality [Data de consulta] 3 de setembre de 2022

[39] Pàgina web oficial dels Objectius de Desenvolupament Sostenibles Disponible a https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ [Data de consulta] 7 de setembre de 2022

APÈNDIX A
Casos d'ús

Taula A.1: CU Iniciar sessió

Nom:	Iniciar sessió
Actor(s):	Repartidor
Descripció: L'usuari iniciarà sessió utilitzant les seues credencials. Una vegada autenticat, passarà a ser un Repartidor autenticat.	
Precondicions: L'usuari compta amb un correu i una contrassenya registrats al sistema.	
Flux: L'usuari emplenarà el seu correu i contrassenya i polsarà el botó d'iniciar sessió.	
Postcondicions: Si el correu i la contrassenya són correctes, l'usuari accedirà a la pantalla d'inici de l'aplicació. Es convertirà en Repartidor autenticat.	

Taula A.2: CU Obrir menú d'opcions

Nom:	CU Obrir menú d'opcions
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà accedir a un menú d'opcions.	
Precondicions: -	
Flux: El repartidor es situarà al menú principal i polsarà sobre la icona corresponent per obrir el menú.	
Postcondicions: El menú d'opcions serà desplegat.	

Taula A.3: CU Afegir paquet

Nom:	Afegir paquet
Actor(s):	Repartidor autenticat
Descripció: El repartidor afegirà un paquet emplenant la direcció d'enviament, el nom del client i l'urgència del paquet.	
Precondicions: -	
Flux: L'usuari obrirà el menú d'opcions i polsarà el botó "Afegir Paquet".	
Postcondicions: Si la direcció d'enviament és vàlida i el nom del client no està buit, el paquet s'afegirà al sistema.	

Taula A.4: CU Editar paquet

Nom:	Editar paquet
Actor(s):	Repartidor autenticat
Descripció: El repartidor editarà la informació d'un paquet elegint un dels existents.	
Precondicions: Hi ha almenys un paquet registrat al sistema.	
Flux: L'usuari obrirà el menú d'opcions i polsarà el botó "Editar Paquet". Després, elegirà el paquet corresponent i polsarà "Editar". En la nova finestra, canviarà la informació desitjada i polsarà "Editar paquet".	
Postcondicions: Si la direcció d'enviament és vàlida i el nom del client no està buit, el paquet s'editarà.	

Taula A.5: CU Accedir al llistat de paquets

Nom:	Accedir al llistat de paquets
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà consultar un llistat amb els paquets a repartir i informació sobre aquests.	
Precondicions: Hi ha almenys un paquet registrat al sistema.	
Flux: L'usuari arrossegarà cap amunt el menú inferior per accedir a la llista de paquets.	
Postcondicions: L'usuari pot observar la llista de paquets presents a la ruta, junt a informació sobre aquests.	

Taula A.6: CU Tancar sessió

Nom:	Tancar sessió
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà tancar la sessió actual .	
Precondicions: -	
Flux: L'usuari obrirà el menú d'opcions i polsarà "Tancar sessió".	
Postcondicions: El Repartidor autenticat passarà a ser Repartidor.	

Taula A.7: CU Ordenar paquets

Nom:	Ordenar paquets
Actor(s):	Repartidor autenticat
Descripció: L'usuari podrà ordenar la llista de paquets segons l'algorisme desitjat	
Precondicions: Hi ha almenys un paquet registrat al sistema.	
Flux: L'usuari polsarà el botó que li permet veure els algorismes i elegirà el desitjat.	
Postcondicions: La llista de paquets s'ordena segons l'algorisme seleccionat. La ruta present al mapa es redibuixa.	

Taula A.8: CU Mostrar mapa

Nom:	Mostrar mapa
Actor(s):	Repartidor autenticat
Descripció: El repartidor observarà un mapa amb, si hi ha, els paquets localitzats amb un punter.	
Precondicions: -	
Flux: El repartidor vorà el mapa just després d'iniciar sessió.	
Postcondicions: El repartidor por consultar el mapa.	

Taula A.9: CU Iniciar ruta

Nom:	Iniciar ruta
Actor(s):	Repartidor autenticat
Descripció: El repartidor iniciarà una vista prèvia de la ruta a recórrer.	
Precondicions: Ha d'existir almenys un paquet en el sistema.	
Flux: El repartidor polsarà sobre el botó "Iniciar ruta" al mapa, o bé accedirà a la llista de paquets i polsarà a un botó similar.	
Postcondicions: El repartidor accedirà a la vista prèvia de la ruta a recórrer.	

Taula A.10: CU Marcar paquet com a entregat

Nom:	Marcar paquet com a entregat
Actor(s):	Repartidor autenticat
Descripció: El repartidor marcarà un paquet com entregat.	
Precondicions: Ha d'existir almenys un paquet en el sistema.	
Flux: El repartidor arrossegarà una targeta de la llista de paquets cap a la dreta, o bé polsarà el botó "Marcar com a entregat"en la pantalla d'Iniciar ruta.	
Postcondicions: El paquet serà marcat com a entregat.	

Taula A.11: CU Consultar informació del paquet al mapa

Nom:	Consultar informació del paquet al mapa
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà accedir a informació d'un paquet polsant el marcador corresponent al mapa.	
Precondicions: Ha d'existir almenys un paquet en el sistema.	
Flux: El repartidor accedirà al mapa i polsarà sobre un marcador.	
Postcondicions: Un pop-up mostrarà informació sobre el paquet corresponent	

Taula A.12: CU Eliminar paquet

Nom:	Eliminar paquet
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà eliminar un paquet present a la ruta a recórrer.	
Precondicions: Ha d'existir almenys un paquet en el sistema.	
Flux: El repartidor accedirà a la llista de paquets i arrossegarà el corresponent cap a l'esquerra. Ací, una alerta es mostrarà preguntant a l'usuari si vol eliminar o posposar el paquet. L'usuari haurà de polsar el botó "Eliminar".	
Postcondicions: El paquet serà borrat del sistema.	

Taula A.13: CU Posposar paquet

Nom:	Posposar paquet
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà posposar un paquet per al següent dia laborable.	
Precondicions: Ha d'existir almenys un paquet en el sistema.	
Flux: El repartidor accedirà a la llista de paquets i arrossegarà el corresponent cap a l'esquerra. Ací, una alerta es mostrarà preguntant a l'usuari si vol eliminar o posposar el paquet. L'usuari haurà de polsar el botó "Posposar".	
Postcondicions: El paquet serà posposat al següent dia laborable.	

Taula A.14: CU Accedir als paquets entregats

Nom:	Accedir als paquets entregats
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà accedir a un llistat amb els paquets marcats com entregats.	
Precondicions: -	
Flux: El repartidor polsarà el botó corresponent en la pantalla en la que es trobe.	
Postcondicions: El repartidor accedirà a un llistat on es troben els paquets ja entregats.	

Taula A.15: CU Canviar última localització

Nom:	CU Canviar última localització
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà canviar l'última localització de la ruta.	
Precondicions: Ha d'existir almenys una última localització registrada al sistema.	
Flux: El repartidor obrirà el menú i polsarà sobre el botó "Definir última localització". S'obrirà una finestra on podrà d'elegir l'última localització desitjada. Després haurà de polsar el botó "Definir última localització".	
Postcondicions: L'última localització de la ruta haurà canviat a la nova elegida per l'usuari.	

Taula A.16: CU Eliminar última localització

Nom:	CU Eliminar última localització
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà eliminar una última localització.	
Precondicions: Ha d'existir almenys una última localització registrada al sistema.	
Flux: El repartidor obrirà el menú i polsarà sobre el botó "Definir última localització". S'obrirà una finestra on podrà consultar les últimes localitzacions registrades. Haurà de polsar la icona corresponent per borrar aquesta localització.	
Postcondicions: L'última localització elegida serà borrada del sistema.	

Taula A.17: CU Afegir última localització

Nom:	CU Afegir última localització
Actor(s):	Repartidor autenticat
Descripció: El repartidor podrà afegir una nova última localització.	
Precondicions: -	
Flux: El repartidor obrirà el menú i polsarà sobre el botó "Definir última localització". S'obrirà una finestra on podrà consultar les últimes localitzacions registrades. Haurà d'escriure la nova adreça on vol definir la nova última localització i, després, polsar en la icona corresponent per validar-la.	
Postcondicions: Si l'adreça escrita és vàlida, la nova localització serà afegida al sistema.	

Taula A.18: CU Enviar missatge a client

Nom:	CU Enviar missatge a client
Actor(s):	Sistema de càlcul de distàncies
Descripció: El sistema de càlcul de distàncies podrà enviar un missatge al client quan el repartidor estigui a prop.	
Precondicions: El repartidor ha d'estar suficientment prop del client, a banda de tenir connexió a Internet.	
Flux: El repartidor s'aproparà el suficient al client i el sistema s'activarà automàticament.	
Postcondicions: El client rebrà un missatge.	

Taula A.19: CU Respondre missatge

Nom:	CU Respondre missatge
Actor(s):	Client
Descripció: El client podrà respondre el missatge enviat pel sistema de càlcul de distàncias, en el que donarà informació sobre la seua situació davant l'entrega.	
Precondicions: El sistema de càlcul de distàncies ha enviat el missatge i el client l'ha rebut.	
Flux: El client rebrà el missatge a Telegram i contestarà mitjançant el seu dispositiu. Finalment, polsarà el botó "Enviar".	
Postcondicions: El sistema s'actualitzarà en consonància a la resposta del client.	

APÈNDIX B
Objectius de Desenvolupmanet Sostenible (ODS)

Taula B.1: Grau de relació del treball amb els Objectius de Desenvolupament Sostenible

Objectius de Desenvolupament Sostenibles	Alt	Mitjà	Baix	No Procedeix
ODS 1. **Fi de la pobresa.**				X
ODS 2. **Fam zero.**				X
ODS 3. **Salut i benestar .**				X
ODS 4. **Educació de qualitat.**				X
ODS 5. **Igualtat de gènere.**				X
ODS 6. **Aigua neta i sanejament.**				X
ODS 7. **Energia assequible i no contaminant.**				X
ODS 8. **Treball decent i creixement econòmic.**	X			
ODS 9. **Industria, innovació i infraestructures.**				X
ODS 10. **Reducció de les desigualtats.**				X
ODS 11. **Ciutats i comunitats sostenibles.**	X			
ODS 12. **Producció i consum responsables.**				X
ODS 13. **Acció pel clima.**		X		
ODS 14. **Vida submarina.**				X
ODS 15. **Vida d'ecosistemes terrestres.**				X
ODS 16. **Pau, justícia i institucions sòlides.**				X
ODS 17. **Aliances per aconseguir objectius.**				X

Dels objectius de desenvolupament sostenible [39] que s'observen a la taula B.1, Pack4You està relacionat amb:

- **Treball decent i creixement econòmic:** Aquest ODS busca la creació de llocs de treball dignes i un augment dels estàndards de vida per a tots, així com un creiximent econòmic mitjançant la modernització tecnològica i la innovació. Com que l'objectiu principal d'aquest projecte és facilitar la vida dels repartidors, s'estan millorant les seues condicions laborals. Gràcies al menor temps de repartiment i poder confiar en l'ordenació automàtica dels paquets, els repartidors estaran baix menys pressió i estrès. A més més, gràcies a l'ús de les noves tecnologies, el rendiment d'aquests és major i, per tant, el creixement econòmic es fa possible.

- **Ciutats i comunitats sostenibles:** Amb aquest objectiu, l'ONU busca fer de les ciutats zones menys contaminants, pel que és necessari reduïr la petjada de carboni de cada habitant. El meu projecte és capaç de reduir el temps de repartiment i la distància recorreguda, pel que també redueix la contaminació dels vehicles utilitzats. A més a més, com que ara poden repartir-se una major quantitat de paquets en un menor temps, caldran menys vehicles per oferir un millor servei als clients.

- **Acció pel clima:** Aquest objectiu busca augmentar la consciència i millorar l'educació respecte a la mitigació del canvi climàtic, així com augmentar l'interés per seguir un estil de vida sostenible amb el medi ambient. Amb la reducció d'emissions pel menor nombre de vehicles a la carretera, les empreses que utilitzen Pack4You estaran incentivant un estil de vida verd i sostenible.