UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONATECH

TREBALL FINAL DE GRAU

Wisebite

La gestió intel·ligent de restaurants

Autor:
Albert Suàrez

Director:
Ernest Teniente

Un treball final de grau corresponent
al Grau en Enginyeria Informàtica

en el

Departament d'Enginyeria de Serveis i Sistemes d'Informació

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

27 de juny de 2017

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Resum

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
Departament d'Enginyeria de Serveis i Sistemes d'Informació

Grau en Enginyeria Informàtica

Wisebite

by Albert Suàrez

La tecnologia està evolucionant de forma exponencial en els últims anys. Tot i així, sembla que hi ha un seguit de sectors en la societat que no s'acaben d'adaptar a aquest canvi. Un d'aquests sectors és el de la restauració. Ja poden existir sistemes d'Intel·ligència Artificial que et realitzen la comanda al supermercat de forma automàtica o bé aplicacions de Realitat Virtual que et permeten estar on vulguis, però encara avui en dia es veuen establiments com bars i restaurants que encara realitzen les comandes a bolígraf i paper. La pregunta que ens realitzem és: per què està passant això? En aquest treball final de grau s'ha estudiat els motius i s'ha analitzat com podria ser la millor solució pel problema comentat.

Wisebite, la plataforma que gestiona de forma intel·ligent els establiments de restauració, té com a objectiu solucionar aquesta problemàtica. Així, no només modernitzaria el sector, sinó que també podria oferir millors resultats en qualitat de servei, eficiència i beneficis econòmics als establiments que decideixin implantar-la. Aquesta aplicació, disponible en plataformes Android, et permetrà gestionar internament el teu establiment administrant les comandes dia a dia, et facilitarà un recull de les estadístiques per millorar el teu restaurant i els teus clients podran interaccionar amb l'establiment creant ells mateixos les comandes i valorant el teu restaurant.

Amb la realització d'aquest treball final de grau es pretén marcar èmfasi en el concepte que els informàtics tenim més poder en la societat del que pensem. Moltes aplicacions han aparegut del desconeixentment i han generat necessitats que mai pensàvem que existirien. *Wisebite* té com a objectiu principal millorar el món de la restauració aportant un sistema plenament genèric i personalitzable a l'usuari, que aquest es pugui fer seva la plataforma, i que amb ella pugui millorar el seu establiment. Convertir una simple aplicació en una eina més del dia quotidià de les persones relacionades amb el sector. Dit d'una altra manera, canviar el món.

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Resumen

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
Departament d'Enginyeria de Serveis i Sistemes d'Informació

Grau en Enginyeria Informàtica

Wisebite

by Albert Suàrez

La tecnología está evolucionando de forma exponencial en los últimos años. Sin embargo, parece que hay una serie de sectores en la sociedad que no acaban de adaptarse a este cambio. Uno de estos sectores es el de la restauración. Ya pueden existir sistemas de Inteligencia Artificial que te realizan el pedido en el supermercado de forma automática o bien aplicaciones de Realidad Virtual que te permitan estar donde quieras, pero aún hoy en día se ven establecimientos como bares y restaurantes que aún realizan los pedidos a bolígrafo y papel. La pregunta que nos realizamos es: ¿por qué está pasando esto? En este trabajo final de grado se ha estudiado los motivos y se ha analizado como podría ser la mejor solución para el problema comentado.

Wisebite, la plataforma que gestiona de forma inteligente los establecimientos de restauración, tiene como objetivo solucionar esta problemática. Así, no sólo modernizaría el sector, sino que también podría ofrecer mejores resultados en calidad de servicio, eficiencia y beneficios económicos a los establecimientos que decidan implantarla. Esta aplicación, disponible en plataformas Android, te permitirá gestionar internamente tu establecimiento administrando los pedidos día a día, te facilitará una recopilación de las estadísticas para mejorar tu restaurante y tus clientes podrán interaccionar con el establecimiento creando ellos mismos los pedidos y valorando tu restaurante.

Con la realización de este trabajo final de grado se pretende poner hincapié en el concepto de que los informáticos tenemos más poder en la sociedad de lo que pensamos. Muchas aplicaciones han aparecido del desconocimiento y han generado necesidades que nunca pensábamos que existirían. *Wisebite* tiene como objetivo principal mejorar el mundo de la restauración aportando un sistema plenamente genérico y personalizable al usuario, que éste se pueda hacer suya la plataforma, y que con ella pueda mejorar su establecimiento. Convertir una simple aplicación en una herramienta más del día cotidiano de las personas relacionadas con el sector. Dicho de otro modo, cambiar el mundo.

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Abstract

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
Departament d'Enginyeria de Serveis i Sistemes d'Informació

Grau en Enginyeria Informàtica

Wisebite

by Albert Suàrez

Technology has been evolving exponentially in last years. However, it seems that there are a lot of fields in society that have not adapted to this change yet. One of these fields is the restoration. There are Artificial Intelligence systems that make the order in the supermarket automatically or there are Virtual Reality applications that allow you to be where you want, but still today you can see establishments like bars and restaurants that still work with pen and paper. The question that we might ask ourselves is: why is this happening? In this degree final project the reasons have been studied and it has been analysed how it could be the best solution for the commented problem.

Wisebite, the platform that intelligently manages restoration establishments, wants to solve this problem. Thus, it would not only modernize the sector, but could also offer better results in terms of service quality, efficiency and economic benefits to establishments that decide to implement it. This application, available on Android platforms, will allow you to manage internally your establishment managing the orders day by day, will provide a set of statistics to improve your restaurant and your customers can interact with the establishment by creating the orders themselves and valuing your restaurant.

With the completion of this degree final project, it is intended to emphasize in the concept that we have more power in society than we think. Many applications have appeared out of ignorance and generated needs that we never thought would exist. *Wisebite*'s main objective is to improve the world of restoration by providing a fully generic and customizable system to the user, so that the user can make the platform his own, and with it can improve its establishment. Convert a simple application into a tool of the daily day of the people related to the sector. Put another way, change the world.

Índex

Capítol 1

Introducció

Aquest projecte és un Treball Final de Grau en Enginyeria Informàtica a la Facultat d'Informàtica de Barcelona (*Universitat Politècnica de Catalunya*). Un projecte amb la finalitat principal de convertir un establiment de restauració qualsevol en quelcom intel·ligent, podent gestionar de forma més eficient, còmode i professional les seves comandes, podent-les analitzar posteriorment i interactuant de forma més activa amb el client de l'establiment.

1.1. Contextualització

El món de la restauració va néixer molts segles enrere i amb el transcurs de la història ha anat evolucionant proporcionalment amb l'evolució del ser humà i els seus costums. Tot i així, el que és clar és que el concepte d'anar a prendre quelcom al bar durant algun moment del dia es segueix mantenint per molt temps que passi, almenys a Espanya.

La tecnologia ha estat quelcom que sempre ha existit, però no amb tanta importància i impacte com té actualment. Des del naixement de l'*smartphone* [1] el 1992 quan IBM va treure el primer pilot de telèfon mòbil amb funcionalitats de PDA incorporades, s'ha anat instaurant a les nostres vides de manera exponencial fins al punt on és pràcticament una part nostra, que sense ella no seria el mateix. D'aquest aspecte se li pot treure tant punts favorables com negatius. Entre els positius tenim sistemes similars als del projecte que estem tractant, el qual dóna infinitats d'avantatges respecte al sistema convencional. Avui en dia, en el context de la societat actual en què vivim, sistemes intel·ligents implantats en els establiments de restauració es veuen a comptagotes, i no pas perquè les plataformes existents siguin dolentes o precàries, sinó per un altre seguit de factors. Factors com pot ser l'impacte econòmic que implica la instauració d'un sistema d'aquestes característiques, la manca d'adaptabilitat al canvi o bé la complexitat d'alguns establiments.

El fet és que des de principis de segle els costums humans canvien amb una rapidesa realment diferent de la de fa segles, tan ràpidament que el món de la restauració en conjunt no s'ha pogut adaptar. El naixement de les noves tecnologies i el poder que tenen avui en dia a la societat es veu reflectit en bars i restaurants, on algun d'ells (i cada cop més) la utilitzen en el negoci. Tot i així, en l'estudi d'aquest tòpic, apareix un seguit de qüestions gens menyspreables, les quals han de ser tractades.

1.1.1. Factor econòmic

Un dels principals problemes per als quals aquest sector no s'ha acabat d'adaptar és el cost de la implantació de la tecnologia en un establiment d'aquestes característiques. Cada restaurant o bar és únic en referència a la resta, per tant, cada un d'ells necessita un sistema adaptat a les seves necessitats, i això es fa pagar.

El desenvolupament d'un sistema genèric és notablement més barat respecte un específic, ja que l'equip encarregat de construir pot vendre-ho posteriorment a més d'un client, així doncs pot ajustar més el preu. En canvi, si estem parlant d'un sistema totalment personalitzat i especialitzat per un establiment, llavors el cost puja considerablement, ja que han de cobrir els costos del disseny i la implementació del sistema.

És aquí doncs on s'estableix un dels tres grans problemes que fa que sistemes d'aquestes característiques no es vegin avui en dia en els establiments de restauració. A més a més se li suma l'època de crisi econòmica viscuda que fa complicar el panorama. Només els establiments que aconsegueixen facturar grans quantitats es poden permetre sistemes com el que comentem.

1.1.2. Adaptació al canvi

En aquest sector ens podem trobar molts tipus d'usuaris. Perfils de gent que sempre busquen ser millors en el sector i fan el possible per estar actualitzats amb la tecnologia d'aquell moment. En canvi, existeixen nombrosos casos d'establiments on els responsables d'aquests no tenen facilitat per adaptar-se al canvi, és a dir, que se satisfan amb el procediment de negoci que sempre han tingut i sempre els ha funcionat, encara que sigui antiquat. Amb dificultats com aquestes no és fàcil instaurar un sistema d'aquestes característiques, ja que els usuaris que la utilitzen no s'adaptarien i, en conseqüència, tindria represàlies negatives.

La implantació de tot sistema en una corporació o empresa, ja sigui enfocat en el món de la restauració o bé en un altre sector, no només recau en la compra de material *hardware* i *software*, sinó que també recau en la implicació dels treballadors que hauran d'interactuar amb aquest nou sistema. Per tant, és obligació dels responsables de tot establiment que vulgui implantar un sistema d'aquestes característiques motivar a l'equip de treballadors en aquest aspecte. La plataforma a implantar ja pot ser molt bona, però si no hi ha voluntat de l'equip en utilitzar-la correctament, molt probablement el procés acabarà en fallida.

1.1.3. Complexitat

Tots els establiments d'aquest sector funcionen de maneres molt diferents acord a les seves característiques i funcionalitats. Alguns d'ells disposen de sistemes molt complexos i complicats en comparació de la competència, cosa que aporta dificultat en la implantació d'un sistema d'aquest tipus. I en contrapartida, implantar una plataforma d'aquestes característiques en un establiment molt simple com pot arribar a ser un bar de poble tampoc acaba de ser del tot útil.

En conseqüència, podem tenir dues situacions que impedeixen el *boom* d'aquests sistemes en el món de restauració. Per una banda, restaurants molt complexos que incapaciten crear un sistema que ho controli tot de forma fàcil. I per altra banda, bars molt simples o senzills que mai s'arribaran a plantejar sistemes d'aquestes característiques.

1.2. Motivació

Durant el quadrimestre anterior a l'inici del Treball Final de Grau vaig estar rumiant profundament cap a on volia encaminar el projecte. Tenia disponible la capacitat de realitzar-ho en l'empresa en la qual estava treballant, i actualment segueixo. Tot i així, donades unes circumstàncies que es comentarà a continuació, vaig decidir encaminar-me a realitzar el projecte de *Wisebite*.

En el meu context familiar i d'amistats he tingut sempre molt present la cultura de la gastronomia, com bé caracteritza el nostre país. Tot i així, amb els anys he anat coneixent persones que es dediquen professionalment al món de la restauració, sigui cambrers, cuiners o administradors d'establiments del sector. En anar amb aquest tipus de persones a prendre quelcom o a menjar un àpat, em feien veure diferent l'establiment de com ho feia abans. Molts cops ens centràvem més en com era el servei i com estava muntat internament la cadena de producció dins l'establiment que pas gaudir de l'àpat que ens estaven servint.

En conseqüència, arran d'això i dels coneixements tècnics que he anat adquirint durant el grau aquests quatre anys, vaig estar pensant com es podria aplicar la tecnologia en establiments d'aquest tipus per tal de millorar la seva eficiència i poder oferir un millor producte als clients.

Per dissenyar i construir una idea més autèntica vaig estar parlant amb aquest grup de persones, que s'ha comentat anteriorment, i se'ls va preguntar com funcionaven els seus establiments i que realitzarien per poder millorar-los al que correspon a la gestió del bar o restaurant. Una gran majoria d'aquests em va comentar que van implantar un sistema de gestió de comandes a través d'un mòbil o tauleta, però que els havia costat molt temps acabar-ho implantant donat al cost que comportava. I no només això, sinó que els hi va costar bastant adaptar-se a causa de la poca usabilitat que tenia el sistema que utilitzaven.

Així doncs, vist quin era l'estat actual, vaig decidir-me a realitzar un estudi de mercat analitzant quines aplicacions i plataformes existien en aquell moment (apartat que comentarem posteriorment) i em vaig adonar que hi havia molta feina per fer. Les aplicacions que aplicaven la filosofia de *Wisebite* estaven bastant obsoletes i no disposaven d'una interfície d'usuari que tendia a la usabilitat, fet que dificultava l'adaptabilitat al canvi dels usuaris.

En conclusió, després d'estudiar bé la proposta vaig parlar l'*Ernest Teniente* i va acceptar ser el meu director d'aquest projecte i Treball Final de Grau.

Capítol 2

Estat de l'art

Per conèixer millor el potencial d'aquest mercat i saber les alternatives per a una major profunditat en el coneixement del sector, s'ha de realitzar un estudi del mercat existent per a comprovar la presència d'aplicacions que són similars, sigui per objectiu o per mercat, a *Wisebite*. A continuació apareixen algunes de les aplicacions que s'han pogut trobar.

El fet d'analitzar cadascuna d'elles et permet veure quines funcionalitats pots oferir al client perquè directament no existeix cap eina que les faciliti o bé per millorar les existents. S'ha volgut destacar sis plataformes similars a *Wisebite*, algunes en millors aspectes que altres. En acabar de valorar cadascuna d'elles, es realitzarà un estudi ja més globalitzat que permeti veure quins avantatges ofereix *Wisebite* al mercat actual.

2.1. Waiterio

Aplicació[2] orientada especialment a substituir el *TPV* d'un bar o restaurant. Disposa de funcionalitats com creació de menús i comandes, convidar companys de feina amb rols associats, visualització en mòbil i tauleta, reports periòdics i generació de factures totals o fraccionades.

L'aplicació disposa d'entre unes 50.000 i 100.000 descàrregues al *Play Store*. En general, conté moltes funcionalitats i té una interfície d'usuari ben cuidada que permet l'ús de l'aplicació de forma més còmode i confortable.

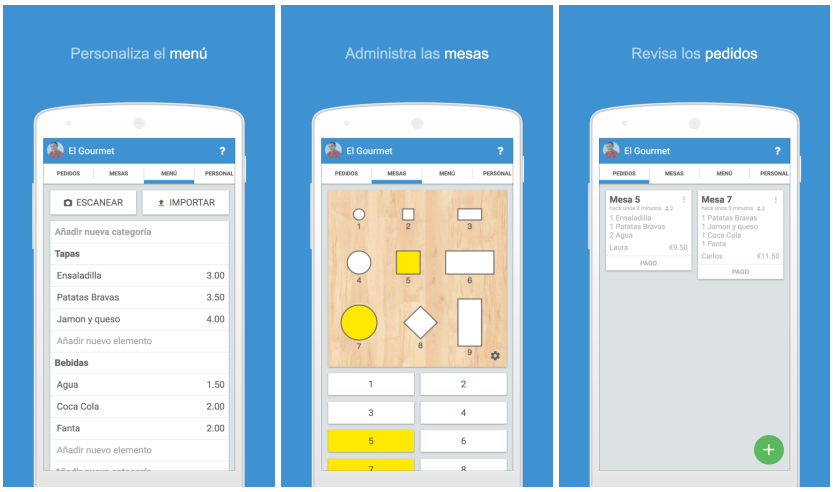

FIGURA 2.1: Captures de pantalla de Waitero

2.2. PrimeTray

Aplicació[3] orientada a les comandes del client a l'establiment. L'usuari té la capacitat de seleccionar un restaurant de la llista i fer la comanda en línia i anar-ho a buscar un cop és notificat. A més a més pot visualitzar l'estat en temps real de la seva comanda, és a dir, si està preparada o encara queda temps per poder-la rebre.

L'aplicació té entre 500 i 1.000 descàrregues a la botiga d'aplicacions d'*Android*. Tot i així té un 4.7 de valoració per part dels usuaris. En general l'aplicació està molt ben enfocada pel seu objectiu, és a dir, millorar la comoditat del client en els establiments de restauració als quals acudeix. Té una gestió de la interfície d'usuari prou bona que aconsegueix que sigui més fàcil utilitzar i familiaritzar-se amb l'aplicació. En contrapartida, les funcionalitats d'aquesta aplicació són bastants escasses. Fet que aconsegueix que perdi molts punts en comparació altres empreses.

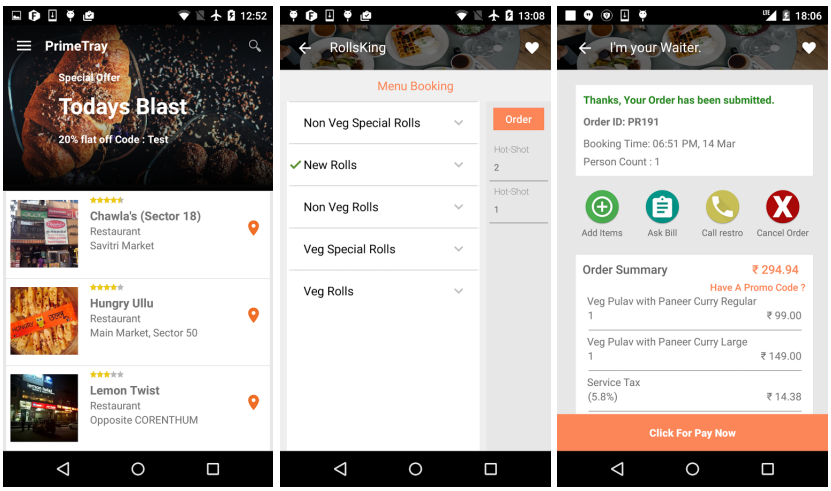

FIGURA 2.2: Captures de pantalla de PrimeTray

2.3. OrderSev

Aplicació[4] orientada especialment a substituir el *TPV* d'un bar o restaurant. Disposa de funcionalitats com creació de menús i comandes, generació de factures i vistes tant des de cuina com del cambrer. Gran punt a favor que disposa aquesta aplicació és la vista web de la qual disposa. Pots accedir de forma multiplataforma segons l'objectiu que tens amb l'aplicació.

Disposa d'entre 500 i 1.000 descàrregues, i amb una nota mitjana de 5.0, però amb només 17 votacions, ha acabat sent una aplicació funcional però amb no gaire repercussió. El punt en contra que és que no li han dedicat el temps en la interfície d'usuari que comporta. Disposa d'unes vistes molt simples i molt poc estètiques.

La gestió de la interfície d'usuari és quelcom molt important a l'hora de valorar una aplicació. Pel simple fet que els usuaris que la utilitzin no es fixaran en si l'algorisme i les tecnologies utilitzades són molt bones o no, sinó en quin aspecte té l'aplicació. És per això doncs que aquesta aplicació no ha acabat de triomfar, pel simple fet que la interfície no era l'adequada pel tipus d'usuari amb el qual treballava.

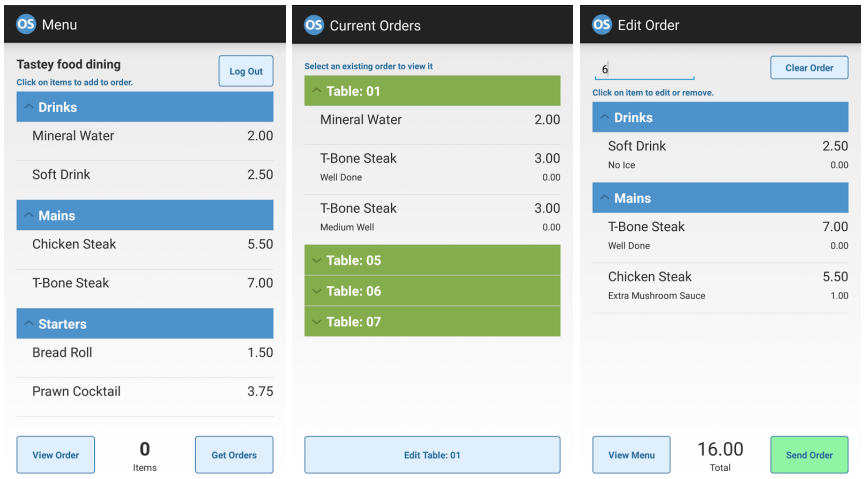

FIGURA 2.3: Captures de pantalla de OrderSev

2.4. TabletWaiter

Aplicació[5] orientada a ser un estil de carta per a l'establiment. Cada taula d'un restaurant hauria d'haver-hi un dispositiu amb aquesta aplicació activa a on es pugui consultar els plats i seleccionar-los, així es rebria a cuina i ja podrien començar a preparar la comanda. També permet cridar al cambrer via aplicació i demanar el compte. Per altra banda, també disposa de la possibilitat que un cambrer la utilitzi per crear la comanda. És a dir, l'aplicació està preparada per ambdós perfils, ja sigui client com cambrer. Permet accedir-hi en la seva versió web, a on es pot crear els menús i parametritzar com vulguis el teu bar o restaurant.

Aquesta aplicació, amb un conjunt de descàrregues situat entre 1.000 i 5.000 i un valor de 3.7 com a puntuació dels usuaris, es situa en un bon lloc en el mercat donat les funcionalitats que disposa. Tot i així, no ha acabat mai de despuntar a causa de la seva gestió de la interfície d'usuari. Les aplicacions per a dispositius mòbils han anat evolucionant en el que el seu disseny es refereix cap a un punt on totes les plataformes segueixen els mateixos patrons de disseny, patrons com *Material Design*[6]. Fet que no aplica aquesta aplicació. La plataforma disposa d'una interfície que no aplica cap tipus de patró de disseny, fet que dificulta la usabilitat de l'usuari.

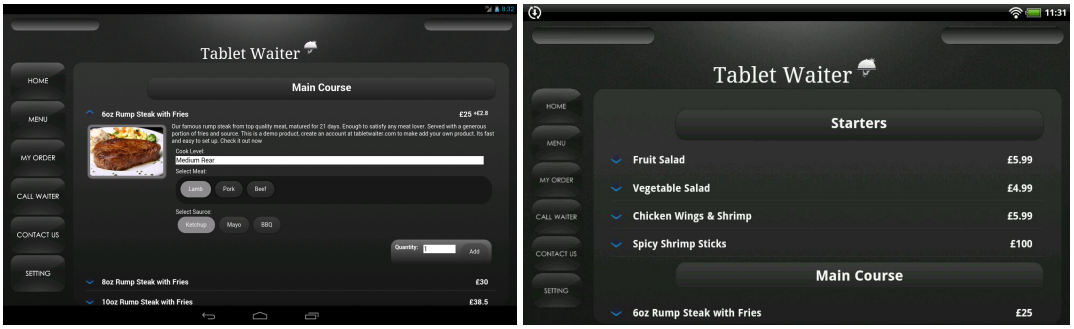

FIGURA 2.4: Captures de pantalla de TabletWaiter

2.5. Cloud Waiter

Aplicació[7] orientada a les comandes del client a l'establiment. L'usuari té la capacitat d'escanejar un codi QR amb el qual podrà accedir a l'aplicació i realitzar la comanda. Així doncs l'usuari client que acudeix a l'establiment podrà estar-hi el temps que vegi precís i podrà demanar el que vulgui sense necessitat d'esperar al cambrer que l'assisteixi. A part, també disposa de la funcionalitat de poder cridar al cambrer perquè s'acosti a la taula tant per consultes com per demanar el compte.

Plataforma amb un volum de descàrregues d'entre 1.000 i 5.000 i una valoració de 4.6 sobre 5 per part dels usuaris, es converteix en una aplicació bastant correcta en l'àmbit en el qual s'especialitza, és a dir, millorar l'experiència del client en l'establiment. Malauradament aquesta aplicació no ha acabat de sortir a la llum pel mateix problema que s'ha comentat en l'aplicació anterior, o sigui, la dolenta gestió d'interfície d'usuari.

FIGURA 2.5: Captures de pantalla de OrderSev

2.6. FastOrder

Aplicació[8] orientada a les comandes del client a l'establiment. L'usuari té la capacitat d'escanejar un codi QR amb el qual podrà accedir a l'aplicació i realitzar la comanda i tot el que sigui necessari. Diferents usuaris poden accedir a la mateixa comanda i modificar-la segons vegin convenient, fet que permet que cadascun dels clients d'una mateixa taula puguin fer la comanda de forma paral·lela.

Aplicació amb més de 5.000 però amb menys de 10.000 descàrregues i amb una valoració de 4.7 per part de la comunitat d'usuaris, es situa amb una de les que disposa de més usuaris en comparació de les que s'han comparat. Aquesta plataforma disposa de funcionalitats un xic limitades en comparació de la competència, però en contrapartida té una interfície d'usuari molt ben cuidada. Fet que la permès situar-se amb el nombre de descàrregues que disposa actualment.

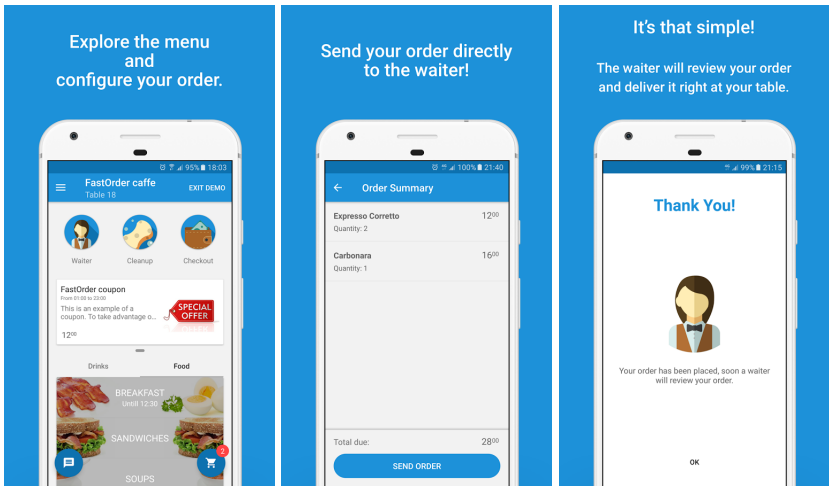

FIGURA 2.6: Captures de pantalla de OrderSev

2.7. Conclusions

Encara que hi ha moltes aplicacions relacionades amb aquest àmbit (si bé, amb objectius i funcionalitats molt diverses, en alguns casos molt dispars al projecte), s'ha fet una selecció prou representativa però tot i així reduïda, amb l'objectiu de realitzar una correcta anàlisi que permeti treure conclusions de forma còmoda i eficaç.

Un cop realitzada aquesta selecció de potencials competidors de l'aplicació, i comprès (a grans trets) les seves funcionalitats i objectius, procedim a un estudi detallat comparant-ho amb la visió d'aquest projecte.

El que s'ha pogut veure estudiant el mercat és que hi ha dos grans grups d'aplicacions. Per una banda, tenim els sistemes que només se centren en la gestió interna de l'establiment i poder realitzar comandes més fàcil i eficientment. Per altra banda, tenim les plataformes que permeten al client, que acudeix a l'establiment, viure una millor experiència i estar còmode en la seva estança. Resumint, hi ha aplicacions que milloren a l'empleat de l'establiment i altres que aporten comoditat al client.

Altre punt realment important, i que s'ha pogut demostrar en l'anàlisi de cada una de les aplicacions anteriors, és la gestió de la interfície d'usuari (*GUI*). Gran part dels usuaris que podrien arribar a utilitzar aquest tipus d'aplicacions o plataformes són usuaris no especialitzats en la tecnologia, també coneguts per l'anglicisme *laypeople*. En conseqüència, és realment vital centrar-se a construir una bona experiència d'usuari i dissenyar una aplicació que sigui fàcil d'utilitzar, intuïtiva i fàcil d'aprendre per qualsevol tipus de perfil d'usuari. Un exemple podria ser *FastOrder*, una aplicació que escasseja clarament de funcionalitats si ho comparem amb la resta d'aplicacions del mercat, però en canvi es situa la segona amb més descàrregues dins del *Play Store*. Per què? Perquè al cap i a la fi el primer impacte que reps d'una aplicació, sigui mòbil o web, és la interfície d'aquesta. Si la plataforma disposa d'una bona experiència per l'usuari, aquest la seguirà utilitzant de forma genèrica i fins i tot l'acabarà recomanant al seu cercle.

En conclusió, un cop estudiat quin és el mercat, es veu clar quina és la tendència i que pot aportar *Wisebite* a aquest sector.

Per un cantó, les nombroses funcionalitats que disposen les aplicacions comentades es podrien agrupar en tres grans grups: gestió de l'establiment, anàlisi de dades i relació amb el client. El primer d'ells es basa en tot el que engloba la digitalització de l'establiment, el segon és l'explotació de les dades digitalitzades per a la millora en l'eficiència i gestió del bar o el restaurant i per últim la millora en l'experiència del client en l'establiment. El fet és que no existeix cap plataforma que implanti les tres funcionalitats, com a molt dues d'elles. *Wisebite* té la intenció d'adaptar les tres en una sola plataforma, i no només això, sinó millorar les prestacions de cada un d'elles.

Per altra banda, i com ja s'ha comentat en estudiar cada un dels exemples anteriors, *Wisebite* té molt a aportar en el que la gestió de la interfície d'usuari es tracta. No només es vol disposar d'un sistema òptim des d'un sentit computacional, sinó tenir una interfície d'usuari usable, intuïtiva i comprensible per qualsevol tipus de client, sobretot fent èmfasi en el sector *laypeople*.

Així doncs, *Wisebite* aportarà un nou estil d'aplicació desconegut en el sector fins al dia d'avui que facilitarà el dia a dia a tota persona relacionada, ja sigui client com empleat.

Capítol 3

Definició de l'abast

Un sistema d'aquestes característiques podria tenir infinitat de requisits i funcionalitats, és per això que és important definir l'abast d'aquest projecte i veure quins són els objectius d'aquest.

3.1. Objectius

L'objectiu principal d'aquest Treball Final de Grau és dissenyar i construir un sistema que aconsegueixi fer la vida més fàcil als empleats de qualsevol establiment de restauració, sigui bar o restaurant, i crear una millor experiència per a qualsevol usuari d'aquests locals, és a dir, millorar la seva estança oferint-li eficiència en el servei i millor tracte per part de l'establiment.

Com ja s'ha comentat anteriorment, l'elevat cost d'implantar un sistema com aquest és la dificultat més gran. És per això que un altre objectiu important és poder construir una plataforma plenament genèrica què la puguis personalitzar al teu gust i, en definitiva, fer-la teva. Així aconseguiràs reduir una quantitat abismal en costos i obtenir majors beneficis gràcies a les funcionalitats d'aquest sistema. Perquè com bé s'ha comentat abans el cost d'implantar un sistema d'aquestes característiques no resideix en la compra d'elements tecnològics com dispositius mòbils, sinó en la compra d'un software especialitzat.

Per altra banda es buscarà oferir la millor experiència d'usuari possible de tal manera que el cost d'aprendre a utilitzar aquesta aplicació sigui el mínim.

Un cop especificada, dissenyada i implementada la plataforma de *Wisebite* s'intentarà llançar a establiments del cercle familiar i d'amistats per així rebre un feedback de quin és el funcionament de l'aplicació en una situació real. I finalment, si la plataforma té un bon impacte, estudiar la viabilitat econòmica del projecte.

3.2. Abast

En el capítol anterior s'ha analitzat les funcionalitats de les aplicacions existents en el mercat actual. En realitzar-ho, s'ha vist que totes elles es podien classificar en tres grans grups. El sistema final estarà format per aquests tres components molt importants que li donaran valor al producte resultant, i marcarà la diferència respecta la competència del mercat, tal com s'ha comentat en l'estudi anterior.

3.2.1. Gestió de comandes

En primer terme, el sistema serà capaç de gestionar les comandes d'un establiment de restauració. La plataforma tindrà la capacitat de crear menús i plats personalitzats, amb les preferències i opcions desitjades. Cadascun d'aquest contindrà informació vital i d'importància com el preu i una petita descripció del seu contingut. Com a empleat de l'establiment que té implantat *Wisebite*, podrà crear comandes fàcilment i sense cap problemàtica, amb l'ajut d'una interfície molt intuïtiva. Totes les peticions seran rebudes de forma automàtica a cuina amb una interfície còmoda i agradable per tal d'agilitzar el procés el màxim possible. Aquesta informació s'anirà actualitzant en temps real sense necessitat d'actualitzar-ho manualment.

Un cop implantat aquest component del sistema s'estarà aconseguint una millora notable en l'eficiència de les comandes. Això provocarà una satisfacció per part de la clientela, ja que rebran les comandes sol·licitades abans, que esdevindrà a uns majors ingressos per a l'establiment donat que acudirà més gent gràcies a la possible fama que es pugui generar per aquest fet.

3.2.2. Anàlisi de l'establiment

L'avantatge més important, i amb diferència, d'emmagatzemar les dades digitalment és la facilitat de realitzar un estudi detallat d'aquestes dades. El sistema tindrà la capacitat de convertir aquestes dades sense massa significat a priori a una font d'informació que serà de gran utilitat per als responsables de l'establiment.

Per què són tan importants les dades?, es podria arribar a preguntar qualsevol. Anteriorment es prenien decisions molt importants a partir de l'experiència de les persones i de la percepció de negoci. En canvi, amb les dades en el nostre poder, es poden prendre decisions objectives i no subjectives, ja que es prenen via dades reals del consumidor. És a dir, coneixem més al nostre client.

De forma periòdica, el sistema reportarà resums on es reflectirà informació de gran valor per a l'establiment. Informació com pot ser el tràfic setmanal, els plats més demanats, ingressos i despeses, comandes per empleat i així un llarg etcètera. Amb aquesta informació disponible esdevindrem al coneixement, és a dir, els responsables del bar o restaurant tindran la capacitat de prendre decisions a partir d'aquesta informació. Decisions que aportaran valor a l'establiment en concret i oferir un millor servei al client, així augmentant els ingressos del bar o restaurant.

3.2.3. Relació amb el client

L'última component té com a objectiu crear un fort vincle entre l'establiment i el client que hi acudeix. Qualsevol usuari d'aquesta aplicació podrà buscar l'establiment que desitgi i veure informació sobre ell com imatges, plats més demanats, valoracions i més informació que li permeti conèixer tot respecte al bar o restaurant sense necessitat d'anar-hi presencialment. Com usuari o client d'aquest establiment, tindrà la possibilitat de demanar la comanda via plataforma amb un simple escaneig d'un codi QR que haurà col·locat a cada una de les taules. Un cop acabada la visita podrà valorar el servei acompanyat de comentaris i imatges de suport per així millorar la comunitat d'usuaris de l'aplicació.

Aquesta component aportarà flexibilitat i comoditat per a tot tipus d'usuari de l'establiment. L'atendran més ràpidament, podrà realitzar la comanda sense pressa i rumiar ben bé que és el que al final es voldrà demanar.

3.3. Obstacles i riscos

Durant tot projecte, sigui de curta o llarga durada, sempre poden sorgir un seguit d'obstacles i imprevistos que poden condicionar el resultat final d'aquest, ja que té un temps limitat per construir-lo. En cas del Treball Final de Grau, estem parlant d'uns quatre mesos aproximadament.

El principal obstacle, que pot esdevenir i serà clau de controlar, serà la gestió del temps. El Treball Final de Grau s'ha de cursar en un període limitat i s'ha d'intentar ajustar a aquest. Caldrà fer una planificació temporal el més realista possible i anar fent punts de control periòdics perquè no es descontroli l'abast del projecte i que la planificació s'adeqüi a la realitat. En cas que en aquests controls ens adonem que s'està desviant en excés es prendrà mesures segons el cas per poder-ho remeiar.

Un altre possible obstacle que pot aparèixer és el desconeixement d'algunes tecnologies necessàries per al desenvolupament del projecte, tecnologies que s'explicaran amb més detall en capítols posteriors. Hi ha característiques i funcionalitats que tindrà el sistema que requereixen uns coneixements tècnics per a poder-les fer realitat, les quals no disposa l'autor del projecte en nivell expert. Pot donar-se el cas també que en realitzar la planificació temporal d'una tasca es calculi un temps X que finalment no es podrà complir donat el desconeixement exacte del cost corresponent, ja que no es desconeixia la tecnologia.

Un altre inconvenient o imprevist que pot sorgir és una gran quantitat d'errors en el codi durant la fase d'implementació del projecte. Sempre solen aparèixer nombrosos errors durant el desenvolupament d'un projecte, però solen haver-hi alguns que requereixen una dedicació extra per solucionar-los. S'ha de saber filtrar quins errors són vitals de solucionar i quins es poden evitar i esquivar-los.

I finalment pot donar-se el cas, tot i ser molt remot, que alguns dels serveis externs que utilitza la plataforma quedin inhabilitats durant un període que dificulti el desenvolupament i testeig de l'aplicació.

Tots aquests possibles problemes que poden sorgir durant el període d'aquest Treball Final de Grau han de ser tractats immediatament de tal manera que no es converteixin en problemes crítics difícils de solucionar. El pla d'acció per solucionar-los en cas que passin es comentarà de forma més detallada en capítols posteriors.

3.4. Metodologia i rigor

Durant el transcurs de la carrera, i en especial l'especialitat d'Enginyeria del Software, he après un seguit de metodologies de treball però, sense dubte, em quedaria amb tot el grup de tècniques que engloben les metodologies àgils. En conseqüència, aquest projecte seguirà una metodologia àgil, en concret una inspirada en la metodologia *Scrum*.

Inicialment, en les primeres setmanes esdevindrà la Fase Inicial del projecte, que ve a ser l'assignatura de Gestió de Projectes (*GEP*). En ella s'especificarà tot el necessari per al sistema que es vol construir. En concret, s'especificarà per un cantó la definició de l'abast i la contextualització del projecte, per altra banda la planificació temporal i finalment la gestió econòmica i de sostenibilitat. A més a més, per tal de situar-se en el punt inicial i poder fer una petita prova de com pot arribar a ser la lectura final, es realitzarà una presentació de cinc minuts explicant el realitzat en aquesta fase inicial.

Un cop tot especificat vindrà la fase intermèdia del projecte, dit d'altra manera, la fase de desenvolupament d'aquest. En ella es crearan un seguit d'*Sprints*, seguint la metodologia àgil, d'unes dues setmanes de duració que s'analitzaran en una retrospectiva posterior amb el director del projecte. Amb aquests *Sprints* serà fàcil validar la feina feta i veure com va el projecte, és a dir, si la planificació inicial realitzada s'adequa al pas del projecte. Cada una d'aquests *sprints* contindrà un conjunt d'*Històries d'usuari* que representaran la feina a realitzar durant aquelles dues setmanes. Una *història d'usuari* es pot entendre com una petita part funcional d'un projecte, és a dir, una funcionalitat d'aquest. La gràcia de dividir-ho en històries d'usuari és poder separar tota la feina que comporta un projecte en tasques molt petites i, a priori, independent entre elles. A més a més, a cada una d'aquestes històries d'usuari se li atribuirà una puntuació que representarà el cost d'implementació d'aquestes, per així poder prioritzar-les segons la disponibilitat del moment.

A la vegada es registrarà totes les accions que es van fent per controlar quant temps es tarda a realitzar cada una de les tasques que es planteja fer. Així, en cada una de les retrospectives, es podrà analitzar si les puntuacions atorgades a cada una de les targetes o bé històries d'usuari són correctes o no.

S'estipularà una nomenclatura i unes convencions fixes durant el desenvolupament del projecte per així tenir un millor control de quines històries d'usuari s'estan realitzant en aquell precís moment i, en general, disposar d'un historial de desenvolupament amb un format únic. Així doncs, en finalitzar la implementació es disposarà d'un codi net i que simplement llegint-ho s'entendrà el seu funcionament.

3.4.1. Metodologia de desenvolupament: convencions de git

Commits
Missatges en anglès i en minúscules començant per un verb comú com per exemple "*add*", "*remove*", "*update*", "*refactor*", "*fix*"o un qualsevol altre, que descrigui en poques paraules l'acció realitzada en el commit, així serà clar el que s'ha modificat sense necessitat d'entrar a veure els canvis.

Feature branch workflow
Model de desenvolupament basat en les "*branches*"(branques) de git. Ajuda a realitzar un desenvolupament incremental del producte, fomenta la cronologia del codi i minimitza els possibles conflictes entre versions quan es desenvolupen. S'utilitza un repositori central com a història oficial del projecte, història que funciona sense cap tipus d'error. Internament, s'usen 2 tipologies de "*branch*":

- *Master branch*: és on està l'historial de codi acceptat i acabat del projecte. És única. Tot funciona dins d'aquesta branca.

- *Feature branches*: branches separades de la master on es realitzen els commits per desenvolupar una història d'usuari del projecte o per resoldre un problema oposat. El nom ha de ser descriptiu sobre el que s'està desenvolupant perquè al llegir-ho es sàpiga què es tracta en aquesta branch.

Una vegada acabat el desenvolupament en un feature branch s'inicia una Pull request, explicat posteriorment, per ajuntar-la amb la branca principal anomenada master. Així doncs aconsegueixes separar la part de codi funcional amb la que ho acabarà sent.

Pull requests
Una pull request és una conversa sobre els canvis realitzats en una branch abans d'ajuntar-ho amb la base genèrica de codi que és la branch master. Això afavoreix una revisió del codi per evitar possibles problemes amb aquell fragment modificat o afegit. I a més poder documentar d'alguna manera el transcurs o evolució del codi.

3.4.2. Metodologia de desenvolupament: gestió de tiquets

Cada vegada que s'hagi detectat algun problema en el codi de l'aplicació que està en la branca principal master s'obrirà un "*issue*"en GitHub, portal que s'explicarà amb deteniment en capítols posteriors. Per tancar-ho o esmentar-ho s'aprofitarà la sintaxi pròpia de GitHub amb els commits:

- *Closes #12* (on 12 és el nombre del issue corresponent). Amb un commit que té aquest missatge aconseguim tancar el "*issue*"determinat.
- *#12* (on 12 és el nombre del issue corresponent). Amb un missatge d'aquest estil aconseguim referir a el ïssueïndicat.

3.4.3. Metodologia de desenvolupament: trello

Per gestionar el projecte seguirem Trello a causa de la seva semblança a les metodologies àgils. El flux de les històries d'usuari passarà per dos taulers dins l'aplicació Trello:

- *Backlog*: tauler únic durant tot el projecte. És el punt d'entrada de qualsevol requisit del projecte. Té 3 llistes, per les quals les històries han d'anar passant abans de poder ser assignades a un sprint. Aquestes llistes corresponen a les diferents fases d'especificació fins que és cada una de les històries d'usuari està totalment definida per tal de ser llançada a algun dels sprints del projecte.
  - Història d'usuari genèrica
  - Història d'usuari amb criteris d'acceptació
  - Història d'usuari amb criteris d'acceptació i punts d'història
- *Taulers de Sprint*: taulers propis, per cada un dels cinc sprints que tindrà el projecte, on s'afegeixen les històries a realitzar que s'hagin decidit en el *Sprint Planning*, considerant aquest concepte com el fet de decidir quines de les històries d'usuari del backlog van a aquesta iteració. Aquestes històries han de sortir del backlog, de la llista d'històries estimades. Per facilitar i uniformar la creació d'aquests taulers, existeix una plantilla de tauler de trello del que es realitzarà una còpia per crear cada tauler. Una vegada copiat es personalitza

segons les necessitats d'aquest sprint durant el *Sprint Planning*. Aquesta plantilla podrà ser modificada a final de cada sprint en la retrospectiva, per poder millorar el flux de treball intern de cada sprint.

3.4.4. Metodologia de desenvolupament: codi Java

Per desenvolupar en Java, ja que la tecnologia Android s'escriu en aquest llenguatge de programació, es seguirà una convenció de noms comú en Java segons estipula la comunitat de desenvolupadors.

- *Funcions i variables*: Escrites en el format CamelCase[9] amb la primera lletra en minúscula.
- *Constant*: Escrites en majúscules i separant paraules amb "_".
- *Classes*: Escrites en el format CamelCase amb la primera lletra en majúscula.
- *Paquets*: Escrits en minúscula i una sola paraula.
- *Documentació*: Seguirem el format de JavaDoc[10].

Capítol 4

Anàlisi de requisits

4.1. Agents implicats

En tot projecte en el qual ens podem trobar, inclòs un treball final de grau com en el que ens trobem, un conjunt o col·lectiu de persones afectades de forma directa o indirecte.

En concret, en el que es refereix *Wisebite*, destaca en especial un dels punts comentats en capítols anteriors. Dins les tres problemàtiques per les quals els usuaris no disposaven d'una implantació d'un sistema de gestió en el seu establiment de restauració és per l'elevat cost que té, és a dir, el problema ve degut per un factor econòmic. Aquest col·lectiu de persones que es veuen dins d'aquesta problemàtica no és pas un grup petit, sinó tot el contrari. És per això que és molt important valorar en aquest projecte quins són els agents implicats.

Les parts interessades o stakeholders[11] d'un projecte són aquelles persones o agrupacions de persones que el projecte els afecta de manera directa o indirecta. Aquests grups poden tenir objectius totalment diferents entre ells, i que cada part interessada jugui un paper clau en el desenvolupament i vida del projecte. Per això és important destacar cada una d'elles.

4.1.1. Director del projecte

En trobar-nos en un Treball Final de Grau, apareix la figura del director. En *Wisebite*, el director és l'Ernest Teniente[12], actualment professor de la Universitat Politècnica de Catalunya. Va ser el primer contacte quant a la inscripció del projecte i és el responsable de guiar a l'autor del projecte durant tot el transcurs d'aquest i supervisar tots els punts que vegi necessaris. Guiarà a l'autor del projecte i facilitarà tots els seus recursos disponibles per a poder construir un bon treball conjuntament amb l'autor d'aquest.

4.1.2. Equip desenvolupador

El grup de desenvolupadors del projecte són un dels actors més importants, per no dir el més important, ja que aporta la capacitat de convertir la idea teòrica a la pràctica. Al parlar d'un treball final de grau, l'equip de desenvolupadors es redueix a una sola persona, l'estudiant i autor d'aquest.

Aquesta persona té com a objectiu iniciar i llançar el projecte endavant, passant per tot el procés d'inscripció de treball. Un cop passada aquesta etapa, l'equip

desenvolupador ha de dissenyar i perfeccionar la idea, definir-la, implementar-la i documentar-la per així poder-la presentar en la fase final del treball.

4.1.3. Establiments de restauració

Com s'ha comentat anteriorment, l'aparició de *Wisebite* té com a objectiu principal convertir i fer evolucionar el món de la restauració. L'aparició de sistemes com aquest aporta un gran canvi a aquest sector. Els responsables de cada un dels establiments disponibles al mercat tindran la possibilitat d'implantar aquest projecte al seu negoci amb l'objectiu de millorar els resultats.

El paper, i la reacció que esdevingui d'aquest col·lectiu, en conèixer l'aparició de *Wisebite* serà de gran importància per definir el futur de la plataforma. Ens podem trobar amb la situació que l'aplicació guanyi gran fama i es faci un bon lloc dins d'aquest sector, o bé oposadament que acabi passant desapercebuda dins de la restauració. Aquest futur es decideix en la reacció d'aquest col·lectiu.

4.1.4. Clients

Si un establiment, sigui bar o restaurant, decideix implantar aquest sistema, no només canviarà la perspectiva de l'empleat sinó també del client o usuari. El comportament que realitzava anteriorment en entrar a aquest establiment haurà de canviar una mica per adaptar-se al nou sistema implantat.

En primera instància, podrà observar com la gestió de comandes dels cambrers ha canviat de forma dràstica i que tota la gestió del restaurant en general ha evolucionat. Per altra banda, com s'ha mencionat en l'abast del projecte, tu com a usuari de *Wisebite* no et fa falta pertànyer a un restaurant determinat en utilitzar l'aplicació, sinó que també pots interactuar amb la plataforma amb el perfil de client, és a dir, cercant els restaurants de la zona, consultar els seus detalls i fins i tot poder realitzar comandes des del seu propi terminal a una taula especificada.

En conseqüència, aquí els clients pertanyents a aquests establiments de restauració són un altre col·lectiu molt important en el transcurs de la història de *Wisebite*.

4.1.5. Competència

Els propietaris i responsables de sistemes similars al d'aquest projecte veuran amenaçada la seva idea i projecció de negoci, propietaris com els de les plataformes i aplicacions comentades en l'estudi de mercat de capítols anteriors.

Aquests altres sistemes poden comportar-se de dues maneres diferents i oposades entre si. Per una banda, podrien aportar millores als seus respectius projectes per així oferir una millor plataforma als usuaris d'aquesta, ja sigui des del perfil treballador com client. O bé per altra banda ignorar-ho i mantenir el seu pla de negoci. Fet que podria fer destacar *Wisebite* com la diferent dins del sector i fer-la important en aquest.

Així doncs, igual que els altres col·lectius, aquest té una importància diferent però igual d'important pel paper que hi juga en l'evolució i futur de la plataforma.

4.2. Requisits funcionals

A *Wisebite*, com en tot projecte, es disposa d'un seguit de requisits funcionals o funcionalitats que defineixen l'ús de la plataforma. El conjunt de tots ells engloben totes les possibilitats de les quals disposa l'usuari en interaccionar amb l'aplicació.

1. **Iniciar sessió**: es permetrà iniciar sessió amb algun dels comptes de *Google* de les quals disposa l'usuari.
2. **Tancar sessió**: es permetrà tancar sessió del sistema.
3. **Veure usuari**: es permetrà veure o consultar els detalls del teu usuari obtenint el nom, el cognom, el correu electrònic, la localització, el nom del restaurant al qual està relacionat (si escau) i el nombre de comandes realitzades (si existeixen), a més de la imatge de perfil.
4. **Editar informació bàsica d'usuari**: es permetrà editar la informació bàsica de l'usuari, englobant el nom, el cognom i la localització.
5. **Canviar imatge de perfil**: es permetrà editar la imatge de perfil permetent pujar una imatge nova emmagatzemada al dispositiu de l'usuari.
6. **Crear restaurant**: es permetrà la creació d'un establiment de restauració indicant tota la informació necessària: nom, localització, descripció, telèfon de contacte, pàgina web, nombre de taules, horaris d'apertura i la carta de plats i menús dels quals disposa el bar o restaurant.
7. **Consultar restaurant**: es permetrà obtenir tota la informació referent al restaurant en qüestió, podent-se així informar sobre l'establiment.
8. **Crear una comanda al teu restaurant**: es permetrà crear una comanda al teu restaurant especificant la taula en la qual es realitza la comanda, i el conjunt de plats i menús que ha especificat el client.
9. **Obtenir les comandes actives**: es permetrà llistar el conjunt de comandes actives dins del teu restaurant, és a dir, totes les comandes les quals encara no han estat cobrades completament, podent veure el percentatge d'elements preparats, entregats i cobrats.
10. **Consultar l'estat d'una comanda**: es permetrà consultar els detalls de la comanda seleccionada. Els detalls constituiran tots els elements que formen la comanda, podent veure si s'ha preparat, entregat i/o cobrat cadascun dels elements.
11. **Cancel·lar una comanda**: es permetrà cancel·lar la comanda, així eliminant tota instància d'aquesta.
12. **Consultar els plats que encara no han estat preparats**: es permetrà obtenir tots els plats que encara no han estat preparats dins del teu restaurant per així poder filtrar-ho pels integrants de la cuina.
13. **Marcar la realització d'un plat**: es permetrà indicar la realització d'un plat per així emmagatzemar-ho al sistema i deixar-ho enregistrat.
14. **Marcar l'entrega d'un plat**: es permetrà indicar l'entrega d'un plat per així emmagatzemar-ho al sistema i deixar-ho enregistrat.

15. **Cobrar una comanda de forma total**: es permetrà cobrar una comanda específica recol·lectant el total de la comanda en qüestió.

16. **Cobrar una comanda de forma fraccionada**: es permetrà cobra una comanda de forma fraccionada, és a dir, seleccionar un subconjunt dels elements de la comanda per així cobrar-ho de forma separada.

17. **Afegir un usuari al restaurant**: es permetrà afegir un usuari a un establiment en específic per així donar-li accés a totes les funcionalitats d'aquest.

18. **Obtenir les estadístiques del teu restaurant**: es permetrà veure les estadístiques del teu establiment de restauració consultant el nombre de comandes, el preu mitjà d'aquestes, el total obtingut, el millor i pitjor plat, el millor i pitjor menú, la millor franja horària, el temps mitjà entre l'inici i el final d'una comanda, la puntuació mitjana i el comptador de valoracions. A més d'anar acompanyat de tres gràfiques que especifiquen el percentatge de plats i menús venuts, i les franges horàries. Totes aquestes dades seran filtrades per dia, setmana o mes.

19. **Canviar la data de l'anàlisi**: es permetrà canviar la data de l'anàlisi i situar-se en un dia desitjat i consultar les estadístiques d'aquell dia, d'aquella setmana i d'aquell mes.

20. **Llistar tots els restaurants**: es permetrà llistar tots els restaurants de la plataforma podent veure els dies d'apertura, el nombre de plats i menús i la valoració mitjana de cadascun d'ells.

21. **Crear una comanda a un restaurant aliè**: es permetrà crear una comanda a un restaurant diferent del de la propietat de l'usuari, en cas que existeixi, especificant tota la informació necessària.

22. **Consultar les valoracions pendents**: es permetrà consultar les valoracions pendents de les quals disposarà l'usuari en qüestió.

23. **Valorar una comanda**: es permetrà valorar una comanda en concret indicant la puntuació dels plats i menús demanats, acompanyats d'un comentari opcional. A més a més, es permetrà valorar de forma global l'opinió sobre el restaurant.

24. **Consultar les valoracions d'un plat o menú**: es permetrà obtenir les valoracions d'un plat o menú per així veure l'opinió d'aquest.

25. **Consultar les valoracions d'un restaurant**: es permetrà obtenir les valoracions d'un restaurant en concret per així veure l'opinió d'aquest.

4.3. Requisits no funcionals

En aquest apartat es realitzarà un repàs de tots els requisits no funcionals[13] o de qualitat dels quals disposa el sistema, és a dir, l'especificació de quelcom sobre el mateix sistema, i de com s'ha de realitzar les accions pertinents a aquest. Per entendreho amb més facilitat s'ha dividit el conjunt de requisits segons el tipus descrit per *Volere*[14].

Requisits d'aparença

- *Tipus de requisit (Volere)*: 10a
- *Descripció*: Disseny atractiu i d'ús senzill que convidarà a l'usuari a fer-ne ús amb més facilitat.
- *Justificació del requisit*: Com l'aplicació treballa sobre un nombre de persones considerable, com és tot el sector de la restauració i els seus respectius clients, cal destacar per l'atractiu de l'aplicació per tal de marcar un abans i un després en l'usuari, és a dir, que gaudeixi de l'experiència a l'hora d'utilitzar *Wisebite*.
- *Condició de satisfacció*: El requisit se satisfarà si s'obté una bona valoració dels usuaris en respecte a l'aparença. Es podrà verificar amb una enquesta de satisfacció al conjunt d'usuaris de l'aplicació.

Requisits d'estil

- *Tipus de requisit (Volere)*: 10b
- *Descripció*: Disseny modern i ambiciós, seguint la tendència en disseny però destacant en punts específics.
- *Justificació del requisit*: La competència del mercat ens obliga a disposar d'un disseny modern per tal destacar sobre la comunitat d'usuaris, i així guanyar usuaris no només per les funcionalitats de la plataforma, sinó per l'estil de l'aplicació.
- *Condició de satisfacció*: El requisit se satisfarà si més de tres quartes parts dels usuaris consideren que *Wisebite* disposa d'un disseny modern, fet que es podrà comprovar amb una enquesta.

Requisits de facilitat d'ús

- *Tipus de requisit (Volere)*: 11a
- *Descripció*: El sistema ha de ser intuïtiu i fàcil d'usar. Complirà els criteris en temes de disseny, de contingut, d'estructura i de presentació fixats pel W3C[15].
- *Justificació del requisit*: Un punt diferenciador important és que l'usuari pugui fer servir el sistema intuïtivament, de manera que no perdi el temps intentant descobrir com funciona i, a més a més, que qualsevol persona sigui capaç de familiaritzar-se amb el sistema. Aquest fet és molt important, ja que la majoria dels usuaris de *Wisebite* no formarà part de la comunitat dels informàtics.
- *Condició de satisfacció*: El requisit se satisfarà si un usuari amb poca experiència en aplicacions aconsegueix usar-lo sense cap problema. Per això, s'utilitzarà un grup de persones inexpert en l'ús d'aplicacions mòbil per veure quina és la seva reacció utilitzant *Wisebite*.

Requisits de latència i velocitat

- *Tipus de requisit (Volere)*: 12a
- *Descripció*: La resposta del sistema ha de ser de menys d'un segon com a mínim en el 95% de les operacions.
- *Justificació del requisit*: Un temps de resposta ràpid permet que l'usuari no perdi el flux o atenció del que està fent amb el sistema. Una plataforma de latència i velocitat dolenta produiria insatisfacció per part de l'usuari.
- *Condició de satisfacció*: El requisit se satisfarà si donat un estudi sobre el rendiment de l'aplicació, aquest confirma que el temps d'espera en cada acció és menor al segon en el 95% dels casos.

Requisits de precisió o exactitud

- *Tipus de requisit (Volere)*: 12c
- *Descripció*: Totes les dates que s'incloguin en l'aplicació tindran el format universal: *DD/MM/AAAA*
- *Justificació del requisit*: És convenient especificar el format de la data, ja que no a tot arreu té el mateix format i podria provocar malentesos i confusions entre els usuaris.
- *Condició de satisfacció*: El requisit se satisfarà si el format de la data i l'hora segueix l'estàndard ISO-8601[16] extens d'estil Europeu (EN 28601).

Requisit de disponibilitat

- *Tipus de requisit (Volere)*: 12d
- *Descripció*: El sistema haurà d'estar disponible les 24 hores del dia durant els 365 dies que conformen l'any.
- *Justificació del requisit*: Els usuaris han de poder utilitzar el sistema en qualsevol moment del dia per tal de poder buscar restaurants o gestionar-los.
- *Condició de satisfacció*: El requisit se satisfarà si el sistema està disponible i completament funcional tot el temps.

Requisits d'adaptabilitat

- *Tipus de requisit (Volere)*: 14c
- *Descripció*: L'aplicació mòbil ha de poder-se veure i executar correctament en els diferents smartphones del mercat, i tenir les mateixes funcionalitats i característiques en tots ells.
- *Justificació del requisit*: L'existència de tants telèfons mòbils diferents i tantes versions d'Android disponibles actualment obliga a garantir que com a mínim es veurà de forma correcta i es podrà executar totes les funcionalitats de la plataforma.
- *Condició de satisfacció*: El requisit se satisfarà si el sistema es pot visualitzar i executar correctament en els principals smartphones del mercat.

Requisit d'immunitat

- *Tipus de requisit (Volere)*: 15e
- *Descripció*: El sistema està protegit d'atacs externs i infeccions per software maliciós.
- *Justificació del requisit*: S'ha de garantir la seguretat per evitar posar en risc la disponibilitat del sistema i la privadesa de les dades dels usuaris. Avui en dia que tot està tan digitalitzat, és un fet molt important per garantir la comoditat de l'usuari a l'hora d'accedir a la plataforma.
- *Condició de satisfacció*: El requisit se satisfarà si s'implementa la normativa de seguretat internacional ISO-17799[17] per tal de garantir la seguretat davant d'atacs externs.

Requisits legals

- *Tipus de requisit (Volere)*: 17a
- *Descripció*: S'aconseguiran tots els drets sobre els serveis externs que s'utilitzin a l'aplicació i a la vegada es compliran les lleis sobre el tractament de dades personals.
- *Justificació del requisit*: Es pactaran acords amb totes les empreses de les quals s'utilitzen els seus serveis, arribant a acords sigui amb la Universitat per poder aprofitar la seva plataforma o amb empreses externes. I també mostrar transparència a l'hora de no compartir dades personals per fins no vinculants al sistema.
- *Condició de satisfacció*: El requisit se satisfarà si no es rep cap denuncia per part de cap servei extern, ni de cap usuari per ús indegut de les dades personals.

4.4. Casos d'ús

Un cop definits i analitzats tots els requisits del sistema, caldrà explicar i especificar els casos d'ús de la plataforma. Per realitzar-ho, s'ha decidit dividir els vint-i-cinc casos d'ús en diferents categories segons el tipus de funcionalitat que vol aconseguir el cas d'ús en concret. El conjunt de casos d'ús es dividirà quatre grups: gestió d'usuaris, gestió de l'establiment, anàlisi de l'establiment i interacció del client.

4.4.1. Gestió d'usuaris

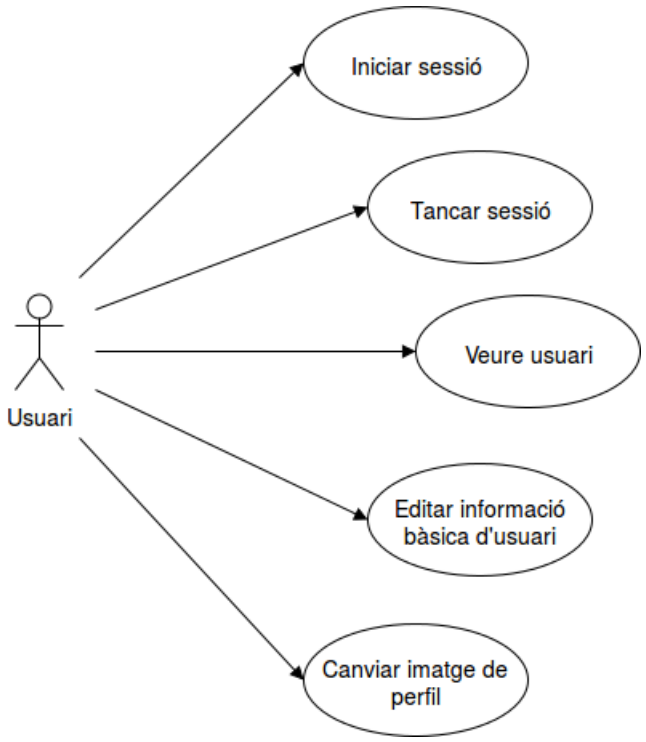

FIGURA 4.1: Diagrama de casos d'ús referent a la gestió d'usuaris

Cas d'ús	#1	Iniciar sessió
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la primera pantalla de l'aplicació.	
Trigger	L'usuari vol iniciar sessió al sistema.	
Escenari principal d'èxit		
1. L'usuari clica al botó d'iniciar sessió. 2. El sistema redirigeix l'usuari al llistat de comptes de Google disponibles al dispositiu. 3. L'usuari selecciona el compte desitjat. 4. El sistema redirigeix a l'usuari a la pantalla principal de l'aplicació, ja amb la sessió iniciada.		

TAULA 4.1: Cas d'ús *Iniciar sessió*

Cas d'ús	#2	Tancar sessió
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol tancar sessió al sistema.	
Escenari principal d'èxit		
1. L'usuari clica sobre la icona de la cantonada esquerra de la barra superior. 2. El sistema mostra a l'usuari un desplegable amb l'opció de tancar sessió. 3. L'usuari clica sobre l'opció de tancar sessió 4. El sistema redirigeix a l'usuari a la primera pantalla de l'aplicació, ja amb la sessió tancada		

TAULA 4.2: Cas d'ús *Tancar sessió*

Cas d'ús	#3	Veure usuari
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol accedir a la seva informació.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la seva imatge situada a la part superior del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla de detall d'usuari.		

TAULA 4.3: Cas d'ús *Veure usuari*

Cas d'ús	#4	Editar informació bàsica d'usuari
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de detall de l'usuari.	
Trigger	L'usuari vol modificar la seva informació bàsica.	
Escenari principal d'èxit		
1. L'usuari clica sobre el botó rodó de la dreta de la part inferior de la pantalla. 2. El sistema redirigeix a l'usuari a la pantalla d'editar usuari. 3. L'usuari modifica algun o alguns dels tres camps que té disponibles: nom, cognom i localització. L'usuari clica sobre el botó superior de la dreta per confirmar els canvis. 4. El sistema emmagatzema els canvis i redirigeix a l'usuari a la pantalla de detall d'usuari.		
Extensions		
2.a L'usuari cancel·la els canvis 2.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 2.a.2 El sistema redirigeix a l'usuari a la pantalla de detall de l'usuari sense cap canvi emmagatzemat.		

TAULA 4.4: Cas d'ús *Editar informació bàsica d'usuari*

Cas d'ús	#5	Canviar imatge de perfil
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de detall de l'usuari.	
Trigger	L'usuari vol modificar la seva imatge de perfil.	
Escenari principal d'èxit		
1. L'usuari clica sobre el botó rodó de la dreta de la part inferior de la pantalla. 2. El sistema redirigeix a l'usuari a la pantalla d'editar usuari. 3. L'usuari clica sobre la imatge d'usuari. 4. El sistema mostra la galeria amb les imatges disponibles dins del dispositiu. 5. L'usuari selecciona la imatge desitjada. 6. El sistema redirigeix a la pantalla d'editar usuari amb la imatge modificada per la seleccionada. 7. L'usuari clica sobre el botó superior de la dreta per confirmar els canvis. 8. El sistema emmagatzema els canvis i redirigeix a l'usuari a la pantalla de detall d'usuari.		
Extensions		
2.a L'usuari cancel·la els canvis 2.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 2.a.2 El sistema redirigeix a l'usuari a la pantalla de detall de l'usuari sense cap canvi emmagatzemat. 6.a L'usuari cancel·la els canvis 6.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 6.a.2 El sistema redirigeix a l'usuari a la pantalla de detall de l'usuari sense cap canvi emmagatzemat.		

TAULA 4.5: Cas d'ús *Canviar imatge de perfil*

4.4.2. Gestió de l'establiment

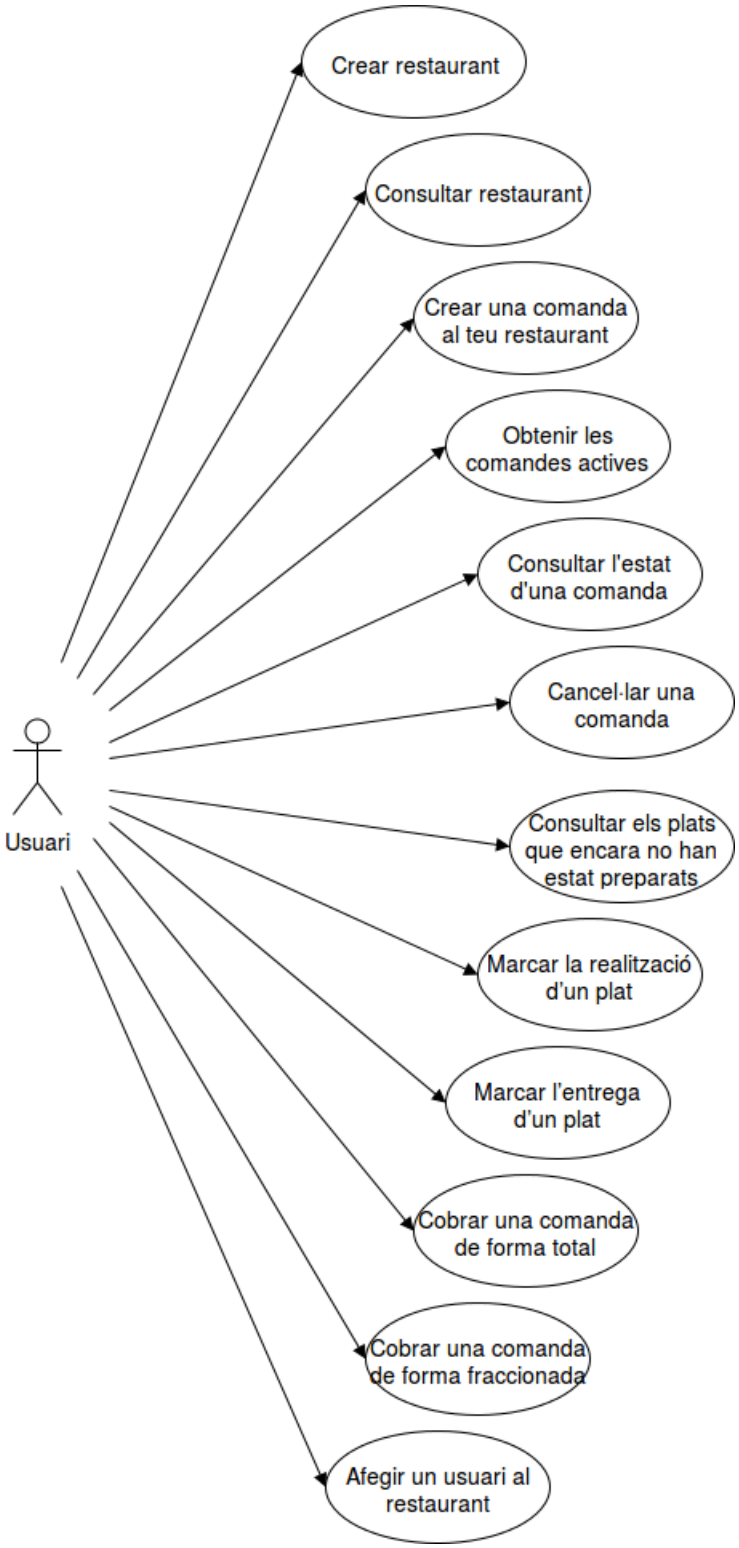

FIGURA 4.2: Diagrama de casos d'ús referent a la gestió de l'establiment

Cas d'ús	#6	Crear restaurant
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol crear un restaurant amb tots els seus detalls i informació.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *Create restaurant* situada a la part superior del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla de creació del restaurant. 5. L'usuari completa tots els camps disponibles (nom, descripció, localització, telèfon de contacte, pàgina web i nombre de taules) i després acciona el botó de següent situat a la part inferior-dreta. 6. El sistema redirigeix a l'usuari a la pantalla de creació d'horaris d'apertura. 7. L'usuari completa l'horari d'apertura i de clausura dels set dies de la setmana, i acciona el botó de següent situat a la part inferior-dreta. 8. El sistema redirigeix a l'usuari a la pantalla de creació de plats i menús. 9. L'usuari crea plats i menús accionant sobre l'opció especifica del menú desplegable de la part inferior-dreta especificant el nom, la descripció i el preu, i el conjunt de plats en cas del menú. Un cop finalitzat, l'usuari acciona el botó de la part superior-dreta per finalitzar la creació del restaurant. 10. El sistema redirigeix a l'usuari a la pantalla principal de l'aplicació, ja amb el restaurant creat i vinculat amb l'usuari actual.		
Extensions		
5.a L'usuari cancel·la els canvis en la informació bàsica 5.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 5.a.2 El sistema redirigeix a l'usuari a la pantalla principal de l'aplicació sense cap canvi emmagatzemat. 7.a L'usuari cancel·la els canvis en l'apertura del restaurant 7.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 7.a.2 El sistema redirigeix a l'usuari a la pantalla d'informació bàsica del restaurant sense cap canvi emmagatzemat. 9.a L'usuari cancel·la els canvis en la creació dels plats i menús 9.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 9.a.2 El sistema redirigeix a l'usuari a la pantalla d'horaris d'apertura del restaurant sense cap canvi emmagatzemat.		

TAULA 4.6: Cas d'ús *Crear restaurant*

Cas d'ús	#7	Consultar restaurant
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol consultar el seu restaurant.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *See restaurant* del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla de vista del restaurant.		

TAULA 4.7: Cas d'ús *Consultar restaurant*

Cas d'ús	#8	Crear una comanda al teu restaurant
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de comandes actives.	
Trigger	L'usuari vol crear una comanda al seu restaurant amb tots els detalls que la caracteritzen.	
Escenari principal d'èxit		
1. L'usuari clica sobre el botó rodó de la part de la part inferior de la pantalla. 2. El sistema mostra un desplegable amb un petit formulari per introduir la taula a on es realitza la comanda. 3. L'usuari escriu el número de taula a on es realitza la comanda i prem *Acceptar*. 4. El sistema redirigeix a l'usuari a la pantalla de creació de comanda. 5. L'usuari selecciona els plats i menús que desitja el client prement el botó de + de cada un dels ítems de la carta. En cas que estiguem parlant d'un menú, s'haurà de seleccionar quins plats es vol del menú determinat. Un cop tot completat, l'usuari clica sobre el botó de la part superior-dreta per finalitzar la creació de la comanda. 6. El sistema redirigeix a l'usuari a la pantalla de comandes actives amb la nova comanda emmagatzemada.		
Extensions		
5.a L'usuari cancel·la els canvis 5.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 5.a.2 El sistema redirigeix a l'usuari a la pantalla de comandes actives sense cap canvi emmagatzemat.		

TAULA 4.8: Cas d'ús *Crear una comanda al teu restaurant*

Cas d'ús	#9	Obtenir les comandes actives
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol consultar les comandes actives.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *Active orders* del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla de comandes actives del restaurant.		

TAULA 4.9: Cas d'ús *Obtenir les comandes actives*

Cas d'ús	#10	Consultar l'estat d'una comanda
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de comandes actives.	
Trigger	L'usuari vol consultar l'estat d'una comanda.	
Escenari principal d'èxit		
1. L'usuari clica sobre alguna de les comandes que disposa en el llistat. 2. El sistema redirigeix a l'usuari a la vista de detall de la comanda.		

TAULA 4.10: Cas d'ús *Consultar l'estat d'una comanda*

Cas d'ús	#11	Cancel·lar una comanda
Actor principal	L'usuari ha accedit a la pantalla de detall de la comanda.	
Trigger	L'usuari vol cancel·lar una comanda.	
Escenari principal d'èxit		
1. L'usuari clica sobre la icona de la cantonada esquerra de la barra superior. 2. El sistema mostra a l'usuari un desplegable amb l'opció de cancel·lar la comanda. 3. L'usuari clica sobre l'opció de cancel·lar comanda 4. El sistema redirigeix a l'usuari a la pantalla principal de l'aplicació, ja amb la comanda cancel·lada		

TAULA 4.11: Cas d'ús *Cancel·lar una comanda*

Cas d'ús	#12	Consultar els plats que encara no han estat preparats
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol consultar els plats que encara no han estat preparats.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *Kitchen* del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla de cuina.		

TAULA 4.12: Cas d'ús *Consultar els plats que encara no han estat preparats*

Cas d'ús	#13	Marcar la realització d'un plat
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de cuina.	
Trigger	L'usuari vol marcar la realització d'un plat no preparat.	
Escenari principal d'èxit		
1. L'usuari selecciona l'ítem que ja ha preparat i confirma la seva elecció prement *Acceptar*. 2. El sistema emmagatzema el canvi i elimina l'ítem del llistat.		

TAULA 4.13: Cas d'ús *Marcar la realització d'un plat*

Cas d'ús	#14	Marcar l'entrega d'un plat
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de detall de la comanda.	
Trigger	L'usuari vol marcar l'entrega d'un plat.	
Escenari principal d'èxit		
1. L'usuari selecciona el botó rodó vinculat amb l'ítem desitjat i que vol marcar com a entregat. 2. El sistema emmagatzema el canvi.		

TAULA 4.14: Cas d'ús *Marcar l'entrega d'un plat*

Cas d'ús	#15	Cobrar una comanda de forma total
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de detall de la comanda.	
Trigger	L'usuari vol cobrar una comanda de forma total.	
Escenari principal d'èxit		
1. L'usuari clica al botó rodó de la part inferior-dreta de la pantalla. 2. El sistema mostra un diàleg en el qual et permet elegir fer el cobrament parcialment o no. 3. L'usuari clica sobre *All*. 4. El sistema tanca el diàleg i en mostra un altre confirmant la recol·lecta del total de la comanda. 5. L'usuari clica sobre *Yes*. 6. El sistema tanca el diàleg i emmagatzema les dades marcant com a cobrada la comanda corresponent.		

TAULA 4.15: Cas d'ús *Cobrar una comanda de forma total*

Cas d'ús	#16	Cobrar una comanda de forma fraccionada
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de detall de la comanda.	
Trigger	L'usuari vol cobrar una comanda de forma fraccionada.	
Escenari principal d'èxit		
1. L'usuari clica al botó rodó de la part inferior-dreta de la pantalla. 2. El sistema mostra un diàleg en el qual et permet elegir fer el cobrament parcialment o no. 3. L'usuari clica sobre *In groups*. 4. El sistema tanca el diàleg i mostra tots els ítems que encara no han estat cobrats de la comanda. 5. L'usuari selecciona els ítems que vol cobrar i clica sobre el botó rodó de la part inferior de la pantalla. 6. El sistema mostra un diàleg confirmant la recol·lecta dels elements seleccionats dins la comanda. 7. L'usuari clica sobre *Yes*. 8. El sistema tanca el diàleg i emmagatzema les dades marcant com a cobrats els plats corresponents.		

TAULA 4.16: Cas d'ús *Cobrar una comanda de forma fraccionada*

Cas d'ús	#17	Afegir un usuari al restaurant
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de detall del restaurant.	
Trigger	L'usuari vol afegir un usuari dins de l'equip del restaurant.	
Escenari principal d'èxit		
1. L'usuari clica sobre el botó rodó de la barra superior. 2. El sistema mostra un diàleg amb un petit formulari per introduir l'adreça de correu electrònic de l'usuari a afegir. 3. L'usuari escriu l'adreça de correu electrònic de l'usuari a afegir. 4. El sistema emmagatzema la informació i tanca el diàleg.		
Extensions		
3.a L'usuari no existeix 3.a.1 L'usuari introdueix una adreça electrònica no registrada a la base de dades de la plataforma. 3.a.2 El sistema reporta la incidència a l'usuari i tanca el diàleg sense cap canvi en la persistència de la plataforma. 3.b L'usuari ja pertany a un restaurant 3.b.1 L'usuari introdueix una adreça electrònica pertanyent a un usuari que ja està vinculat a un restaurant. 3.b.2 El sistema reporta la incidència a l'usuari i tanca el diàleg sense cap canvi en la persistència de la plataforma.		

TAULA 4.17: Cas d'ús *Afegir un usuari al restaurant*

4.4.3. Anàlisi de l'establiment

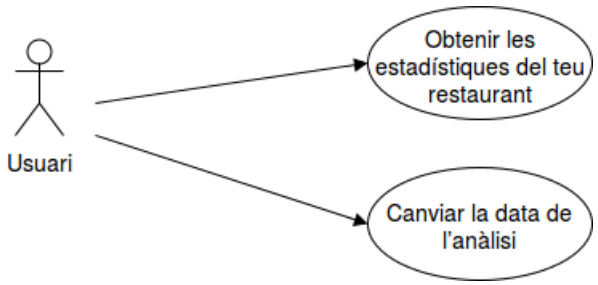

FIGURA 4.3: Diagrama de casos d'ús referent a l'anàlisi de l'establiment

Cas d'ús	#18	Obtenir les estadístiques del teu restaurant
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol consultar les estadístiques del seu restaurant.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *Analytics* del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla d'estadístiques.		

TAULA 4.18: Cas d'ús *Obtenir les estadístiques del teu restaurant*

Cas d'ús	#19	Canviar la data de l'anàlisi
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla d'estadístiques.	
Trigger	L'usuari vol canviar la data de visualització per analitzar les estadístiques d'un altre període.	
Escenari principal d'èxit		
1. L'usuari clica sobre el botó amb icona de calendari situat a la barra superior. 2. El sistema mostra un diàleg amb un calendari per seleccionar una data. 3. L'usuari selecciona un dia del calendari i prem *Acceptar*. 4. El sistema refresca la vista amb la nova data seleccionada i tanca el diàleg.		

TAULA 4.19: Cas d'ús *Canviar la data de l'anàlisi*

4.4.4. Interacció del client

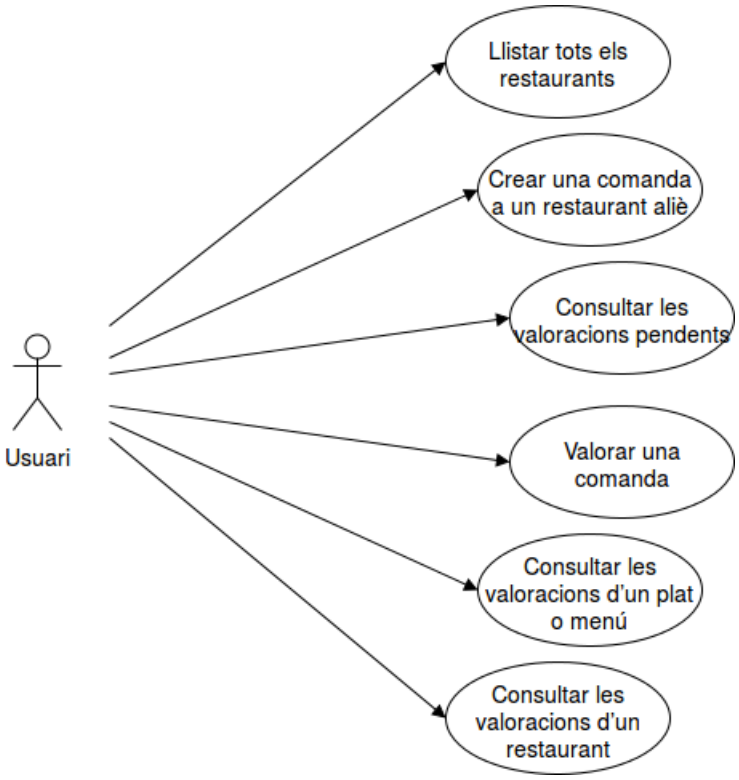

FIGURA 4.4: Diagrama de casos d'ús referent a la interacció del client

Cas d'ús	#20	Llistar tots els restaurants
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol consultar els restaurants emmagatzemats dins la plataforma.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *List restaurants* del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla referent al llistat de restaurants.		

TAULA 4.20: Cas d'ús *Llistar tots els restaurants*

Cas d'ús	#21	Crear una comanda a un restaurant aliè
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla referent al llistat dels restaurants.	
Trigger	L'usuari vol crear una comanda a un restaurant aliè.	
Escenari principal d'èxit		
1. L'usuari selecciona algun dels restaurants del llistat. 2. El sistema redirigeix a l'usuari a la pantalla de vista d'un restaurant. 3. L'usuari clica sobre el botó rodó de la barra superior. 4. El sistema mostra un desplegable amb un petit formulari per introduir la taula a on es realitza la comanda. 5. L'usuari escriu el número de taula a on es realitza la comanda i prem *Acceptar*. 6. El sistema redirigeix a l'usuari a la pantalla de creació de comanda. 7. L'usuari selecciona els plats i menús que desitja prement el botó de + de cada un dels ítems de la carta. En cas que estiguem parlant d'un menú, s'haurà de seleccionar quins plats es vol del menú determinat. Un cop tot completat, l'usuari clica sobre el botó de la part superior-dreta per finalitzar la creació de la comanda. 8. El sistema redirigeix a l'usuari a la pantalla de visualització del restaurant amb la nova comanda emmagatzemada.		
Extensions		
7.a L'usuari cancel·la els canvis 7.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 7.a.2 El sistema redirigeix a l'usuari a la pantalla de visualització del restaurant sense cap canvi emmagatzemat.		

TAULA 4.21: Cas d'ús *Crear una comanda a un restaurant aliè*

Cas d'ús	#22	Consultar les valoracions pendents
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla principal de l'aplicació.	
Trigger	L'usuari vol consultar les seves valoracions pendents.	
Escenari principal d'èxit		
1. L'usuari clica sobre el *Navigational drawer* amb l'objectiu de fer-lo desplegar. 2. El sistema desplega el menú lateral. 3. L'usuari clica sobre la pestanya anomenada *Pending reviews* del menú lateral. 4. El sistema redirigeix a l'usuari a la pantalla de valoracions pendents.		

TAULA 4.22: Cas d'ús *Consultar les valoracions pendents*

Cas d'ús	#23	Valorar una comanda
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla de valoracions pendents.	
Trigger	L'usuari vol valorar una de les comandes que té pendents per valorar.	
Escenari principal d'èxit		
1. Selecciona una de les comandes que té per valorar. 2. El sistema redirigeix a l'usuari a la vista de valoració. 3. L'usuari puntua amb una puntuació fins a 5 estrelles cada un dels plats que formen la comanda i afegeix un comentari opcional que complementa la valoració. A més a més, també valora el restaurant de forma global amb una puntuació i un comentari opcional. Un cop finalitzat acciona el botó de la part inferior de la pantalla. 4. El sistema emmagatzema les dades introduïdes i redirigeix a l'usuari a la pantalla de valoracions pendents.		
Extensions		
3.a L'usuari cancel·la els canvis 3.a.1 L'usuari clica sobre la marxa enrere de la barra superior. 3.a.2 El sistema redirigeix a l'usuari a la pantalla de valoracions pendents sense cap canvi emmagatzemat.		

TAULA 4.23: Cas d'ús *Valorar una comanda*

Cas d'ús	#24	Consultar les valoracions d'un plat o menú
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla referent al llistat dels restaurants.	
Trigger	L'usuari vol consultar les valoracions d'un plat o menú d'un restaurant aliè.	
Escenari principal d'èxit		
1. L'usuari selecciona algun dels restaurants del llistat. 2. El sistema redirigeix a l'usuari a la pantalla de vista d'un restaurant. 3. L'usuari manté clicat sobre un dels plats o menús del restaurant. 4. El sistema mostra un diàleg amb estadístiques i informació bàsica sobre l'ítem seleccionat. 5. L'usuari clica sobre el botó del diàleg amb títol *Reviews*. 6. El sistema redirigeix a l'usuari a la pantalla de valoracions de l'ítem seleccionat.		

TAULA 4.24: Cas d'ús *Consultar les valoracions d'un plat o menú.*

Cas d'ús	#25	Consultar les valoracions d'un restaurant
Actor principal	Usuari	
Precondició	L'usuari ha accedit a la pantalla referent al llistat dels restaurants.	
Trigger	L'usuari vol consultar les valoracions d'un restaurant.	
Escenari principal d'èxit		
1. L'usuari selecciona algun dels restaurants del llistat. 2. El sistema redirigeix a l'usuari a la pantalla de vista d'un restaurant. 3. L'usuari clica sobre la icona de la cantonada esquerra de la barra superior. 4. El sistema mostra a l'usuari un desplegable amb l'opció de consultar les valoracions sobre el restaurant. 5. L'usuari clica sobre l'opció de consultar les valoracions sobre el restaurant. 6. El sistema redirigeix a l'usuari a la pantalla de valoracions del restaurant seleccionat.		

TAULA 4.25: Cas d'ús *Consultar les valoracions d'un restaurant*

Capítol 5

Especificació

Després d'haver explicat l'origen de *Wisebite*, haver estudiat les solucions actuals del mercat i suggerit una de millor i haver analitzat els requisits i les funcionalitats del sistema, cal especificar els models que representaran les entitats que formaran el projecte. En primera instància, es mostrarà l'esquema conceptual que defineix el treball i s'explicarà cada una de les entitats que l'engloben. Per altra banda, s'analitzarà l'esquema de comportament que interacciona entre l'usuari i el sistema.

5.1. Esquema conceptual

L'esquema conceptual d'un sistema és la representació gràfica dels models que caracteritzen aquest. En Enginyeria del Software això és conegut com un diagrama de classes[18]. Primerament, s'explicarà textualment les entitats que participen en l'esquema conceptual i posteriorment es mostrà el diagrama de classes representatiu.

5.1.1. Descripció de les classes

L'esquema conceptual del projecte *Wisebite* disposa de nou classes que s'explicaran a continuació.

- **User**: Un usuari és l'entitat que representa a les persones que utilitzaran l'aplicació i les seves funcionalitats. Un usuari té un *id*, que l'identifica, un *correu electrònic* (email), un *nom* (name), un *cognom* (lastName) i una *localització* (location). Cada usuari pot o no tenir una imatge de perfil, treballa o no a un restaurant, pot crear moltes comandes, pot disposar d'un seguit de comandes a valorar i pot tenir d'un conjunt de valoracions realitzades emmagatzemades dins la plataforma.
- **Image**: Una imatge és l'entitat que representa una imatge emmagatzemada al sistema. Una imatge té un *id*, que l'identifica, una *ruta* per localitzar-la quan sigui necessari (imageFile) i una *descripció* (description). Cada imatge pot estar relacionada o amb un usuari o bé amb un restaurant.
- **Restaurant**: Un restaurant és l'entitat que representa a l'establiment de restauració que engloba l'objectiu de la plataforma. Un restaurant té un *id*, que l'identifica, un *nom* (name), un *telèfon* de contacte (phone), una *descripció* (description), una *localització* (location), un *nombre de taules* (numberOfTables) i la direcció de la *pàgina web* (website). Cada restaurant pot disposar de fins a set horaris d'apertura, pot tenir moltes imatges associades, molts usuaris formant l'equip de treball de l'establiment, pot tenir un conjunt de comandes externes, un conjunt de plats i menús que composen la carta de l'establiment i un grup d'avaluacions realitzades pels usuaris de la plataforma.

- **OpenTime**: Un horari d'apertura és l'entitat que representa l'horari d'inici i final d'una franja horària. Un horari d'apertura té un *id*, que l'identifica, un *horari d'inici* (startDate) i un *horari fi* (endDate). Cada horari d'apertura està relacionat amb un restaurant.

- **Dish**: Un plat és l'entitat que representa al plat de forma individual. Un plat té un *id*, que l'identifica, un *nom* (name), una *descripció* (description) i un *preu* (price). Cada plat forma part d'un restaurant, pot ser part d'un menú especific jugant el paper de plat principal, secundari o alternatiu, pot tenir un conjunt de valoracions dels usuaris de l'aplicació i forma part d'un seguit de línies de comanda.

- **Menu**: Un menu és l'entitat que representa a un menú de la carta de l'establiment. Un menú té un *id*, que l'identifica,un *nom* (name), una *descripció* (description) i un *preu* (price). Cada menú forma part d'un restaurant i pot tenir un conjunt de valoracions dels usuaris de l'aplicació.

- **Order**: Una comanda és l'entitat que representa la petició d'un client dins d'un establiment de restauració. Una comanda té un *id*, que l'identifica, una *data* d'inici (date), un *número de taula* (tableNumber) i una *data d'última modificació* (lastDate). Cada comanda es creada per un dels usuaris de la plataforma, pot estar en la llista de comandes a valorar per part d'un usuari, pot ser part del conjunt de comandes externes de l'aplicació i conté un conjunt de línies de comanda.

- **OrderItem**: Una línia de comanda és l'entitat que representa un dels elements del llistat que forma una comanda. Una línia de comanda té un *id*, que l'identifica, pot estar o no *preparat* (ready), pot estar o no *estregat* (delivered), pot estar o no *pagat* (paid) i té una *característica diferent* (differentFeature). Cada línia de comanda forma part d'una comanda i té un plat dins del restaurant associat.

- **Review**: Una valoració és l'entitat que representa l'opinió d'un usuari sobre un plat, un menú o un restaurant. Una valoració té un *id*, que l'identifica, una *puntuació* (points), un *comentari* (comment) i una *data* de realització (date). Cada valoració està sempre relacionada o bé amb un plat, o amb un menú o amb un restaurant i ha estat realitzada per un usuari.

5.1.2. Diagrama de classes

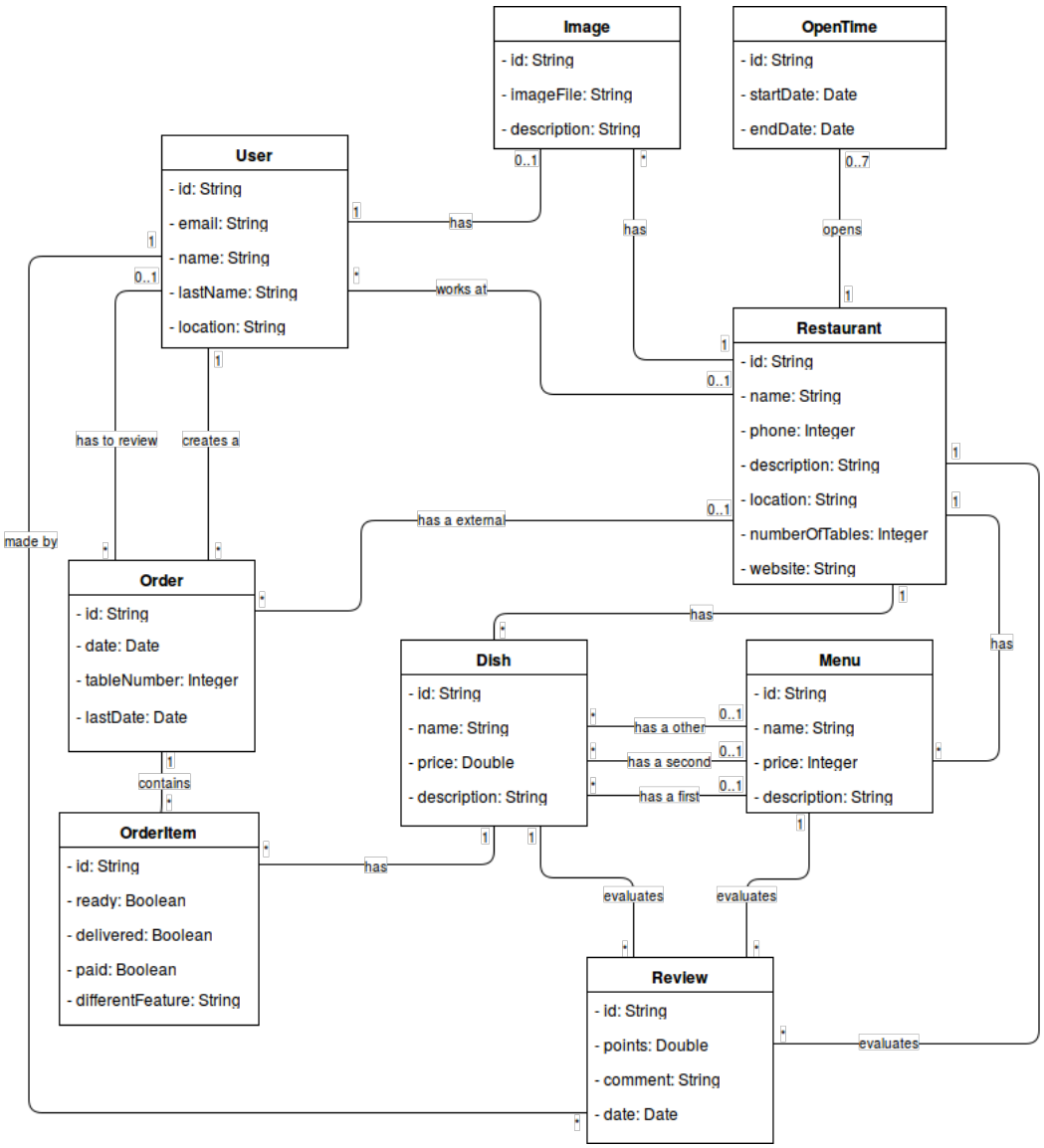

FIGURA 5.1: Diagrama de classes

5.2. Esquema del comportament

Un cop analitzades les classes del model, s'ha de proposar tots els esdeveniments que interaccionen entre l'usuari i el sistema, en els quals es mostrarà inicialment el diagrama de seqüència i el contracte corresponent.

Cas d'ús #1

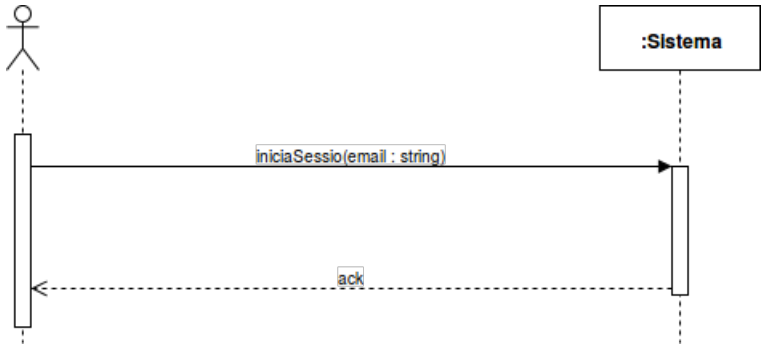

FIGURA 5.2: Esquema de comportament del cas d'ús #1

Context	Sistema :: iniciaSessio(email : string)
Precondició	La connexió amb el sistema d'autentificació de Google ha funcionat correctament i el correu electrònic *email* és vàlid.
Postcondició	L'usuari inicia sessió en cas que ja estigues registrat a la plataforma i es crea en cas que no existís.

Cas d'ús #2

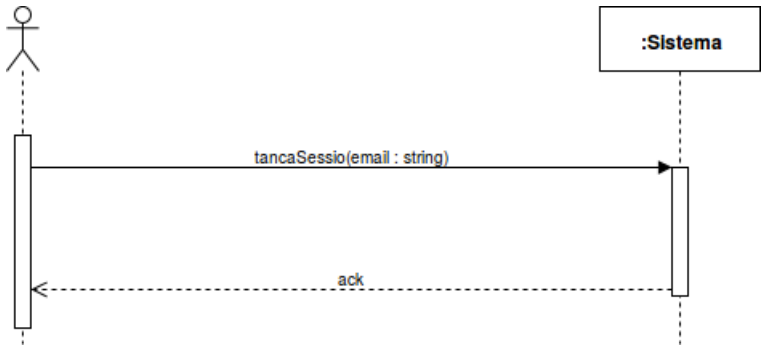

FIGURA 5.3: Esquema de comportament del cas d'ús #2

Context	Sistema :: tancaSessio(email : string)
Precondició	La connexió amb el sistema d'autentificació de Google ha funcionat correctament i el correu electrònic *email* existeix a la base de dades.
Postcondició	L'usuari tanca sessió satisfactòriament i deixa de tenir accés a les funcionalitats de la plataforma.

Cas d'ús #3

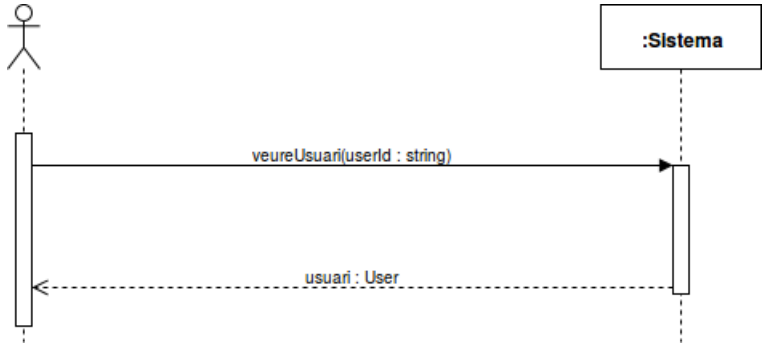

FIGURA 5.4: Esquema de comportament del cas d'ús #3

Context	Sistema :: veureUsuari(userId : string)
Precondició	L'usuari amb id *userId* existeix dins la base de dades.
Postcondició	L'aplicació mostra tota la informació relacionada amb l'usuari amb id *userId*.

Cas d'ús #4

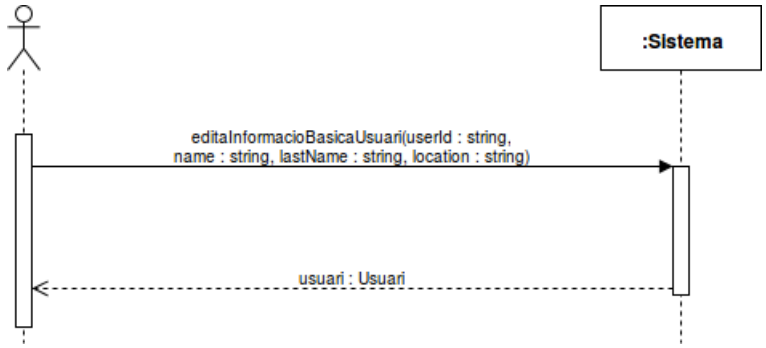

FIGURA 5.5: Esquema de comportament del cas d'ús #4

Context	Sistema :: editaInformacioBasicaUsuari(userId : string, name : string, lastName : string, location : string)
Precondició	L'usuari amb id *userId* existeix dins la base de dades.
Postcondició	L'aplicació enregistra els canvis i mostra l'usuari modificat.

Cas d'ús #5

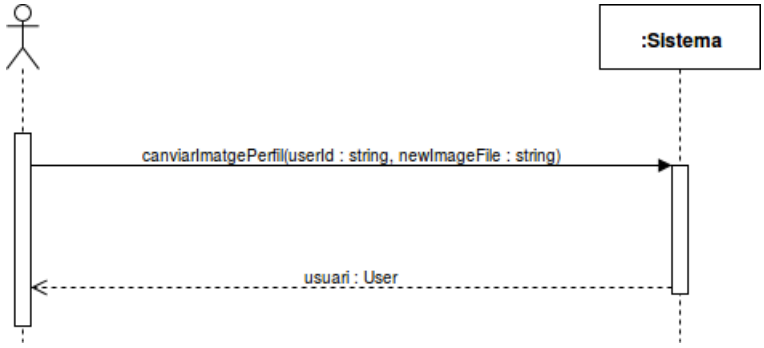

FIGURA 5.6: Esquema de comportament del cas d'ús #5

Context	Sistema :: canviarImatgePerfil(userId : string, newImageFile : string)
Precondició	L'usuari amb id *userId* existeix dins la base de dades.
Postcondició	L'aplicació emmagatzema la imatge, l'associa a l'usuari i el mostra modificat.

Cas d'ús #6

FIGURA 5.7: Esquema de comportament del cas d'ús #6

Context	Sistema :: crearRestaurant(name : string, description : string, location : string, phone : string, website : string, numberOfTables : integer, openTimes : Set(TupleType(startTime : date, endTime : date)), dishes : Set(TupleType(name : string, description : string, price : double)), menus : Set(TupleType(name : string, description : string, price : double, dishes : Set(TupleType(name : string, description : string, price : double)))))
Precondició	-
Postcondició	L'aplicació emmagatzema el restaurant amb els paràmetres indicats i l'associa a l'usuari creador.

Cas d'ús #7

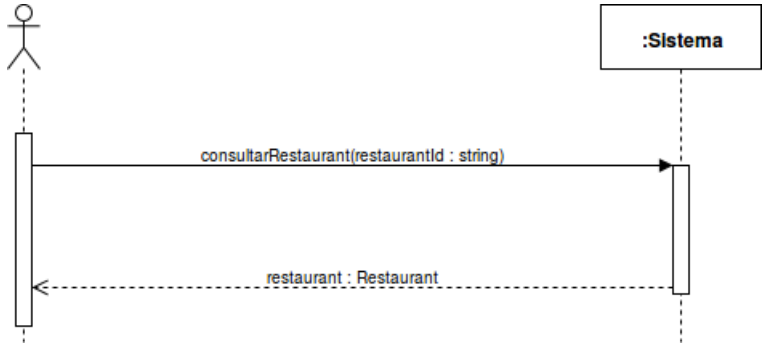

FIGURA 5.8: Esquema de comportament del cas d'ús #7

Context	Sistema :: consultarRestaurant(restaurantId : string)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades.
Postcondició	L'aplicació mostra tota la informació relacionada amb el restaurant amb id *restaurantId*.

Cas d'ús #8

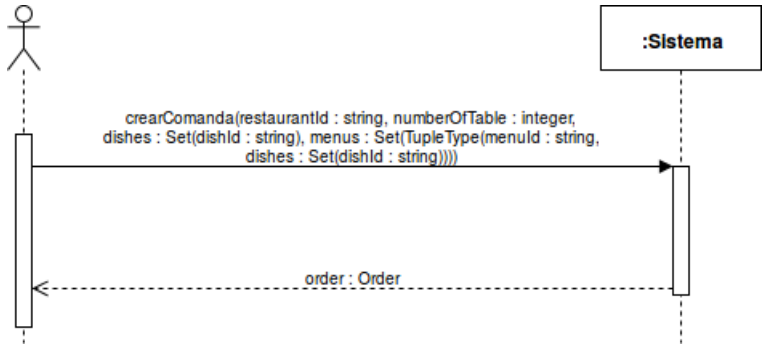

FIGURA 5.9: Esquema de comportament del cas d'ús #8

Context	Sistema :: crearComanda(restaurantId : string, numberOfTable : integer, dishes : Set(dishId : string), menus : Set(TupleType(menuId : string, dishes : Set(dishId : string))))
Precondició	El restaurant amb id *restaurantId*, els plats amb id *dishId* i els menus amb id *menuId* existeixen dins la base de dades.
Postcondició	L'aplicació emmagatzema la informació indicada als paràmetres i mostra la comanda creada.

Cas d'ús #9

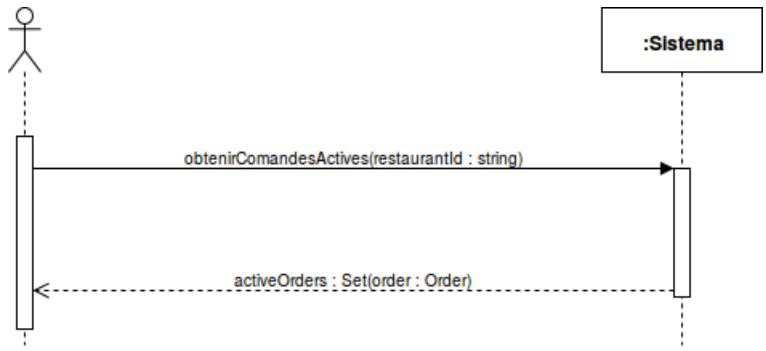

FIGURA 5.10: Esquema de comportament del cas d'ús #9

Context	Sistema :: obtenirComandesActives(restaurantId : string)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades.
Postcondició	L'aplicació mostra un llistat amb les comandes actives, és a dir, les que encara no han estat finalitzades.

Cas d'ús #10

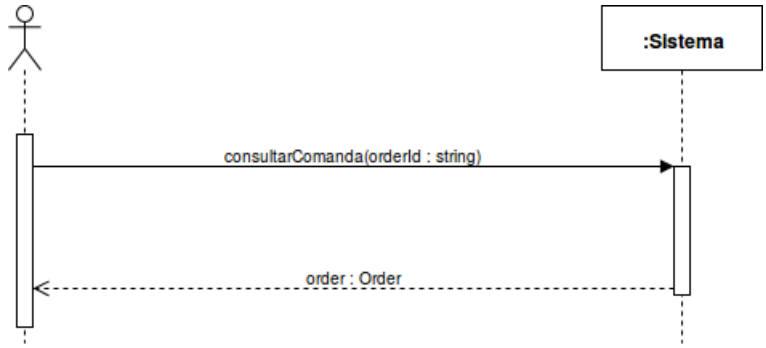

FIGURA 5.11: Esquema de comportament del cas d'ús #10

Context	Sistema :: consultarComanda(orderId : string)
Precondició	La comanda amb id *orderId* existeix dins la base de dades.
Postcondició	L'aplicació mostra tota la informació relacionada amb la comanda amb id *orderId*.

Cas d'ús #11

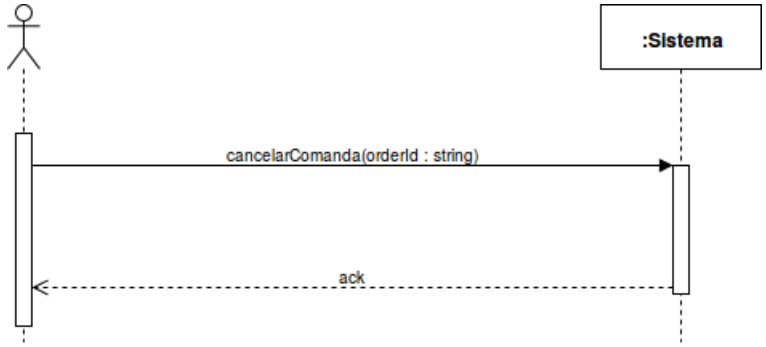

FIGURA 5.12: Esquema de comportament del cas d'ús #11

Context	Sistema :: cancelarComanda(orderId : string)
Precondició	La comanda amb id *orderId* existeix dins la base de dades.
Postcondició	L'aplicació cancel·la la comanda eliminant tota instància relacionada amb ella.

Cas d'ús #12

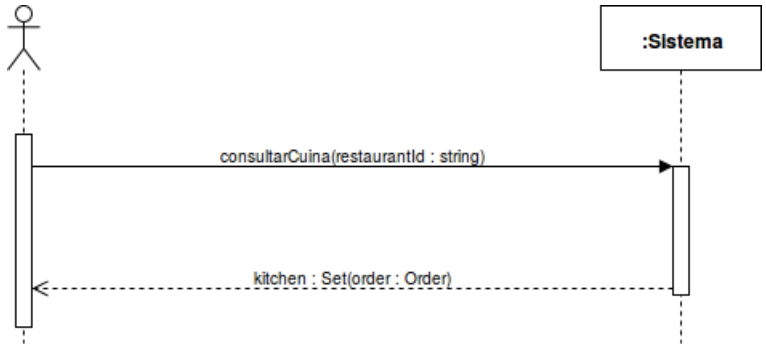

FIGURA 5.13: Esquema de comportament del cas d'ús #12

Context	Sistema :: consultarCuina(restaurantId : string)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades.
Postcondició	L'aplicació mostra totes les comandes les quals encara existeix algun ítem que no ha estat preparat.

Cas d'ús #13

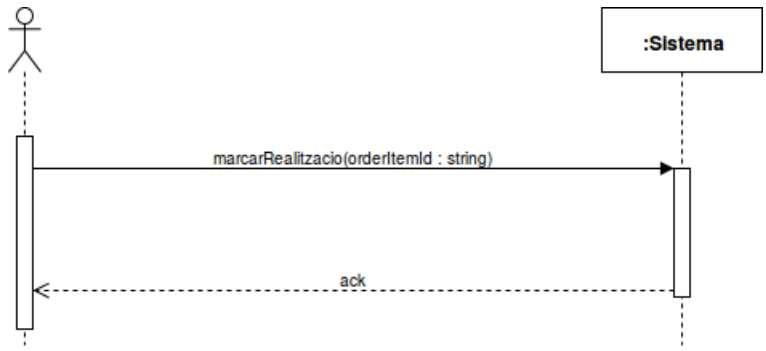

FIGURA 5.14: Esquema de comportament del cas d'ús #13

Context	Sistema :: marcarRealitzacio(orderItemId : string)
Precondició	La línia de comanda amb id *orderItemId* existeix dins la base de dades.
Postcondició	L'aplicació marca l'ítem com a realitzat i mostra l'actualització.

Cas d'ús #14

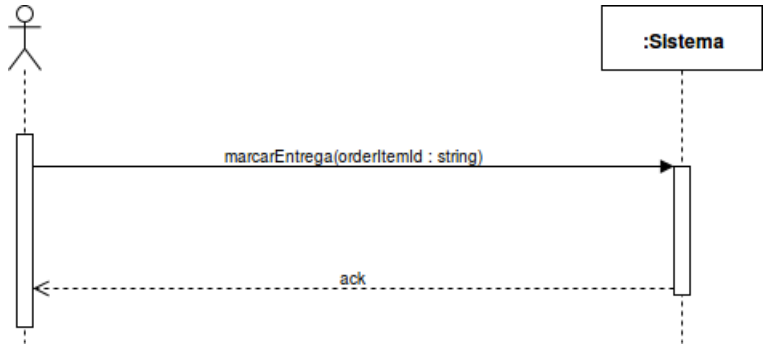

FIGURA 5.15: Esquema de comportament del cas d'ús #14

Context	Sistema :: marcarEntrega(orderItemId : string)
Precondició	La línia de comanda amb id *orderItemId* existeix dins la base de dades.
Postcondició	L'aplicació marca l'ítem com a entregat i mostra l'actualització.

Cas d'ús #15

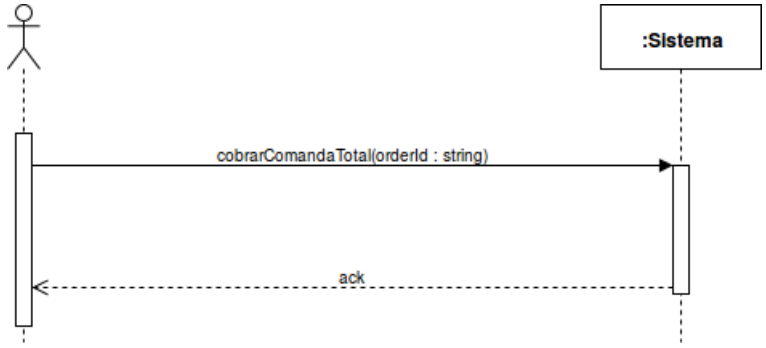

FIGURA 5.16: Esquema de comportament del cas d'ús #15

Context	Sistema :: cobrarComandaTotal(orderId : string)
Precondició	La comanda amb id *orderId* existeix dins la base de dades.
Postcondició	L'aplicació indica per tot ítem de la comanda el seu nou estat de cobrat i mostra l'actualització.

Cas d'ús #16

FIGURA 5.17: Esquema de comportament del cas d'ús #16

Context	Sistema :: cobrarComandaFraccionada(orderId : string, orderItems : Set(orderItemId : string))
Precondició	La comanda amb id *orderId* i les línies de comanda amb id *orderItemId* existeixen dins la base de dades.
Postcondició	L'aplicació marca com a pagades totes les línies de comanda seleccionades i mostra l'actualització.

Cas d'ús #17

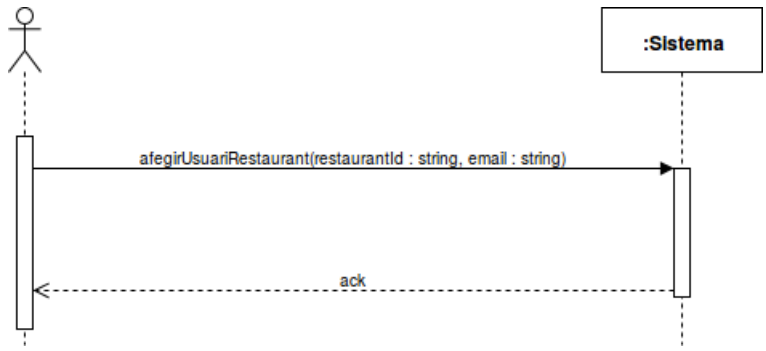

FIGURA 5.18: Esquema de comportament del cas d'ús #17

Context	Sistema :: afegirUsuariRestaurant(restaurantId : string, email : string)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades i el correu electrònic *email* és vàlid.
Postcondició	L'aplicació afegeix l'usuari amb correu electrònic *email* dins el restaurant especificat en cas que no formi part de cap altre restaurant i existeix a la base de dades, altrament ho notifica i no realitza cap canvi.

Cas d'ús #18

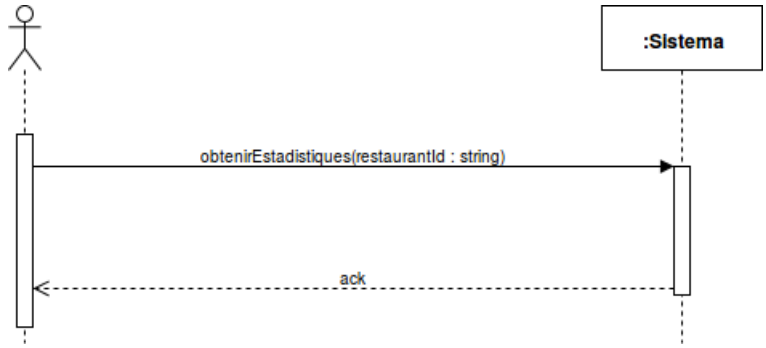

FIGURA 5.19: Esquema de comportament del cas d'ús #18

Context	Sistema :: obtenirEstadistiques(restaurantId : string)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades.
Postcondició	L'aplicació mostra tota la informació estadística relacionada amb el restaurant.

Cas d'ús #19

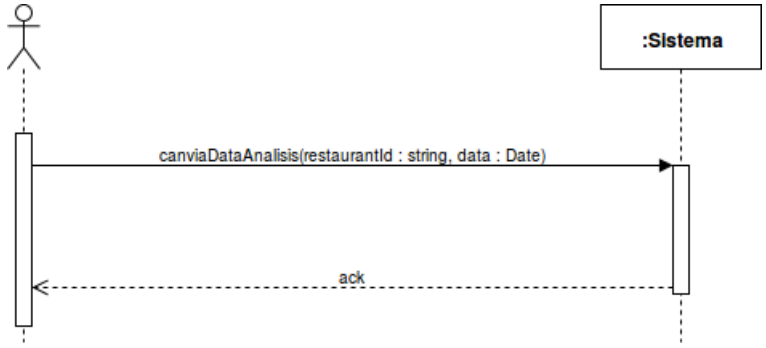

FIGURA 5.20: Esquema de comportament del cas d'ús #19

Context	Sistema :: canviaDataAnalisis(restaurantId : string, data : Date)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades i la data *date* és vàlida.
Postcondició	L'aplicació calcula de nou les estadístiques amb la nova data *date* i mostra l'actualització.

Cas d'ús #20

FIGURA 5.21: Esquema de comportament del cas d'ús #20

Context	Sistema :: llistarRestaurants()
Precondició	-
Postcondició	L'aplicació mostra un llistat amb tots els restaurant emmagatzemats dins l'aplicació.

Cas d'ús #21

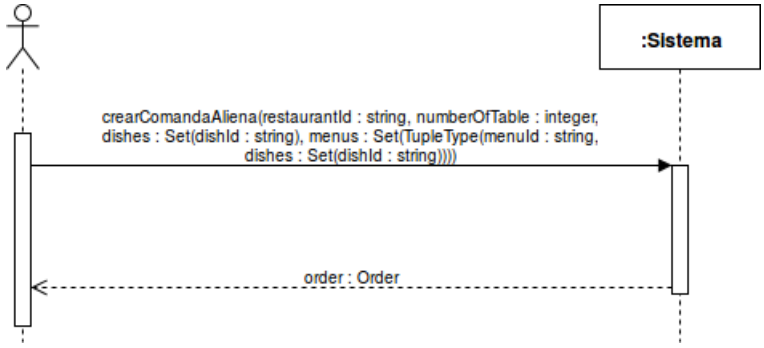

FIGURA 5.22: Esquema de comportament del cas d'ús #21

Context	Sistema :: crearComandaAliena(restaurantId : string, numberOfTable : integer, dishes : Set(dishId : string), menus : Set(TupleType(menuId : string, dishes : Set(dishId : string))))
Precondició	El restaurant amb id *restaurantId*, els plats amb id *dishId* i els menus amb id *menuId* existeixen dins la base de dades.
Postcondició	L'aplicació emmagatzema la informació indicada als paràmetres, indica la comanda com a externa dins al restaurant i mostra la comanda creada.

Cas d'ús #22

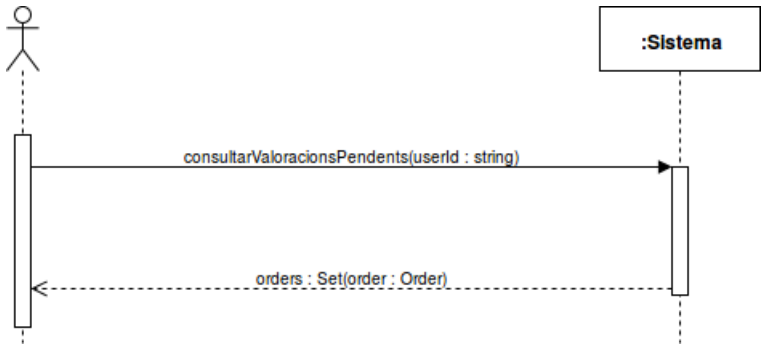

FIGURA 5.23: Esquema de comportament del cas d'ús #22

Context	Sistema :: consultarValoracionsPendents(userId : string)
Precondició	L'usuari amb id *userId* existeix dins la base de dades.
Postcondició	L'aplicació mostra el conjunt de comandes que estan pendents de valorar.

Cas d'ús #23

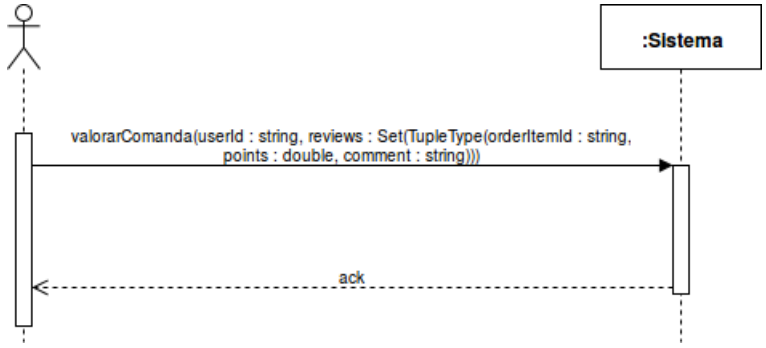

FIGURA 5.24: Esquema de comportament del cas d'ús #23

Context	Sistema :: valorarComanda(userId : string reviews : Set(TupleType(orderItemId : string, points : double, comment : string)))
Precondició	L'usuari amb id *userId* existeix dins la base de dades.
Postcondició	L'aplicació emmagatzema la valoració i ho notifica a l'usuari.

Cas d'ús #24

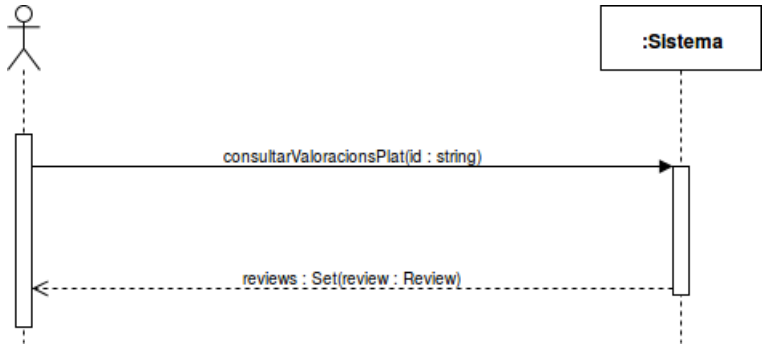

FIGURA 5.25: Esquema de comportament del cas d'ús #24

Context	Sistema :: consultarValoracionsPlat(id : string)
Precondició	El plat o el menú amb l'identificador *id* existeix dins la base de dades.
Postcondició	L'aplicació mostra les valoracions de l'element especificat.

Cas d'ús #25

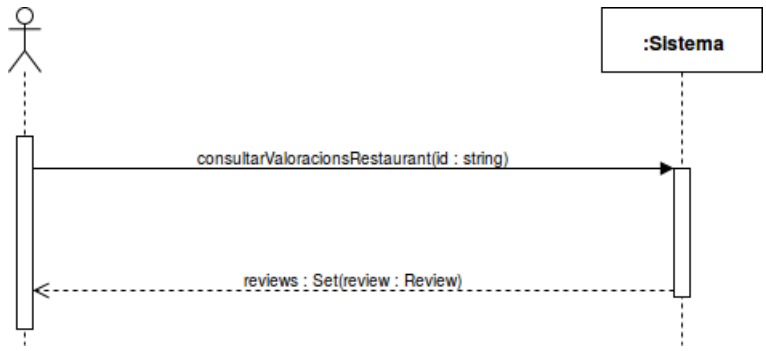

FIGURA 5.26: Esquema de comportament del cas d'ús #25

Context	Sistema :: consultarValoracionsRestaurant(id : string)
Precondició	El restaurant amb id *restaurantId* existeix dins la base de dades.
Postcondició	L'aplicació mostra les valoracions del restaurant especificat.

Capítol 6

Disseny

Després d'haver mencionat, de forma molt específica, com és l'aplicació tant en requisits com en objectius, hem de veure com està construïda internament, és a dir, quin tipus d'arquitectura tècnica segueix, quin disseny de base de dades s'ha establert, com s'ha dissenyat el software implementat i amb quins patrons s'ha construït i finalment com s'ha definit la interfície de l'usuari.

6.1. Arquitectura del sistema

Després d'haver explicat quin és l'abast i quins objectius té *Wisebite*, es va prendre la decisió que el projecte esdevingués a una plataforma mòbil per a dispositius Android. La justificació d'aquest fet ve dividida en dos punts, primer el fet que sigui un mòbil i no pas una aplicació web, per exemple, i l'altre pel fet que només s'hagi implementat per Android i no pas per cap altra sistema operatiu d'aquest tipus.

En primera instància, el motiu principal pel qual *Wisebite* ha estat implementat i dissenyat per dispositius mòbils és el dinamisme que aporten i del qual no disposen els dispositius de sobretaula. El món de la restauració és un sector de moviment continu, tant per part de l'empleat com per part del client. Un treballador d'un bar o restaurant es mourà de forma contínua per l'establiment i un client no sol acudir al mateix restaurant sempre. En conseqüència, el fet que *Wisebite* estigui disponible en telèfons mòbils o bé tauletes permet als usuaris d'aquesta plataforma interactuar amb ella des de qualsevol lloc i de forma més còmode interactuant amb la pantalla tàctil que disposa el terminal.

Per altra banda, *Wisebite* s'ha especialitzat en sistemes operatius Android i no pas en algun altre per un seguit de factors que es comenten a continuació.

En primer lloc és important recordar un dels tres factors pels quals sistemes d'aquest tipus no han acabat d'introduir-se dins el sector: el factor econòmic. En el mercat dels dispositius mòbils és vist i reconegut que els dispositius Android, donat el gran número de terminals en els quals opera, són de preu més reduït que pas els sistemes iOS, implementats per Apple. Així doncs, és important oferir hardware barat als establiments que implantin *Wisebite* per així reduir l'impacte econòmic que els pot ocasionar.

En segon lloc, per què només per Android i no pas també per iOS? És cert que existeix un seguit de *frameworks* que et permet desenvolupar aplicacions mòbils híbrides, és a dir, per a tots els sistemes operatius mòbils.

La problemàtica principal d'aquest tipus de desenvolupament és el fet de no poder utilitzar tots els avantatges que et permet un sistema operatiu natiu en concret. Per exemple, si analitzem les aplicacions mòbils natives d'Android, veiem que tot el *backend* de l'aplicació pot ser escrit en Java, C++ o bé Kotlin i el *frontend* amb llenguatge d'etiquetes com és XML. En canvi, si es realitzes amb un dels *frameworks* que ofereix el mercat s'escriuria en HTML, CSS i JavaScript, com si es tractés d'una pàgina web. Són maneres d'implementar aplicacions molt diferents, i per molt que aquests *frameworks* ho vulguin simular al màxim no ho acaben d'aconseguir del tot. Així doncs, donat que un dels objectius clau que té *Wisebite* és destacar per la seva interfície és molt millor implementar en llenguatge natiu donats els avantatges que t'ofereix.

Per tant, donada aquesta justificació, l'aplicació correrà en dispositius exclusivament Android a partir de l'*API 21* o versió *5.0 Lollipop*, és a dir, en un 88.6%[19] dels dispositius de la marca de Google.

Aquesta aplicació es comunica amb una base de dades no relacional (NoSQL) anomenada *Firebase*. En apartats posteriors dins d'aquest mateix capítol s'explicarà com s'ha implementat l'esquema de dades dins d'aquest gestor, però abans en aquest apartat és important conèixer el perquè de l'elecció de *Firebase* com a base de dades per a *Wisebite*.

Firebase va ser comprada per Google el maig del 2016, fet que aporta serietat, fiabilitat i robustesa com a base de dades, és a dir, pertany a una de les institucions més importants dins el sector de la informàtica, per no dir el que més. Tot i així, el factor més important i pel qual es va decidir utilitzar aquesta base de dades no relacional és que la comunicació es realitza en temps real. Aquest aspecte és realment important sabent quina és la temàtica de l'aplicació. Es considera molt important aquest fet, ja que l'actualització automàtica de les dades sense necessitat de refresc manual facilita moltíssim la feina d'un cambrer o d'un cuiner a l'hora d'interactuar amb la plataforma. Així doncs, per aquest fet es va decidir implantar la base de dades dins de *Firebase*.

Per últim, l'aplicació emmagatzemarà les dades temporals en un *SQLite* que actuarà de memòria cau dins de *Wisebite*. La justificació de l'ús d'aquesta tecnologia ve donat pel fet que Android la utilitza de forma predefinida en les seves aplicacions natives.

Per clarificar més el concepte i veure quina és la interacció de l'usuari amb *Android*, *Firebase* i *SQLite*, es mostra a continuació una gràfica de l'arquitectura tècnica que s'ha decidit utilitzar a *Wisebite*.

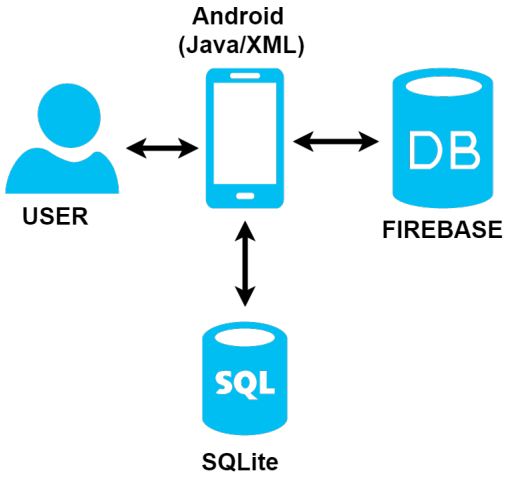

FIGURA 6.1: Arquitectura tècnica del sistema

6.2. Disseny de la base de dades

Com s'ha comentat en l'apartat anterior, la base de dades de *Wisebite* serà *Firebase*. Aquesta base de dades funciona amb una estructura en forma d'arbre basant-se en el format JSON[20].

Existeix un node pare o arrel anomenat *wisebite-f7a53* del qual pengen tots els nodes restants. Per a tal de dissenyar una bona base de dades NoSQL de tipus clau-valor, com es *Firebase*, és necessari tenir en compte un seguit de conceptes per seguir les bones pràctiques que les mateixes guies del gestor recomanen:

- El fet d'accedir a un node, fa que hagi de descarregar tot l'arbre del qual penja aquest node (amb tots els seus respectius fills). Per tant, la solució és disgregar els elements. Cal tenir cura en relacionar els components o nodes de la base de dades per poder-los processar correctament. Un bon mètode seria associar els elements via el seu identificador i no pas tot l'element.
- Per realitzar relacions entre, per exemple, un restaurant i els usuaris als quals hi treballen podríem pensar en afegir un node fill per a cada usuari del restaurant. Això és ineficient en cas de tenir restaurants amb un gran nombre d'usuaris en ells. En conseqüència, la solució seria utilitzar índexs. Per a cada restaurant s'hauria de tenir un llistat d'usuaris on es representen els usuaris pertanyents a ell.

Llavors, en definitiva, el missatge més important és que cal tenir en compte que encara que *Firebase* tingui una estructura en forma d'arbre, no s'ha de construir esquema de profunditat molt gran, sinó millor crear un arbre amb la major amplada possible. Això farà més eficient les consultes a la base de dades.

Després d'haver estudiat i entès quines són les bones pràctiques d'una base de dades no relacional de tipus clau-valor, s'ha dissenyat de tal manera que pengen nou nodes del node arrel, on allà s'emmagatzemen totes les instàncies d'aquella entitat. Cadascuna d'elles serà identificada a partir d'un identificador aleatori. Cadascun d'aquest nodes representen les nou entitats que han estat explicades en capítols anteriors. A continuació es mostra l'estructura de la base de dades en format JSON.

```
"wisebite-f7a53" : {
   "dish" : {
      "-KlPZWq5d4cFO6NJ0nc4" : {
         "description" : "Classic",
         "id" : "-KlPZWq5d4cFO6NJ0nc4",
         "name" : "Paella",
         "price" : 12,
         "reviews" : {
            "-KlyJ20Nr3OpnA0JLqk5" : true,
            "-Km3Q9par9GavvZHt-4L" : true,
            "-KmNrn1Gxu66N6tc5u32" : true
         }
      },
      ...
   },
   "image" : {
      "-KlPTsD8xjazZ8KceJ6E" : {
         "description" : "Profile photo",
         "id" : "-KlPTsD8xjazZ8KceJ6E",
         "imageFile" : "https://lh5.googleuser..."
      },
      ...
   },
   "menu" : {
      "-KlPeMorCJsXKHh024Ui" : {
         "description" : "nu",
         "id" : "-KlPeMorCJsXKHh024Ui"
         "mainDishes" : {
            "-KlPeLzph1_Sz7UGcnMc" : true
         },
         "name" : "menu",
         "otherDishes" : {
            "-KlPeLzrWp_u0Fx3bNPW" : true
         },
         "price" : 0.5,
         "secondaryDishes" : {
            "-KlPeLzrWp_u0Fx3bNPV" : true
         }
      },
      ...
   },
   "openTime" : {
      "-KlPY8bUts4c9domVn5d" : {
         "endDate" : {
            "date" : 29,
            "day" : 1,
            "hours" : 22,
            "minutes" : 0,
            "month" : 11,
            "seconds" : 0,
```

```
            "time" : -180000000,
            "timezoneOffset" : 0,
            "year" : 69
         },
         "id" : "-KlPY8bUts4c9domVn5d",
         "startDate" : {
            "date" : 29,
            "day" : 1,
            "hours" : 8,
            "minutes" : 0,
            "month" : 11,
            "seconds" : 0,
            "time" : -230400000,
            "timezoneOffset" : 0,
            "year" : 69
         }
      },
      ...
   },
   "order" : {
      "-KlZianOlkFpEkNFoiF7" : {
         "date" : {
            "date" : 1,
            "day" : 4,
            "hours" : 19,
            "minutes" : 35,
            "month" : 5,
            "seconds" : 58,
            "time" : 1496338558814,
            "timezoneOffset" : -120,
            "year" : 117
         },
         "id" : "-KlZianOlkFpEkNFoiF7",
         "lastDate" : {
            "date" : 1,
            "day" : 4,
            "hours" : 20,
            "minutes" : 37,
            "month" : 5,
            "seconds" : 53,
            "time" : 1496342273505,
            "timezoneOffset" : -120,
            "year" : 117
         },
         "orderItems" : {
            "-KlZian7yy0XgIJaGrCV" : true,
            "-KlZianHOSo5hbzboxWl" : true,
            "-KlZianKOdNMI1r06jnh" : true,
            "-KlZianMs7fEFfK3JJw7" : true
         },
         "tableNumber" : 9
```

```
      },
      ...
   },
   "orderItem" : {
      "-KlYSoFkb_Z7WmMsvNEy" : {
         "delivered" : false,
         "dishId" : "-KlPeMorCJsXKHh024Uh",
         "id" : "-KlYSoFkb_Z7WmMsvNEy",
         "menuId" : null,
         "paid" : true,
         "ready" : false
      },
      ...
   },
   "restaurant" : {
      "-KlPZWqHH5fEgUoCrGG5" : {
         "description" : "De tota la vida, de poble.",
         "dishes" : {
            "-KlPZWq-xR39JAJBEitN" : true,
            "-KlPZWq1k46MlYinfido" : true,
            "-KlPZWq3w_Zwu63am50p" : true,
            "-KlPZWq4OshyHXIufnFO" : true,
            "-KlPZWq5d4cFO6NJ0nc4" : true
         },
         "externalOrders" : {
            "-Kldj4NCYd-3ZN9H93Mh" : true,
            "-Kldk-C2wrOFKZmrtOPB" : true,
            "-Kldt2L6ZRJouCK75rzP" : true,
            "-KlyIcuVp1Lx3U7235PN" : true,
            "-Km3Q3B4u_Sq65WCv0bq" : true,
            "-KmNqGKk-EURs6mBnQNz" : true
         },
         "id" : "-KlPZWqHH5fEgUoCrGG5",
         "location" : "Sant Celoni",
         "menus" : {
            "-KlPZWq7lAVXkCYqe-Z9" : true,
            "-KlPZWqDD_UhNEVauAtt" : true,
            "-KlPZWqF14E1wMPYIHT6" : true
         },
         "name" : "Cerveseria Ceres",
         "numberOfTables" : 30,
         "openTimes" : {
            "-KlPY8bUts4c9domVn5d" : true,
            "-KlPY8bnpSPVrvXvEL6F" : true,
            "-KlPY8brvVHqL0KXCfEi" : true,
            "-KlPY8bz9HQijw46-WWE" : true,
            "-KlPY8c2YzO4D7yZ99zU" : true
         },
         "phone" : 666544283,
         "reviews" : {
            "-KlyJ20VJ77rNt1Yfg2y" : true,
```

```
            "-Km3Q9pddyCuqWYlCz-C" : true,
            "-KmNrn1L9CSgClEUHv_I" : true
         },
         "users" : {
            "alsumo95@gmail@com" : true
         },
         "website" : "ceres.com"
       },
       ...
    },
   "review" : {
      "-KltwNoh4fDoG7tIWZYx" : {
         "comment" : "Rico!",
         "date" : {
            "date" : 8,
            "day" : 4,
            "hours" : 9,
            "minutes" : 21,
            "month" : 5,
            "seconds" : 12,
            "time" : 1496906472539,
            "timezoneOffset" : -120,
            "year" : 117
         },
         "id" : "-KltwNoh4fDoG7tIWZYx",
         "points" : 4,
         "userId" : "alsumo95@gmail@com"
      },
      ...
   },
   "user" : {
      "alripol95@gmail@com" : {
         "email" : "alripol95@gmail.com",
         "id" : "alripol95@gmail@com",
         "imageId" : "-Km0MSeurykImTbkBPEl",
         "lastName" : "Ripol",
         "myOrders" : {
            "-Km0RcztK3XoafmEAx4P" : true
         },
         "myRestaurants" : {
            "-Km0Nbs9ioiglQBNVNvv" : true
         },
         "name" : "Albert",
         "ordersToReview" : {
            "-Km0RcztK3XoafmEAx4P" : true
         }
      },
      ...
   }
}
```

6.3. Disseny de software

Durant el transcurs de la carrera, i en especial durant l'especialitat d'Enginyeria del Software, s'ha emfatitzat molt en el concepte que programar bé no és simplement que funcioni el programa implementat, sinó que segueixi uns patrons específics que et permeti tenir un codi fàcil d'entendre i re-usable, ja que al cap i a la fi el codi es llegeix més cops dels que s'escriu.

És per això que en la implementació del projecte de *Wisebite* s'ha seguit un conjunt de patrons de disseny de software anomenats patró repositori, patró servei, patró factoria i el més conegut anomenat MVC (Model-Vista-Controlador).

6.3.1. Patrons de disseny

S'ha utilitzat, com s'ha comentat anteriorment, quatre patrons de disseny durant la implementació de *Wisebite*.

- **Patró repositori**: utilitzat per evitar l'accés directe a les dades, és a dir, des de la vista o més concretament en Android, des de l'*Activity*. En aquest cas es permet desacoblar al màxim l'accés a *Firebase*. Tota classe que serà traduïda en un objecte persistent en *Firebase*, seguint el model de la base de dades mostrat al capítol anterior, ha d'implementar una interfície *Entity*, i definirem una interfície *Repository* parametritzada, sobre la qual caldrà implementar una classe *RepositoryXXXX* que la implementi (on XXXX defineix l'objecte que emmagatzema el repositori), que seran instanciades i utilitzades amb els seus mètodes ja definits per accedir a *Firebase*. Aquesta classe *RepositoriXXXX* hereta d'una altre anomenada *FirebaseRepository* per així afegir una capa més d'abstracció.

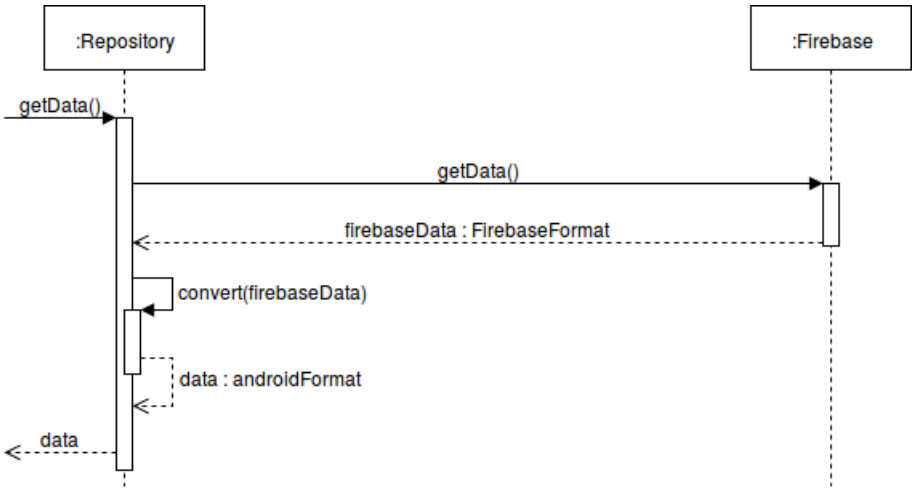

FIGURA 6.2: Patró repositori

- **Patró servei / injecció de dependència**: utilitzat per crear una capa d'abstracció sobre els dipòsits prèviament esmentats. En aquestes noves classes, les quals hereten d'una interfície anomenada *Service*, ens permet executar la lògica del programa, sense necessitat d'accedir directament als repositoris. Aquests serveis contenen una instància única del repositori. Gràcies a la injecció de dependència podem realitzar tests de la lògica del programa sense haver de fer

servir la base de dades real mitjançant *mocking* dels repositoris. Malauradament, donat al factor temporal i que especificarem en capítols posteriors, no s'ha posat a la pràctica la utilitat dels tests. Tot i així, el codi del programa està ben preparat per poder afegir tests fàcilment.

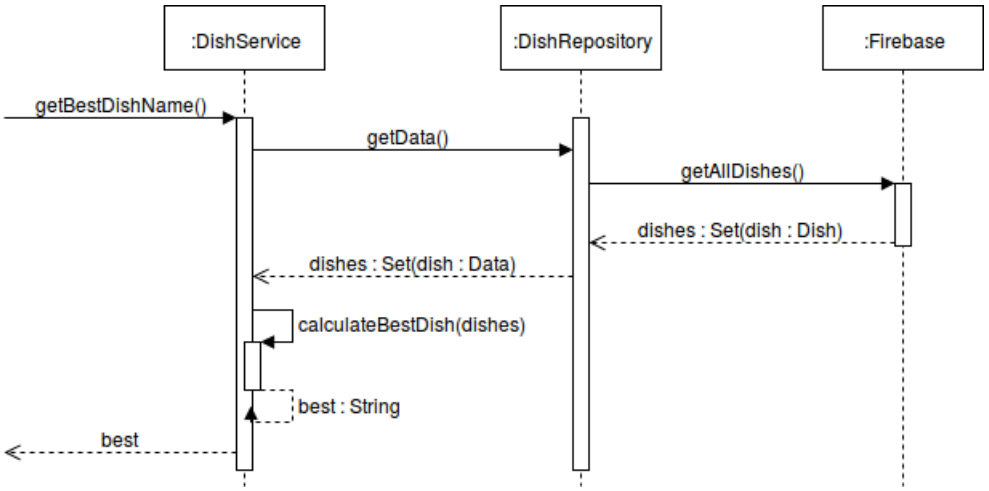

FIGURA 6.3: Patró servei

- **Patró factoria**: utilitzat per a la crida de serveis amb la creació d'un singleton anomenat *ServiceFactory*. Permet abstreure la funcionalitat de creació del servei, permetent així evitar la creació repetitiva del servei a cada ús que se li dóna, fet que comportaria tenir moltes instàncies del mateix objecte repartides per l'execució de l'aplicació. Separa la instanciació del servei del seu ús. Permet que es faci injecció de dependències sense haver de saber l'estructura d'aquesta.

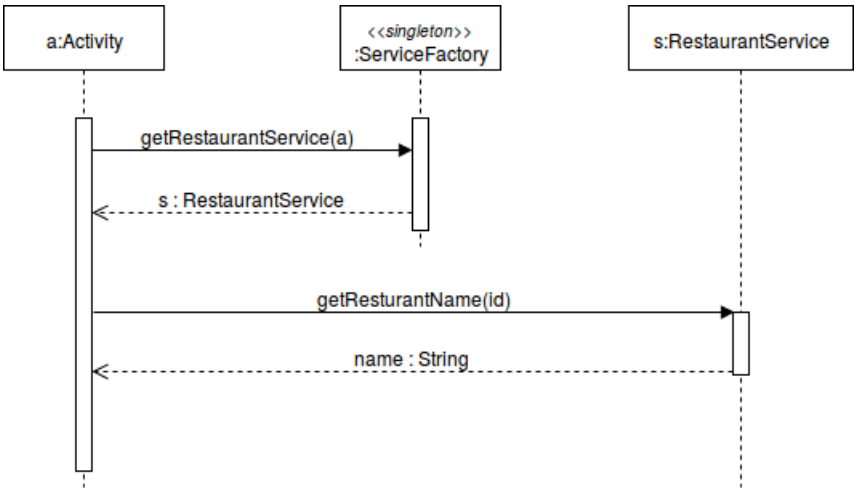

FIGURA 6.4: Patró factoria

- **MVC**: utilitzat per organitzar l'estructura de l'aplicació. En el cas de *Wisebite* en tractar-se d'Android les vistes són les activities. D'altra banda els controladors s'han anomenat serveis.

Totes les classes Java comentades han estat adjuntades a l'apèndix A.

6.4. Disseny de la interfície

Com es va comentar en capítols anteriors, concretament en l'estudi de mercat, actualment existeixen aplicacions que en qüestió de funcionalitats s'assemblen o s'acosten al que busca *Wisebite*, però la gran majoria d'aquestes no han dedicat molt temps en l'estudi de la gestió de la interfície d'usuari. En canvi, en aquest projecte s'ha dedicat més del temps mínim requerit per construir una interfície el millor possible per l'usuari.

El primer factor que es va tenir en compte va ser l'ús de les directrius que marca el *Material Design*, un estil de disseny d'interfícies que va néixer amb l'API 21 o versió 5.0 d'Android. Amb l'aparició d'aquest estil s'ha buscat en tot moment facilitar l'ús de les aplicacions a les persones que els hi costa més utilitzar aplicacions per a smartphones. Interfícies molt més intuïtives, de colors i formes característiques, amb més dinamisme en la navegació per la plataforma i, sobretot, un patró de disseny estable en tots els dispositius del mercat.

Wisebite ha decidit seguir aquest model, ja que és molt important no canviar molt la interfície a la qual està acostumat l'usuari amb altres aplicacions. Per aquest motiu doncs s'ha decidit seguir els dissenys preestablerts que facilita les guies de *Material Design*.

Durant la navegació de l'aplicació s'ha buscat tenir transicions entre vistes per aplicar-li un extra de dinamisme en l'experiència de l'usuari. Es manté en tot moment a l'usuari assabentat de quin és l'estat de l'aplicació per així adquirir un grau més alt d'atenció. S'ha buscat que les vistes fossin les més intuïtives possibles, i s'ha decidit utilitzar una paleta de colors neutra amb l'ús del gris, negre i blanc per no ser massa arriscats en el joc de colors i buscar quelcom no massa lluny de la normalitat.

Per aconseguir un producte el més pròxim a l'usuari real s'ha realitzat un seguit d'iteracions que han estat provades per usuaris externs al sistema. Se'ls hi mostrava una versió de l'aplicació, la qual la provaven sense cap tipus de guia prèvia ni aprenentatge i s'estudiaven quins eren els seus moviments naturals a l'hora d'interactuar amb la plataforma. A més a més, s'escoltava l'opinió dels usuaris respecte als aspectes que ells i elles millorarien. Tot i així, la metodologia d'iteracions serà explicada amb més detall en capítols posteriors.

Un cop explicat el concepte amb el qual s'ha treballat de base, a continuació es mostren vuit captures de pantalla de la plataforma per així poder visualitzar l'estat del producte final, després d'haver seguit les directrius comentades anteriorment.

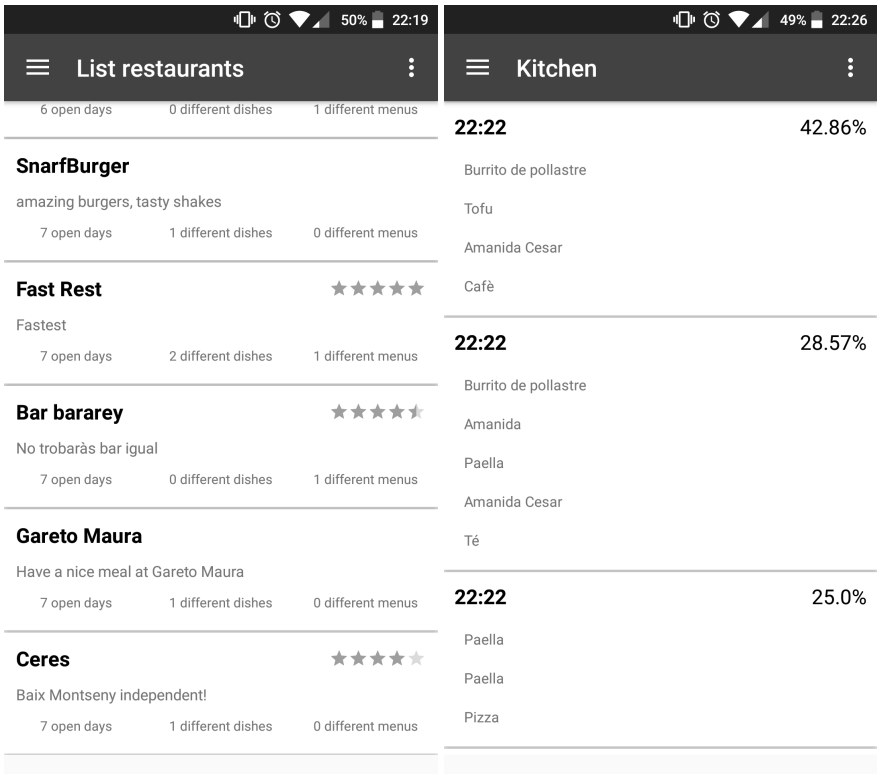

FIGURA 6.5: Captures de pantalla de Wisebite: llistat de restaurants i mode cuina

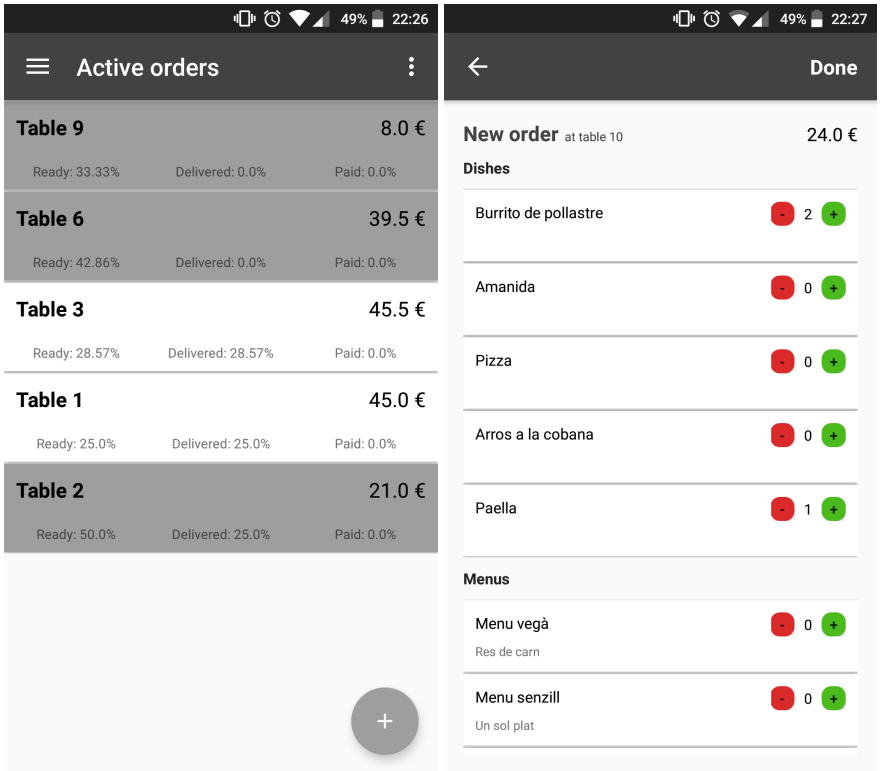

FIGURA 6.6: Captures de pantalla de Wisebite: llistat de comandes actives i consulta de comanda

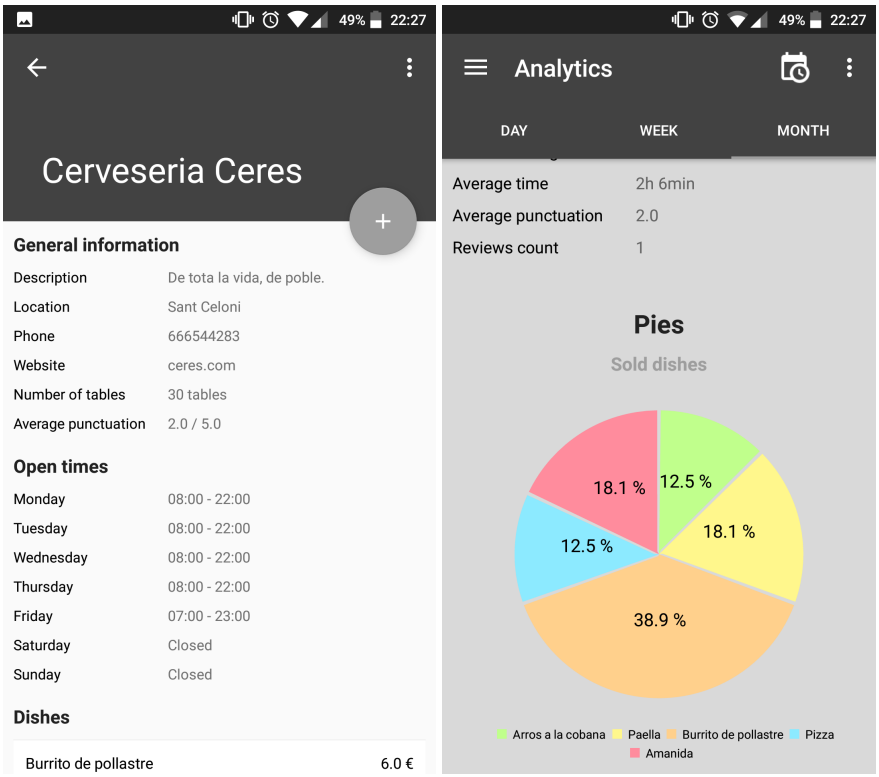

FIGURA 6.7: Captures de pantalla de Wisebite: consulta de restaurant i estadístiques

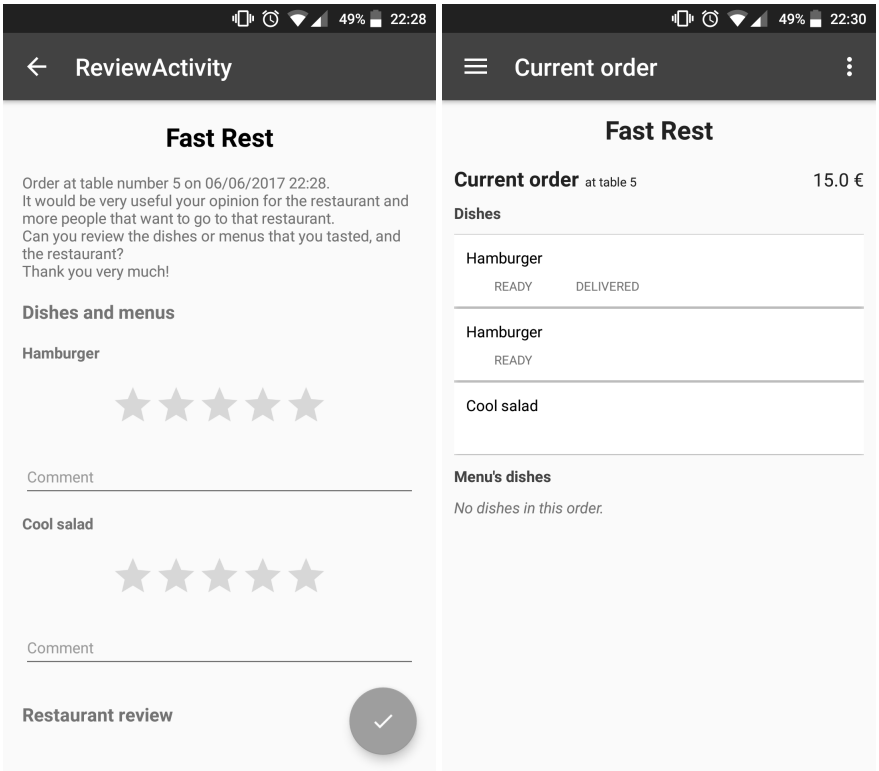

FIGURA 6.8: Captures de pantalla de Wisebite: valoració i comanda actual

Capítol 7

Implementació

Un cop decidit com serà el sistema, és a dir, quins requisits, objectius i abast tindrà, i quin disseny s'aplicarà en ell, el següent es parla de la implementació per si mateixa. Les decisions preses anteriorment i explicades en capítols anteriors han marcat el que s'explicarà en aquest capítol. En primera instància es realitzarà un repàs per les tecnologies i eines utilitzades durant el desenvolupament de *Wisebite*, i per acabar es seguirà pels detalls de la implementació del projecte.

7.1. Tecnologies i eines utilitzades

Durant el desenvolupament del projecte s'ha utilitzat tot tipus de tecnologies, és per això que s'ha decidit subdividir-ho en quatre conceptes. Començant pels llenguatges de programació, seguint per les bases de dades, mencionant les eines utilitzades i finalitzant amb les llibreries externes utilitzades.

7.1.1. Llenguatge de programació

Java[21]
Com es va comentar anteriorment en la decisió de l'arquitectura, es va decidir construir una aplicació Android. Aquest sistema operatiu permet desenvolupar el *backend* de les seves aplicacions natives en Java, C++ i Kotlin. Donat que el desenvolupador tenia més coneixement en desenvolupament d'aplicacions Android en Java, es va decidir utilitzar aquest llenguatge. A més a més, en ser el llenguatge de programació més utilitzat en el món des de ja fa anys, la comunitat d'usuaris és molt gran. En conseqüència, la probabilitat de trobar solucions a possibles problemes que es poden ocasionar és molt més alta que pas amb els altres dos llenguatges.

XML[22]
El desenvolupament del *frontend* en Android es realitza en llenguatge d'etiquetes, específicament XML. A diferència del *backend*, en aquest cas no es tenia elecció dins del context de desenvolupament d'aplicacions natives.

LaTeX[23]
La realització de la memòria del projecte ha estat realitzada amb el llenguatge de software lliure anomenat LaTeX. És un llenguatge que et permet realitzar documents pdf a través d'unes directives. L'esforç inicial d'aprendre aquest llenguatge es va contemplar en la planificació temporal del projecte, i tot i conèixer que seria d'un període més o menys considerable, es va contemplar l'esforç com útil en el projecte per la professionalitat i la robustesa que reflecteix el document.

FIGURA 7.1: Llenguatges de programació

7.1.2. Bases de dades

Firebase[24]
Com s'ha comentat en capítols anteriors amb més detall, aquest projecte utilitza Firebase com a base de dades. Aquesta base de dades no relacional de tipus clau-valor ha aportat molts avantatges en el desenvolupament de l'aplicació donades les seves característiques, que anaven acord amb els objectius de *Wisebite*.

SQLite[25]
Al ser una plataforma Android, l'ús d'SQLite ve implícit en el desenvolupament del projecte. Aquesta base de dades actua com a memòria cau per així optimitzar l'obtenció de recursos durant la navegació per l'aplicació.

FIGURA 7.2: Bases de dades

7.1.3. Eines

Android Studio[26]
Google, com a propietari d'Android, recomana l'ús d'Android Studio per al desenvolupament de les seves aplicacions natives. A més a més, gràcies a la col·laboració de *JetBrains*, aquest IDE disposa de moltes funcionalitats que faciliten l'ús del desenvolupador. Es pot nodrir d'un gran seguit de *plugins* que permet fer-te el programa teu i desenvolupar amb més qualitat.

Genymotion[27]
Durant el desenvolupament, les proves principals de l'aplicació s'han realitzat amb un terminal físic. Tot i així, per provar com funciona l'aplicació en diferents dispositius s'ha utilitzat Genymotion, un emulador al qual li pots afegir tot tipus de dispositiu Android, sigui mòbil o tauleta. Així s'ha pogut experimentar i provar com ha quedat el disseny de la interfície en plataformes diferents.

TeXstudio[28]
Per al desenvolupament de la memòria s'ha utilitzat el IDE de LaTeX, anomenat TeXstudio. Aquest IDE et permet escriure en LaTeX, de forma més agradable i ràpida

gràcies principalment als seus *shortcuts*. A més a més, pots alimentar aquest programa de plugins que et facilitaran més la redacció de la memòria, com per exemple un plugin de *Softcatalà* que et permet tenir un corrector ortogràfic i gramatical de català durant l'escriptura del document final.

Git[29]
L'ús de Git durant el desenvolupament de *Wisebite* ha estat vital per a la construcció del producte final. El poder de poder tenir un conjunt de versions del teu programa ha permès desenvolupar de forma més fàcil i sense por a provar coses noves i no trencar el codi anterior.

GitHub[30]
GitHub és considerat el gestor de repositoris remots més conegut del sector. Gràcies a les funcionalitats de les quals disposa, com poden ser les *issues* o les *pull requests*, han permès tenir un codi documentat i poder tenir un bon històric del desenvolupament del projecte. En apartats posteriors s'explicarà amb més detall com a funcionat el flux de treball amb l'ús de GitHub.

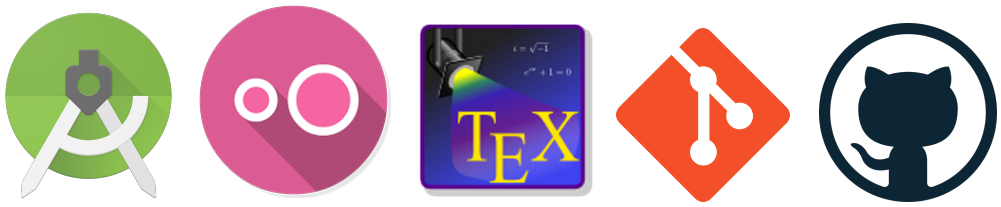

FIGURA 7.3: Eines

7.1.4. Llibreries externes

Google Play Services[31]
La majoria d'aplicacions Android utilitzen aquesta llibreria, ja que et permet accedir a totes les APIs disponibles de les quals disposa Google. En aquest projecte s'ha utilitzat només per això l'API d'autentificació, ja que el *Log in* de l'aplicació es fa via Google.

FloatingActionButton[32]
Durant el desenvolupament del projecte es va voler integrar un *FloatingActionButton*, botó molt característic del *Material Design*, però amb un menú integrat. Característica que no estava disponible en les llibreries natives d'Android. És per això que es va integrar aquesta llibreria externa de software lliure que ho permetia.

Picasso[33]
En les funcionalitats de gestió d'usuari, era necessària la interacció de les imatges amb l'aplicació. Concretament, es necessitava carregar les imatges des d'un URL. Es va estar investigant quina era la millor opció i es va acabar optant per Picasso. Una llibreria externa de software lliure que et permet carregar imatges via un URL de forma molt fàcil. Aquesta llibreria a més t'emmagatzema de forma automàtica la càrrega d'imatge a la memòria cau, fet que optimitza la plataforma i millora el flux de l'usuari amb aquesta.

MPAndroidChart[34]
Una de les funcionalitats de les quals disposa l'aplicació, i que s'ha esmentat anteriorment, és l'anàlisi de dades del restaurant. És per això que es va voler introduir un conjunt de gràfiques per visualitzar més fàcilment l'estat estadístic del restaurant. *MPAndroidChart* és una llibreria de software lliure que et permet realitzar tot tipus de gràfiques, ja sigui un histograma, un gràfic de barres o circular i molts més.

Lombok[35]
Per optimitzar més el desenvolupament del projecte es va voler utilitzar la llibreria externa de software lliure anomenada Lombok, que et permet tenir codis més nets i més fàcils d'entendre. Per exemple, amb un simple @Getter a la capçalera d'una classe et permet tenir totes les funcions get dels atributs privats de la classe en qüestió.

FIGURA 7.4: Llibreries externes

7.2. Detalls de la implementació

Durant la implementació del projecte s'han seguit un conjunt de directrius específiques que han marcat l'estat final del producte. En aquest apartat s'explicarà en primera instància la metodologia d'implementació que s'ha seguit per obtenir un bon codi, i finalment com s'ha redactat la memòria amb l'ús, com s'ha comentat anteriorment, de LaTeX.

7.2.1. Implementació de l'aplicació

La implementació de la qual disposa *Wisebite* no conté cap algorisme de computació molt complex, més aviat són operacions trivials. És per això que s'ha volgut emfatitzar en com el codi ha anat evolucionant i com s'ha volgut remarcar la importància de tenir un codi ben documentat, ja que el codi sempre es llegirà més cops dels que s'escriurà. Per tant, es important tenir aquest aspecte en compte.

Al seguir una metodologia àgil, com es va comentar en els primers capítols, és fàcil subdividir les tasques en històries d'usuari, les quals componen una funcionalitat independent del sistema. Així doncs, utilitzant les eines que ens facilita *git* i *GitHub*, es realitzava una branca per cada una de les històries d'usuari que s'implementava i cada *bug* o error que es trobava. Entenem com a branca o *branch* un mètode per ramificar part del codi d'un projecte, és a dir, desenvolupar part del codi de forma independent a la resta.

Aquest concepte té molt sentit en treballs en equip, ja que molts membres poden treballar de forma independent tot i tocar codi del mateix projecte. Malgrat que aquest projecte sigui individual no impedeix que les branques deixin de ser útils. Permet veure, amb una mica de retrospectiva, l'evolució del codi i garantir que el codi publicat en la branca principal (coneguda com *master*) és el correcte i funcional en plenes condicions.

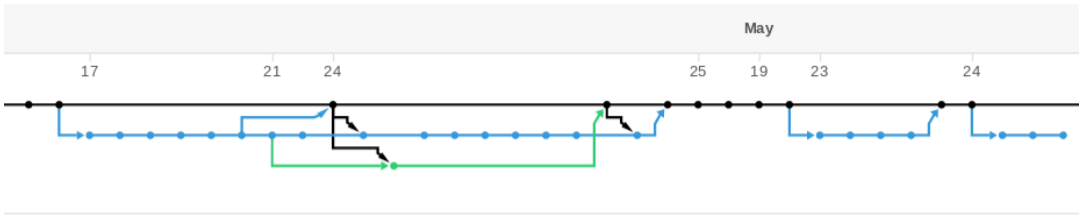

FIGURA 7.5: Gestió de branques

Un cop el codi d'una branca estava finalitzat es creava una *pull request* per revisar els canvis i acceptar-la en cas que fos correcte. Entenem *pull request* com un conjunt de canvis que es volen revisar per ser acceptats i ser fusionats amb la branca principal. Al ser un treball individual la revisió només la realitza el desenvolupador del fragment de codi a revisar, però et permet disposar d'un historial de canvis i saber quina ha estat l'evolució i quan es van realitzar els canvis.

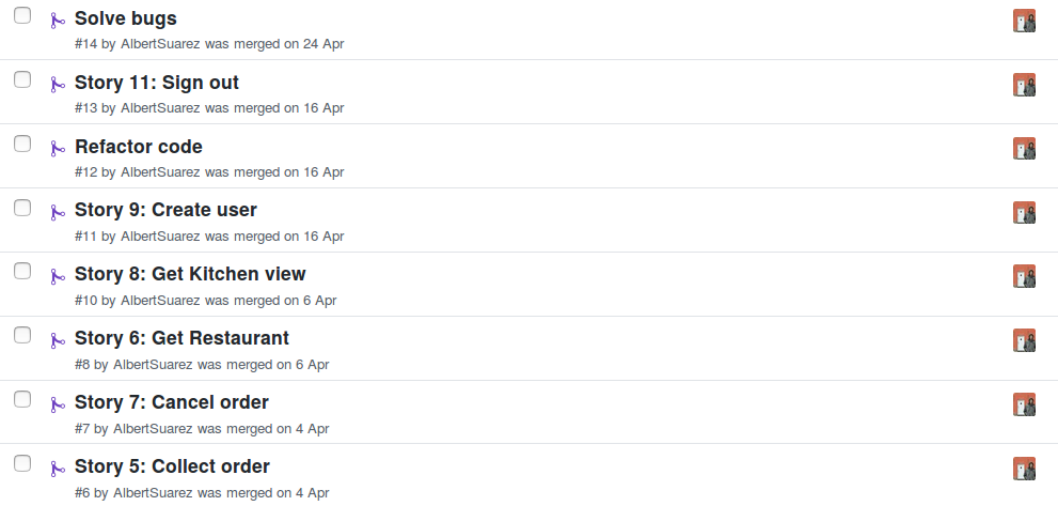

FIGURA 7.6: Gestió de pull requests

Així doncs el codi va evolucionant segons s'ha explicat, és a dir, mantenint la branca principal sempre funcional i desenvolupant en branques independents de la *master*. Seguint aquesta metodologia es pot obtenir un codi prou documentat. Tot i així, com es pot arribar a entendre, hi ha punts dins la cronologia del codi que són més importants que la resta. Per aquest motiu s'ha decidit realitzar diferents versions del codi, les quals s'han anat provant amb usuaris externs al sistema, aspecte que es comentarà en el capítol següent.

Per a poder posar a la pràctica aquest fet s'utilitzen els *tags* o les *releases* de GitHub. Aquestes eines et permeten marcar de manera destacada un punt de la cronologia del codi i posar-li un nom i una descripció. Així doncs, durant l'historial el codi de *Wisebite* s'han indicat deu etiquetes amb les respectives versions del codi.

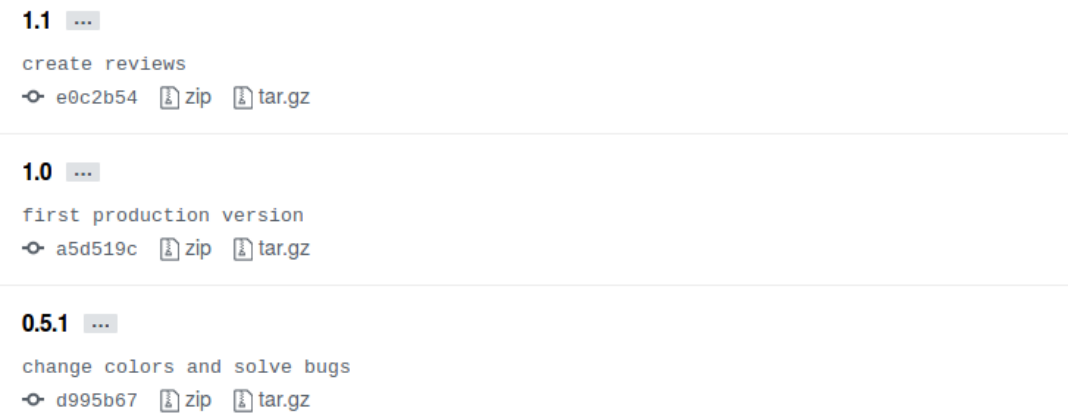

FIGURA 7.7: Gestió de tags

Per altra banda també s'ha utilitzat les *issues* de GitHub com un estil de bloc de notes amb els errors o millores que es volen realitzar a la plataforma i que en aquell moment no eren prioritats.

I per últim, i ja lluny de la gestió de versions del codi i de la seva cronologia, s'ha utilitzat *JavaDoc* com a estil de documentació del codi Java. Per així emfatitzar en el concepte que el codi es llegeix més cops del que s'escriu.

7.2.2. Implementació de la memòria

La metodologia que s'ha seguit per construir i redactar la memòria d'aquest treball final de grau consisteix en un conjunt de directrius.

En primera instància es va dissenyar meticulosament el *outline* de la memòria repassant la manquesa d'elements crucials que no havien de faltar. Perquè el marge d'error fos el menor possible el director del projecte va ajudar l'alumne facilitant-li exemples de memòries d'alumnes que ell va portar amb anterioritat perquè veies l'estil que ha de tenir una memòria. Així doncs, amb la combinació d'aquests dos aspectes es va dissenyar el *outline* o índex complet.

Com a següent pas de la cronologia de la memòria, es van buscar plantilles en format LaTeX. Un cop es va trobar la plantilla adient al que es necessitava, es va adaptar al projecte en qüestió i es va començar a redactar. El mètode de redacció s'ha basat, semblant a la implementació del codi, en les *issues* de GitHub. Cadascun dels apartats s'enregistrava en el repositori i se li atribuïa una etiqueta indicant si l'apartat s'havia de modificar o bé redactar d'un bon inici.

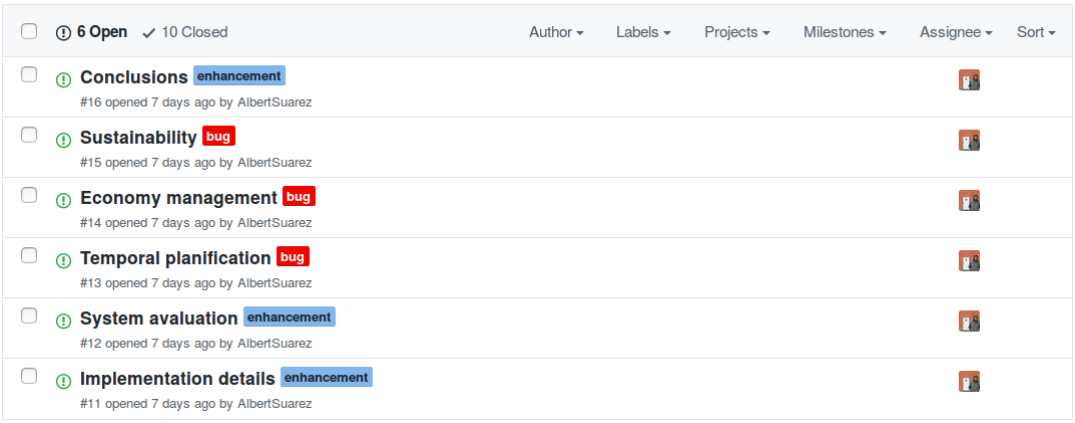

FIGURA 7.8: Gestió d'issues

Les diferents versions de la memòria s'han anat validant per part del director del projecte, per així conèixer si el progrés d'aquesta és el correcte.

Capítol 8

Avaluació del sistema

Tot projecte ha d'estar validat i avaluat per un conjunt de directrius concretes per contemplar el projecte com a finalitzat. En aquest capítol s'explicarà com s'han realitzat aquestes proves i quines eines s'han utilitzat. Les proves s'han realitzat durant el desenvolupament i la fase final d'aquest, quan ja el producte pràcticament es podia donar per finalitzat.

8.1. Proves durant el desenvolupament

Com s'ha comentat en el capítol anterior, s'ha tingut molta cura en l'enregistrament de versions de l'aplicació. Quan un conjunt d'històries d'usuari es considerava implementat, es versionava aquella part del codi i es marcava constància en el repositori.

Durant el desenvolupament de les funcionalitats sempre es provava el funcionament d'aquestes fins que es complissin tots els criteris d'acceptació de la història d'usuari corresponent. Tot i així, el fet que un sol usuari proves aquell conjunt de funcionalitats no es va considerar suficient per validar si aquella versió del codi era funcional amb els requisits especificats.

Partint d'aquest aspecte, es va decidir utilitzar les funcionalitats de les quals disposa *Google Play* per penjar versions d'aplicacions alfa i/o beta. Són versions que no són públiques de cara al públic però que si pots retransmetre a un conjunt d'usuaris concret. Es va decidir demanar el favor a un seguit d'estudiants de la facultat, als quals se'ls demanava la prova de les funcionalitats concretes d'aquella versió. S'especificava que prestessin atenció en els moviments que realitzaven i com era la interacció amb la plataforma. Un cop es provava l'aplicació, es realitzava una retrospectiva amb cadascun d'ells per veure quins aspectes havien trobat positius i quins mancaven en el flux de l'aplicació.

Per a poder seguir aquest procediment més pròximament es va decidir utilitzar les funcionalitats que et permetia *Firebase*. Ja que, a més de ser una base de dades com s'ha comentat anteriorment, disposa d'un conjunt d'eines que et permet saber les estadístiques de l'aplicació i, en cas de fallida, quina traça d'error havia generat dita fallida. Així doncs, quan hi havia una fallida en el transcurs de l'aplicació aquesta era notificada a *Firebase* i aquest captava tota la informació d'error. Informació com la mateixa traça d'error però també el model del mòbil utilitzat, la versió d'Android, el percentatge de bateria en aquell precís moment i més detalls per a poder identificar de la forma més fàcil possible el motiu d'aquella fallida.

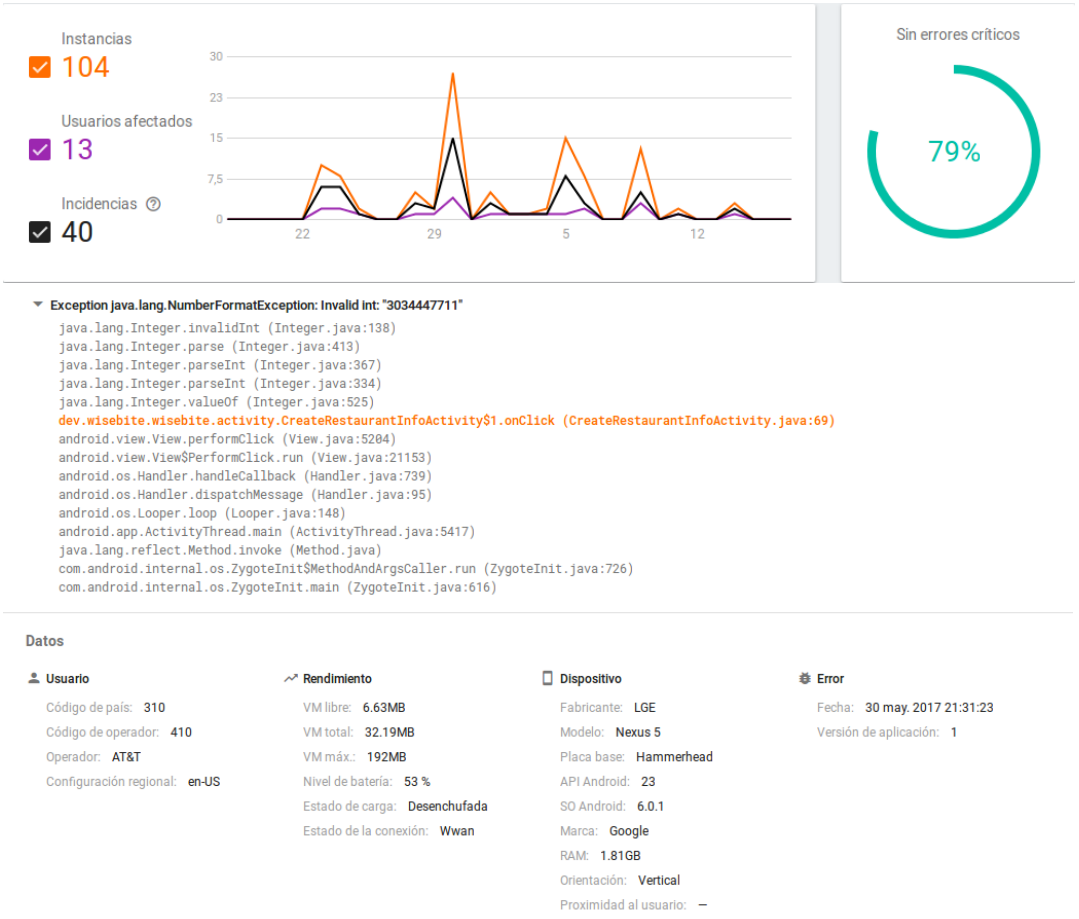

FIGURA 8.1: Estadístiques d'errors proporcionades per Firebase

I així es va arribar fins a la versió *1.0* de la plataforma. Un cop arribats a aquest punt l'aplicació va ser desplegada de forma definitiva al *Google Play* i va ser publica a tot el món.

8.2. Proves finals

Un cop l'aplicació va formar part de la galeria d'aplicacions de *Google Play*, es va decidir emfatitzar les proves a un conjunt d'usuaris que no fossin de la branca d'informàtics, sinó possibles usuaris de l'aplicació en un mercat real. Es va contactar amb coneguts del sector de la restauració perquè es descarreguessin l'aplicació i la provessin. A aquests usuaris no se'ls va donar cap tipus de formació ni d'aprenentatge previ, ja que es volia estudiar com era d'intuïtiva *Wisebite*.

Després de dues setmanes de proves amb aquests usuaris, els quals van acabar convertint-se en un conjunt de 26 persones, es van versionar dues vegades més fins a la versió *1.2* de l'aplicació. Durant aquesta evolució del codi es va escoltar la seva veu i es va implementar les millores que es veien necessàries per *Wisebite*.

Capítol 9

Planificació temporal

En aquest capítol es presentarà la planificació inicial que es va realitzar, els possibles pla d'acció en cas d'alteracions i el que realment ha passat, és a dir, com ha anat la planificació i si hi ha hagut desviacions.

9.1. Calendari

La durada del treball final de grau és de quatre mesos i mig, és a dir, unes 18 setmanes. Comença el 13 de febrer amb l'inici del quadrimestre i acaba entre els dies 26 i 30 de juny amb la defensa oral del projecte. Tot i així, es podrà donar el projecte per finalitzat una setmana abans amb l'entrega de la memòria final perquè el director i ponents del projecte tinguin temps per analitzar-ho amb deteniment.

Cal dir per això que poden sorgir inconvenients durant el transcurs del projecte que alterin la planificació inicial planejada, o per altra banda, haver planificat a la baixa i finalitzar-lo abans de l'esperat. Tot i així, es realitzaran controls periòdics per controlar acordament que la planificació no se surti de la ruta esperada i, en cas que ho faci, corregir el rumb per adaptar-se.

Cal mencionar en la planificació, que es té en compte el fet que l'estudiant està cursant altres assignatures i que en períodes esporàdics del quadrimestre hagi de dedicar més temps a tasques d'aquestes, i que en conseqüència deixi en *Stand by* les tasques relacionades amb el treball final de grau.

9.2. Recursos

Durant l'execució d'un projecte, sempre hi ha un ús d'un conjunt de recursos. Per a la descripció d'aquests recursos necessaris per a la realització d'aquest projecte, cal especificar que existeixen tres grups de recursos: personals, materials i de software.

9.2.1. Recursos personals

Durant el període comentat anteriorment, l'estudiant i autor del projecte li dedicarà unes 30 hores setmanals al desenvolupament i realització del treball final de grau. A més a més, se li afegeix l'ajut en correccions, assessorament i seguiment del director d'aquest.

9.2.2. Recursos materials

Serà necessària la presència d'un lloc de treball físic per dur a terme el projecte. Aquest lloc de treball pot ser únic o variat segons la disponibilitat de l'estudiant en aquell moment. A més a més, això comportarà costos extres com per exemple electricitat.

Per a la implementació i redacció de la documentació serà necessària la disponibilitat completa d'un ordinador, tant sigui portàtil com de sobretaula. Aquest ordinador haurà de tenir instal·lat tot el programari necessari per construir el projecte. Serà necessari incloure un cost extra per instal·lar dit programari.

A més a més, com a últim recurs material, també tenir en compte els servidors en els quals s'emmagatzemarà les dades que utilitzarà la plataforma per funcionar, concretament *Firebase* com s'ha comentat anteriorment.

9.2.3. Recursos de software

Durant tota l'execució del treball final de grau s'utilitzarà un gran nombre de programes software, que seran necessaris per a la realització d'aquest. Un grapat d'aquest programari software ha estat comentat amb més detall en apartats anteriors.

En primer terme, s'utilitzarà alguna de les aplicacions disponibles a *Google Apps*[36] com *Drive*, *Docs*, *Sheets* o *Slides*. Totes elles ajudaran a poder redactar la documentació del projecte i emmagatzemar tot ella en un mateix lloc multiplataforma.

Per altra banda, en ser una aplicació Android, s'utilitzarà l'entorn de desenvolupament Android Studio[26], recomanat pel mateix Google. Amb l'ajut de la seva interfície gràfica, plugins i el IDE en si mateix, facilitarà molt la construcció de la plataforma *Wisebite*.

Encara que el treball final de grau no sigui un treball en equip, sinó individual, això no menysprea l'ús d'un control de versions. En aquest projecte s'utilitzarà Git com a eina, i estarà emmagatzemada als servidors de *GitHub*[30]. Amb aquesta eina serà més fàcil poder gestionar l'evolució de l'aplicació i poder recuperar versions antigues.

Per últim, en aplicar una metodologia àgil s'utilitzarà *Trello*[37] per a la gestió i control del projecte. Amb la capacitat de crear targetes dins d'un llistat i associar-li un pes, facilita en gran mesura la implantació de Scrum en aquest projecte.

9.3. Descripció de les tasques

Una part imprescindible per a la planificació temporal del projecte és realitzar una anàlisi molt detallada de les tasques a realitzar durant les tres fases del treball final de grau. Marcant èmfasi en el fet que aquest projecte estarà sota els ideals de la metodologia àgil, per tant, funcionarà via històries d'usuari i iteracions o sprints del projecte. Tota aquesta informació serà explicada a continuació.

9.3.1. Fase inicial

La primera part del projecte es basa en l'especificació general del que es vol construir. En aquesta fase inicial es detallarà el context, estudi de l'art i l'abast per una banda, la planificació temporal i la descripció de les tasques per altra i per últim un informe sobre la gestió econòmica i sostenibilitat del projecte.

A més a més d'això, seguint la metodologia Scrum, es crearà el *backlog* inicial del projecte. Un backlog ve a ser un llistat d'històries d'usuari que componen la plataforma, on cada història d'usuari és una característica o funcionalitat de l'aplicació totalment independent a la resta. A cada una d'aquestes històries d'usuari se li haurà d'atribuir un pes valorant el cost que tindrà la seva implementació. Un cop definits aquests pesos serà molt més fàcil poder prioritzar les tasques a realitzar. Una història d'usuari es donarà per finalitzada quan compleixi cada un dels criteris d'acceptació que haurà de tenir dins la targeta de cada història d'usuari.

Un cop definit tot això passarem a la següent fase del projecte. Cal tenir cura en realitzar una bona fase inicial, ja que pot condicionar de manera notable el procés del projecte.

9.3.2. Iteracions del projecte

Seguint la metodologia Scrum, s'haurà d'aplicar el desenvolupament incremental i vertical, és a dir, anar implementant les funcionalitats o característiques de la plataforma ordenant-les per prioritat. Per a portar-ho a la pràctica es durà a terme un seguit de cinc sprints o iteracions. A final de cada sprint, tindrem una versió de la plataforma que funcioni i es pugui entregar a un hipotètic client.

Un sprint es compon d'un seguit d'històries d'usuari ponderades. L'objectiu és aconseguir que totes elles compleixin els criteris d'acceptació al final del període de l'sprint.

El primer sprint comença el 13 de març i tindrà una durada de dues setmanes, com cada un dels quatre sprints restants. Seguint a aquest ritme, està previst finalitzar la cinquena i última iteració per al 22 de maig. Inicialment es planificaran les iteracions perquè totes elles ponderin aproximadament el mateix. Tot i així, segons la velocitat que es vegi en el desenvolupament d'aquestes històries, es replantejarà o no en la retrospectiva que sempre es farà a final d'sprint.

Tot i que el 22 de maig encara quedarà un mes per a la lectura del treball final de grau, l'objectiu és finalitzar la implementació per aquelles dates. Així es tindrà suficient temps per dedicar-li a la fase final de projecte.

9.3.3. Fase final

Un cop s'hagi finalitzat les cinc iteracions del projecte es passarà a la fase final d'aquest. En aquesta etapa es redactarà la memòria i la documentació necessària per al projecte. Es disposarà pràcticament d'un mes per a meditar i construir una bona documentació, i preparar la lectura que es tindrà a finals del mes de juny.

9.4. Diagrama de Gantt

Un cop analitzat el calendari del qual disposem, els recursos que seran utilitzats i la descripció detallada de les tasques a realitzar, és important adjuntar el diagrama de Gantt[38]. Així es podrà disposar d'una visió completa de quina és la planificació inicial del projecte.

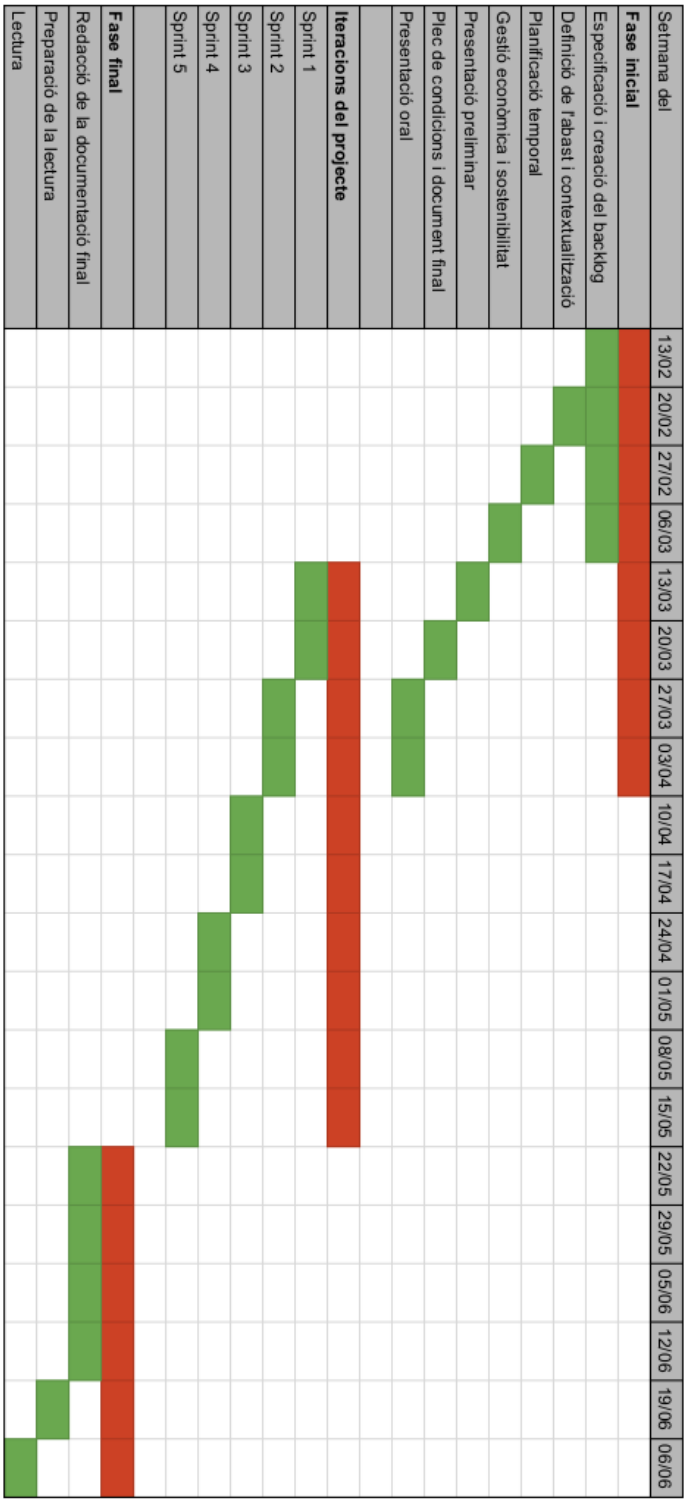

FIGURA 9.1: Diagrama de Gantt

9.5. Valoració d'alternatives i pla d'acció

Durant el desenvolupament del projecte poden sorgir imprevistos que impediran la construcció d'aquest. Pot haver-hi dos tipus de desviacions: mala planificació de temps o imprevistos inesperats o personals.

En primer cas, la mala planificació de temps, pot esdevenir de dues maneres. Per una banda, que hagi estat a l'alça. En aquest cas no hi hauria massa problemàtica, ja que si s'acabessin les històries d'una iteració abans de la data límit, s'afegirà la següent història d'usuari del backlog amb esperances de poder finalitzar dins l'sprint. En canvi, per altra banda, pot sorgir que s'hagi estimat a la baixa. En aquest cas, s'intentarà aplicar unes hores extres en el desenvolupament de les històries pendents o, en cas que sigui impossible, deixar-les per fer i replantejar els següents sprints en la retrospectiva.

En segon lloc, ens podem trobar amb algun imprevist inesperat o bé personal. En casos com aquest, s'analitzarà el cas en especial, ja que pot sorgir qualsevol tipus de problema. Un cop analitzat, se li intentarà donar una solució amb l'objectiu de poder finalitzar de forma correcta la iteració en la qual estiguem situats. En cas que sigui impossible, es tractarà el tema en concret amb el director del projecte.

9.6. Planificació definitiva

En la primera fase del projecte es va realitzar una planificació inicial a on s'especificava quines tasques es realitzarien durant el Treball Final de Grau, com s'organitzarien i, sobretot, quant temps es dedicaria en cada una d'elles. Després d'haver transcorregut i haver finalitzat el projecte, hi ha hagut punts que es van estimar correctament i s'han pogut dur a terme sense cap tipus de problemàtica però d'altres que s'han vist forçats a modificar. Un cop analitzats es veurà si finalment, fent balanç, ha modificat l'objectiu final de projecte.

9.6.1. Acord amb la planificació inicial

En el seu moment es va definir una metodologia de treball inspirada en la metodologia àgil Scrum, amb un seguit de cinc sprints o iteracions a on es desenvoluparia la implementació del projecte. L'objectiu era definir un *backlog* inicial amb totes les *històries d'usuari* que contindria el projecte i, en cada un dels *sprint plannings*, atribuir un conjunt d'aquestes a aquell sprint.

Després d'aquests d'implementar el projecte es pot dir que s'ha seguit aquest patró o metodologia de treball. A més a més, comentar que la durada de les dues setmanes per iteració que es va atribuir a la planificació inicial s'ha respectat amb una variació màxima de 2 o 3 dies.

Tot i així, es va decidir repartir les històries d'usuari segons la disponibilitat de la qual disposava l'alumne per implementar-les durant el període de l'sprint. Malgrat aquesta decisió, el conjunt de les cinc iteracions ha complert amb la planificació inicial.

9.6.2. Modificacions

El canvi més important que s'ha realitzat acord amb la planificació inicial del projecte, però que no modifica el fet d'arribar a l'objectiu final en el temps acordat, és l'alteració en l'ordre de les tasques. En el seu moment es va estipular la idea de realitzar durant dos mesos tota la implementació del projecte i, en l'últim més, redactar la memòria documentant tota la implementació realitzada durant la fase intermèdia.

El canvi ha estat degut per un seguit de factors. Tant el director del projecte com l'alumne que el porta a càrrec es van adonar que realitzar l'ordre de tasques estipulat inicialment portaria a la problemàtica de no tenir una bona documentació final a entregar. En primer lloc pel poc temps que es disposava en comparació del que es necessita per realitzar una bona memòria, i per altra banda el fet de documentar tot al final podria donar lloc a no realitzar una documentació de la implantació del tot acurada donat el temps entre finalitzar el codi i explicar-lo. A més a més, el fet d'anar intercalant implementació amb documentació faria més amè el transcurs del projecte.

Per altra banda, hi ha hagut modificacions en la implementació de les *històries d'usuari*. Inicialment, en realitzar i definir *backlog* inicial, es van crear un seguit de *features* sense contemplar del tot el temps que es disposava per implementar-les. Llavors, durant la realització de la iteració, l'alumne es va adonar que aquestes històries d'usuari aportaven un esforç extra en l'sprint que no es veia reflectit en el resultat final. És a dir, l'esforç que es necessitava per implementar aquestes històries d'usuari no era equivalent al valor que aportava a l'usuari final del producte. En conseqüència, per tal de solucionar aquesta problemàtica, es va decidir presidir d'aquestes *features* donat el temps acotat del projecte.

9.6.3. Conclusions

Finalment, es pot considerar que tot i les modificacions en comparació de la planificació inicial, no hi ha hagut cap problemàtica en arribar de forma correcta al *deadline* del projecte. Ja amb el projecte finalitzat, s'ha implementat les tres fases del treball, corresponents als tres punts que definien l'abast del projecte: gestió interna de l'establiment, anàlisi del restaurant i interacció del client.

Paral·lelament s'ha documentat la memòria dins el temps establert i ha donat temps a una revisió detallada d'aquesta.

Per fer-se una idea de com ha anat l'evolució tant del codi de l'aplicació com el de la memòria, s'adjunta a continuació dues gràfiques proporcionades per *GitHub* que mostren el temps de dedicació a cada una de les dues tasques: codi i memòria.

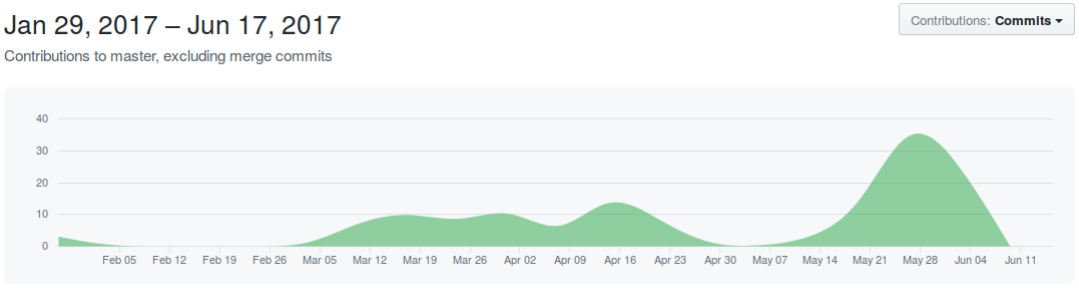

FIGURA 9.2: Contribucions en el codi de l'aplicació

FIGURA 9.3: Contribucions en la memòria

Capítol 10

Gestió econòmica

En aquest apartat es tractarà el cost econòmic que causarà la construcció d'aquest projecte. El treball final de grau és un projecte no remunerat, és a dir, que l'estudiant no rebrà cap recompensa econòmica per a la realització d'aquest. Tot i així, suposarem l'existència d'un equip de treball compost per un cap de projecte, un grup d'analistes i dissenyadors i un conjunt de programadors i provadors. Per així simular quin cost tindria un projecte d'aquest tipus en cas de ser part d'una *startup*.

10.1. Identificació i estimació de costos

Per a la construcció del pressupost[39] d'aquest projecte es necessita tenir en compte els costos directes, que representen als recursos humans, els costos indirectes, que es reflecteixen als recursos materials, i a les contingències i imprevistos. Tots ells, després de la seva anàlisi, esdevindran al pressupost final d'aquest projecte.

10.1.1. Costos directes

Els costos directes són el conjunt de despeses que corresponen al grup de recursos humans que necessita el projecte. Tot i que aquest projecte, com s'ha comentat abans, no té cap tipus de remuneració, es fa només amb afany acadèmic i és només individual, suposarem l'existència d'un equip de treball qualificat econòmicament. Per apropar-nos més al mercat s'ha assignat un responsable d'equip a cada una de les tasques reflectides al diagrama de *Gantt*. S'ha estimat que el cap de projecte tindrà un salari al voltant dels 45€/hora, el grup d'analistes i dissenyadors 35€/hora i l'equip de programadors i provadors de 20€/hora.

Aleshores, com a resultant d'aquesta anàlisi sorgeix la taula següent amb la quantitat d'hores, cost unitari i total de cada una de les activitats del projecte.

ACTIVITAT	HORES	COST UNITARI	COST TOTAL
Recerca i plantejament de la idea	15h	45€/hora	675€
Fase inicial	75h	45€/hora	3.375€
Sprint 1: disseny	10h	35€/hora	350€
Sprint 1: implementació	30h	20€/hora	600€
Sprint 2: disseny	10h	35€/hora	350€
Sprint 2: implementació	30h	20€/hora	600€
Sprint 3: disseny	10h	35€/hora	350€
Sprint 3: implementació	30h	20€/hora	600€
Sprint 4: disseny	10h	35€/hora	350€
Sprint 4: implementació	30h	20€/hora	600€
Sprint 5: disseny	10h	35€/hora	350€
Sprint 5: implementació	30h	20€/hora	600€
Fase final	50h	45€/hora	2.250€
TOTAL	340h	—	11.050€

TAULA 10.1: Costos directes

Les hores calculades per activitat s'han calculat amb el supòsit que ha d'haver-hi un treball, en mitjana, de 30 hores setmanals dedicades al projecte. Finalment, hi haurà una dedicació humana de 340 hores, amb un cost de 11.050€ considerant els supòsits anteriors.

10.1.2. Costos indirectes

Durant tot el projecte apareixeran situacions quotidianes que repercutiran en el pressupost final amb connotativa indirecte com els que s'exposen a continuació.

En primer terme, s'ha de valorar la connexió a Internet com a cost indirecte. El seu cost mensual és de 33€, no obstant s'ha de tenir en compte que no s'utilitzarà el 100% del seu ús en tasques relacionades amb el treball final de grau. Per això se li atribueix un 30% de dedicació.

En segon lloc, s'ha de valorar el hardware utilitzat. En aquest projecte només s'utilitzarà un portàtil per al desenvolupament i construcció del projecte i, addicionalment, un telèfon mòbil amb sistema operatiu Android per a realitzar les proves pertinents segons vagi evolucionant el producte. El cost del portàtil se situa als 500€ juntament amb els 310€ del telèfon mòbil. No obstant això, es considera una vida útil de 4 anys per al portàtil i de 2 anys i mig per al mòbil. Per tant, se li atribueix un 10% i un 17% de dedicació a cada un dels dispositius tecnològics respectivament.

Per últim no oblidar-nos de les impressions en paper que es realitzaran durant el projecte. Tant durant el desenvolupament d'aquest com en la memòria final s'imprimirà documentació. S'ha valorat que s'utilitzaran 500 impressions durant tot el projecte a 0.05€ la unitat.

ACTIVITAT	COST	DEDICACIÓ	COST TOTAL
Internet	33€	30%	10€
Hardware: Portàtil	500€	10%	50€
Hardware: Telèfon mòbil	310€	17%	53€
Impressions	25€	100%	25€
TOTAL	—	—	138€

TAULA 10.2: Costos indirectes

10.1.3. Contingències

En tot pressupost sempre s'ha de reservar un 15% de contingències, que s'aplicaran tant en els costos directes com indirectes.

ACTIVITAT	COST	PERCENTATGE	COST TOTAL
Costos directes	11.050€	15%	12.708€
Costos indirectes	138€	15%	159€
TOTAL	—	—	12.867€

TAULA 10.3: Contingències

10.1.4. Imprevistos

Durant el transcurs del desenvolupament del projecte és possible que apareguin imprevistos que afectin el pressupost.

El primer es considera la mala planificació en temps del projecte. Pot donar-se el cas en què algunes setmanes s'hagi d'invertir més temps en la programació i desenvolupament de la plataforma per complir els terminis establerts en la planificació inicial. S'ha valorat afegir unes 30 hores extres durant tot el projecte, les quals seran atribuïdes al paper de programador o provador, recordant el seu salari de 20€/hora. Per tant, es valora aquest cost addicional amb 600€.

L'altre possible imprevist que es pot trobar el projecte és alguna incidència amb el hardware utilitzat, és a dir, l'ordinador portàtil o el telèfon mòbil. S'ha valorat en 150€ la possible reparació davant d'un presumpte problema que impedeixi el desenvolupament d'aquest projecte.

ACTIVITAT	COST
Hores extra	600€
Incidència amb el hardware	150€
TOTAL	750€

TAULA 10.4: Imprevistos

10.1.5. Pressupost final

Per a la realització del pressupost final no s'ha tingut en compte cap tipus de variació en els costos, com podria ser l'augment de salaris, donat que el projecte és de curta durada. Per altra banda no s'ha tingut en compte cap tipus d'amortització o

marge de benefici donat que és un projecte de caràcter acadèmic sense ànim de lucre ni guanyar diners amb el resultat d'aquest.

ACTIVITAT	**COST**
Costos directes	11.050€
Costos indirectes	138€
Continències	1.679€
Imprevistos	750€
TOTAL	13.617€

TAULA 10.5: Pressupost final

10.2. Control de gestió

Per a portar un bon control de si el pressupost és el correcte i tot va segons hauria d'anar es durà a terme un seguit d'accions per portar-ho tot al dia, i que s'expliquen a continuació.

El primer consisteix en el registre d'accions relacionades amb el projecte, sigui programació d'aquest, reunions o redacció de la documentació. Es mantindrà una taula amb un registre de les hores empleades en el projecte detallant en quin àmbit sestà treballant. Aquestes dades es revisaran de forma periòdica, sobretot en les retrospectives de les iteracions, per veure si el temps dedicat en cada una de les tasques és el previst anteriorment.

Aleshores al final del projecte s'agafarà aquestes dades i es farà una anàlisi comparantho amb el pressupost inicial establert, calculant els costos directes i indirectes reals i els imprevistos que hi ha hagut.

Si es veu que hi ha una desviació notable comparant la realitat amb el pressupost planejat, es prendrà cura en veure quin ha estat el punt on s'ha fallat en la planificació i aprendre de l'error.

Capítol 11

Sostenibilitat i compromís social

En tot projecte, inclòs el de final de grau, es disposa d'una valoració en el que respecta l'economia, la sociabilitat i el medi ambient del projecte. Cada un d'aquests caràcters tindrà associada una valoració, que és explicada posteriorment, que totes juntes esdevindran a la matriu de sostenibilitat[40], adjuntada al final del capítol.

11.1. Sostenibilitat econòmica

En aquest projecte s'ha realitzat un pressupost on inclou una bona avaluació de costos, incloent-hi recursos materials i humans repartits proporcionalment per cada una de les tasques representades al diagrama de *Gantt*. En aquest pressupost se li ha afegit també el cost per ajustaments, actualitzacions i reparacions possibles durant la vida del projecte, cosa que el converteix en competitiu en un escenari de producte real.

Tot i que el pressupost ha acabat tenint un valor no massa alt, es pot obtenir un menor cost retallant, filant prim o col·laborant amb altres entitats acadèmiques o professionals. No obstant això, ens exposem als riscos que això comporta.

Aleshores, després d'haver realitzat l'estudi econòmic del projecte i haver obtingut un pressupost resultant d'aquest, es pot valorar aquest projecte com notablement bo econòmicament. Finalment ha sorgit un cost, comptant cada un dels detalls, prou baix pels objectius que es vol complir.

En conseqüència, el caràcter econòmic del projecte té una valoració de 7.

11.2. Sostenibilitat social

Com s'ha comentat anteriorment estem en una situació a on, tot i disposar d'alta tecnologia, el món de la restauració no s'acaba de modernitzar. I gran culpa d'això ho té l'elevat cost d'implantar aquesta tecnologia en aquest tipus d'establiments donada la personalització que necessiten. És per això que l'aparició de *Wisebite* pot solucionar aquesta problemàtica.

Un dels objectius primordials d'aquest projecte és realitzar un canvi notable en el caràcter social de les persones que treballen o tenen alguna relació amb el món gastronòmic o de la restauració. És per això que, després d'una bona anàlisi, s'extreu com a conclusió que és un punt molt important en el desenvolupament d'aquesta plataforma, ja que el producte final pot esdevenir a molts canvis en el dia a dia de

les persones relacionades amb aquest món. No obstant això, cal destacar que no existeix una necessitat alta en l'existència d'aquest tipus de sistemes. Tot i així, l'aparició de *Wisebite* pot capgirar la situació i oferir un servei que qualsevol establiment de restauració, sense excepció, pot utilitzar.

En conseqüència, el caràcter social rep una valoració de 9.

11.3. Sostenibilitat ambiental

Aquest projecte no contempla el caràcter ambiental de la matriu de sostenibilitat, ja que no és el seu objectiu. Per tant converteix l'estimació d'aquest punt en una tasca complicada.

No obstant això, cal destacar que el poc ús de material contaminant fa que la petjada ecològica no augmenti. Tot i que el projecte no té cap incentiu ni motivació en voler-la reduir. Per tant, ni s'inverteix temps ni recursos en voler-la disminuir o augmentar.

Per tant, se li atribueix una valoració de 7.

11.4. Taula de sostenibilitat

Així doncs, després d'haver analitzat cada un dels tres punts que descriuen l'anàlisi de sostenibilitat del projecte, s'adjunta a continuació la matriu de sostenibilitat que defineix *Wisebite* des d'aquest aspecte.

CARÀCTER	**VALORACIÓ**
Econòmic	7
Social	9
Ambiental	7
TOTAL	23

TAULA 11.1: Taula de sostenibilitat

Capítol 12

Conclusions

Després d'haver explicat tot respecte al projecte de *Wisebite*, començant per la formulació problema, seguint pel disseny d'aquest, prosseguint amb la implementació i definint els costos temporals, econòmics i sostenibles de l'aplicació, arribem a la finalització del treball final de grau. En aquest últim capítol es detallarà el resultat obtingut, s'analitzarà en una retrospectiva i s'especularà en quin serà el futur de *Wisebite*.

12.1. Resultat

En la fase inicial es va detallar un seguit de funcionalitats que havia de tenir l'aplicació i que es tenia com a objectiu per a la fase final d'aquest. Ara com ara, i ja amb l'aplicació implementada, es pot afirmar que s'ha arribat al final amb tots els aspectes de l'abast complet.

L'aplicació *Wisebite*, que està disponible per tot el món al *Google Play*[41], compleix amb els objectius preestablerts i podria ser utilitzada per qualsevol establiment de restauració. Tot i així, es considera que un projecte mai té data de finalització, és a dir, que sempre se'l pot nodrir amb més funcionalitats. Malgrat aquest aspecte, ara com ara, *Wisebite* disposa de les següents funcionalitats:

- Crear un restaurant amb tota la seva informació bàsica, horaris d'apertura durant la setmana i els seus respectius plats i menús.
- Crear comandes, a on s'especifica el nombre de taula del restaurant.
- Mostra les comandes actives, considerant dues de les perspectives que disposa l'aplicació, és a dir, cambrer i cuiner.
- Veure els detalls del seu restaurant, podent consultar tota la informació un desitja.
- Afegir un altre usuari al seu restaurant.
- Estudi de les estadístiques del teu restaurant amb l'ajuda d'algunes gràfiques i dades d'utilitat per a poder realitzar una anàlisi d'aquest i millorar la qualitat de l'establiment.
- Llistar la resta dels restaurants en l'aplicació, per així actuar com a client.
- Crear una comanda a un restaurant aliè i veure el seu estat en temps real, visualitzant quan està preparat, entregat i pagat.

- Valoració dels plats i menús i del restaurant en si després de la creació de la comanda en qüestió.
- Veure els comentaris i les puntuacions d'altres usuaris sobre tots els restaurants i els seus respectius plats i menús.
- Personalitzar a gust de l'usuari el seu perfil editant la informació bàsica i la imatge de perfil.

Així doncs, es pot considerar que la plataforma compleix amb els tres punts que es va indicar a l'abast del projecte. L'aplicació pot gestionar internament l'establiment, així eliminant el bolígraf i el paper. La plataforma permet analitzar les dades emmagatzemades i treure-li un profit. I a més a més, com a client de qualsevol establiment de restauració que implementi *Wisebite* pot interactuar amb ell realitzant comanda, seguint-la en temps real i valorar-la, així contribuint a la comunitat.

12.2. Retrospectiva

Un cop acabat el projecte, m'agradaria ficar èmfasi en un conjunt de punts pel qual recordaré aquest treball amb bon gust de boca.

En primer lloc, remarcar el fet que aquest treball ha estat un projecte personal que se'm va ocórrer durant l'estiu passat. Tot i tenir l'opció de realitzar el treball final de grau en l'empresa que estava treballant, i que encara hi pertanyo, i permetrem anar més relaxat durant el quadrimestre, faig decidir llançar-me a la piscina i atrevir-me amb el projecte de *Wisebite*. Ara, si ho miro amb retrospectiva, tot l'esforç dedicat en ell ha contribuït de manera positiva en molts aspectes en la meva carrera personal i professional.

Per altra banda, m'agradaria ressaltar la metodologia que s'ha posat en pràctica en el projecte i el disseny de software que ha estat implementat. Un dels fets que més m'agrada personalment de l'especialitat elegida són justament els dos mencionats, i poder-los utilitzar ambdós en la finalització del meu grau és quelcom d'agrair. Tant és el gust que tinc per aquests dos aspectes que potser finalment m'acabo especialitzant en ells en un hipotètic master a cursar.

Així doncs, després d'aquests quatre mesos d'esforç en poder tenir *Wisebite* complet i haver satisfet els objectius marcats a inicis de quadrimestre han valgut molt la pena. Acabo el grau universitari amb la satisfacció d'haver fet bé la feina.

12.3. Futur

Durant el transcurs d'aquests quatre mesos en els quals he estat dedicant-me a la construcció de *Wisebite*, companys de la universitat em preguntaven quina era la temàtica del meu treball final de grau, i un cop els hi explicava gran part d'aquest conjunt es sorprenia i em preguntava que si tenia intenció de llançar-ho com a *startup* en finalitzar-lo.

Aquest aspecte mai m'ho havia plantejat d'un bon inici, i de fet encara ho poso en dubte si seria possible. Però el fet que algun professor de la facultat em comentés

que la idea era molt bona per tirar-la endavant com a empresa, em va motivar bastant en aquest aspecte.

Així doncs, tot i encara no ser una idea molt segura, es té la intenció de millorar el projecte durant aquest estiu i optimitzar-lo per tenir-ho llest a principis de setembre. En aquell moment, en cas de disposar d'un *Wisebite* llest per ser llançat al mercat, es buscarà com provar-ho en restaurants de confiança per veure la reacció dels membres de l'establiment. I ja doncs, qui sap que ens proporcionarà el futur.

12.4. Aprenentatge final

Com a conclusió, m'agradaria emfatitzar en el concepte més important que he après després d'haver finalitzat aquest treball final de grau, i en conseqüència, el grau en enginyeria informàtica. Un professor dels que m'he anat trobant durant el transcurs del grau un dia em va comentar un fet que em va marcar en la meva manera de pensar. Em va fer reflexionar sobre el paper que té un enginyer informàtic a la societat.

Aquests últims anys la tecnologia ha evolucionat de tal manera que té el poder de generar necessitats a la societat, és a dir, té la capacitat de canviar el comportament de l'ésser humà en el seu dia quotidià. I qui ho aconsegueix això? Els enginyers informàtics que llancen a la llum aquests projectes que fan canviar la manera de veure el món.

Així doncs, amb un possible èxit de *Wisebite* estaríem canviant totalment el món de la restauració. I tot això amb una simple plataforma. Ja que, tot i que no ho pensem, els informàtics tenim el poder de canviar el món.

FI.

Índex de figures

Índex de taules

Bibliografia

[1] TheSnugg.com. *A Brief History of Smartphones*. 2014. URL: http://www.thesnugg.com/a-brief-history-of-smartphones.aspx (cons. 23-2-2017).

[2] Waiterio. *Waiterio POS Restaurant POS Bar*. 2017. URL: https://www.waiterio.com/ (cons. 24-2-2017).

[3] Prime Tray. *Prime Tray*. 2017. URL: http://www.primetray.in/ (cons. 23-2-2017).

[4] OrderSev. *OrderServ Kitchen Order Management and EPOS System*. 2017. URL: http://www.orderserv.com (cons. 24-2-2017).

[5] Tablet Waiter. *Tablet Waiter - Android Apps on Google Play*. 2016. URL: https://goo.gl/E5hLbe (cons. 24-2-2017).

[6] Google. *Material Design para Android*. 2017. URL: https://developer.android.com/design/material/index.html?hl=es (cons. 16-5-2017).

[7] Cloud Waiter. *Cloud Waiter*. 2017. URL: http://www.cloudwaiter.es (cons. 24-2-2017).

[8] FastOrder. *FastOrder*. 2017. URL: https://fastorder-app.com/ (cons. 24-2-2017).

[9] WardCunningham. *Camel Case*. 2014. URL: http://wiki.c2.com/?CamelCase (cons. 18-5-2017).

[10] Oracle. *javadoc*. 2016. URL: http://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html (cons. 18-5-2017).

[11] BusinessDictionary.com. *What is a stakeholder? Definition and meaning*. 2017. URL: http://www.businessdictionary.com/definition/stakeholder.html (cons. 23-2-2017).

[12] Ernest Teniente. *Ernest Teniente's home page*. 2017. URL: http://www.essi.upc.edu/~teniente/ (cons. 23-2-2017).

[13] Wikipedia. *Requisito (sistemas)*. 2017. URL: https://es.wikipedia.org/wiki/Requisito_(sistemas) (cons. 10-6-2017).

[14] Atlantic Systems Guild Ltd. *Volere requirements Home page*. 2017. URL: http://www.volere.co.uk/ (cons. 10-6-2017).

[15] W3C. *World Wide Web Consortium (W3C)*. 2017. URL: http://www.w3c.es/ (cons. 10-6-2017).

[16] ISO. *ISO 8601 Date and time format*. 2017. URL: https://www.iso.org/iso-8601-date-and-time-format.html (cons. 10-6-2017).

[17] ISO. *ISO/IEC 17799:2005 - Information technology – Security techniques – Code of practice for information security management*. 2005. URL: https://www.iso.org/standard/39612.html (cons. 10-6-2017).

[18] Wikipedia. *Diagrama de clases*. 2017. URL: https://es.wikipedia.org/wiki/Diagrama_de_clases (cons. 14-6-2017).

[19] Statista. *Android version market share 2017*. 2017. URL: https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/ (cons. 15-6-2017).

[20] JSON.org. *JSON*. 2017. URL: http://www.json.org/json-es.html (cons. 15-6-2017).

[21] Cecilio Álvarez. *Java, el lenguaje más usado y su evolución*. 2015. URL: https://www.genbetadev.com/java-j2ee/java-el-lenguaje-mas-usado-y-su-evolucion (cons. 16-6-2017).

[22] Gabriel Ceravolo. *Haxe en Español: Atlas, Xml y Packers*. 2015. URL: http://haxelatam.blogspot.com.es/2015/04/atlas-xml-y-packers-parte-2.html (cons. 16-6-2017).

[23] Sylvain Benner. *LaTeX layer*. 2016. URL: http://spacemacs.org/layers/+lang/latex/README.html (cons. 16-6-2017).

[24] Firebase. *Legacy Website*. 2017. URL: https://www.firebase.com (cons. 28-2-2017).

[25] Wikipedia. *SQLite*. 2017. URL: https://en.wikipedia.org/wiki/SQLite (cons. 17-6-2017).

[26] Android Studio. *Android Studio. The Official IDE for Android*. 2017. URL: https://developer.android.com/studio/index.html (cons. 2-3-2017).

[27] Taringa. *Genymotion en Debian GNU/Linux*. 2016. URL: http://www.taringa.net/posts/linux/18808334/Genymotion-en-Debian-GNU-Linux.html (cons. 17-6-2017).

[28] TeXstudio.org. *TeXstudio*. 2017. URL: http://www.texstudio.org/ (cons. 17-6-2017).

[29] SCM. *Git*. 2017. URL: https://git-scm.com/ (cons. 17-6-2017).

[30] GitHub. *Wisebite*. 2017. URL: https://github.com/Wisebite (cons. 2-3-2017).

[31] Google Play. *Servicios de Google Play*. 2017. URL: https://play.google.com/store/apps/details?id=com.google.android.gms&hl=es (cons. 17-6-2017).

[32] Clans. *Clans/FloatingActionButton: Android Floating Action Button based on Material Design specification*. 2017. URL: https://github.com/Clans/FloatingActionButton (cons. 17-6-2017).

[33] Square. *Picasso*. 2017. URL: http://square.github.io/picasso/ (cons. 17-6-2017).

[34] PhilJay. *PhilJay/MPAndroidChart: A powerful Android chart view / graph view library, supporting line- bar- pie- radar- bubble- and candlestick charts as well as scaling, dragging and animations.* 2017. URL: https://github.com/PhilJay/MPAndroidChart (cons. 17-6-2017).

[35] The Project Lombok Authors. *Project Lombok*. 2017. URL: https://projectlombok.org/ (cons. 17-6-2017).

[36] GSuite. *G Suite – Gmail, Drive, Docs and More*. 2017. URL: https://gsuite.google.com/ (cons. 2-3-2017).

[37] Trello. *Wisebite*. 2017. URL: https://trello.com/wisebite (cons. 2-3-2017).

[38] Wikipedia. *Diagrama de Gantt*. 2017. URL: https://es.wikipedia.org/wiki/Diagrama_de_Gantt (cons. 18-6-2017).

[39] Infoautónomos. *Cómo Hacer Un Presupuesto Paso a Paso: Elementos Y Recomendaciones Básicas*. 2015. URL: http://infoautonomos.eleconomista.es/marketing-y-ventas/como-hacer-un-presupuesto/ (cons. 9-3-2017).

[40] OAS.org. *6.1 Evaluación Del Impacto Ambiental, Económico Y Social*. 2015. URL: http://www.oas.org/dsd/publications/unit/oea49s/ch29.htm (cons. 11-3-2017).

[41] ModernNet. *Wisebite - Aplicaciones de Android en Google Play*. 2017. URL: https://play.google.com/store/apps/details?id=dev.wisebite.wisebite&hl=es (cons. 18-6-2017).

Apèndix A

Patrons de disseny

A.1. Interfície Entity

```
1  package dev.wisebite.wisebite.utils;
2
3  import java.io.Serializable;
4
5  public interface Entity extends Serializable {
6
7      String getId();
8
9      void setId(String id);
10
11 }
```

A.2. Interfície Repository

```
1  package dev.wisebite.wisebite.firebase;
2
3  import com.google.firebase.database.ChildEventListener;
4  import com.google.firebase.database.DataSnapshot;
5
6  import java.util.List;
7
8  import dev.wisebite.wisebite.utils.Entity;
9
10 /**
11  * Created by albert on 13/03/17.
12  * @author albert
13  */
14 public abstract class Repository<T extends Entity> implements
       ChildEventListener {
15
16     private OnChangedListener listener;
17
18     /**
19      * Just for the Firebase when needed
20      */
21     public Repository () {
22
23     }
24
25     public interface OnChangedListener {
26         enum EventType {Added, Changed, Removed, Moved, Full}
27
28         void onChanged(EventType type);
29     }
30
31     public void setOnChangedListener(OnChangedListener
           listener) {
32         this.listener = listener;
33     }
34
35     public abstract T insert(T item);
36
37     public abstract T insertInternal(T item);
38
39     public abstract T update(T item);
40
41     public abstract T updateInternal(T item);
42
43     public abstract void delete(String id);
44
45     public abstract void deleteInternal(String id);
46
```

```
47      public abstract boolean exists(String id);
48
49      public abstract T get(String id);
50
51      public abstract List<T> all();
52
53      protected void notifyChange(OnChangedListener.EventType
            type) {
54          if (listener != null){
55              listener.onChanged(type);
56          }
57      }
58
59      @Override
60      public void onChildAdded(DataSnapshot dataSnapshot,
            String s) {
61          insertInternal(convert(dataSnapshot));
62          notifyChange(OnChangedListener.EventType.Added);
63      }
64
65      @Override
66      public void onChildChanged(DataSnapshot dataSnapshot,
            String s) {
67          updateInternal(convert(dataSnapshot));
68          notifyChange(OnChangedListener.EventType.Changed);
69      }
70
71      @Override
72      public void onChildRemoved(DataSnapshot dataSnapshot) {
73          deleteInternal(convert(dataSnapshot).getId());
74          notifyChange(OnChangedListener.EventType.Removed);
75      }
76
77      @Override
78      public void onChildMoved(DataSnapshot dataSnapshot,
            String s) {
79          updateInternal(convert(dataSnapshot));
80          notifyChange(OnChangedListener.EventType.Moved);
81      }
82
83      protected abstract T convert(DataSnapshot data);
84
85  }
```

A.3. Classe d'exemple Repository

```
1  package dev.wisebite.wisebite.repository;
2
3  import android.content.Context;
4
5  import com.google.firebase.database.DataSnapshot;
6
7  import java.util.LinkedHashMap;
8  import java.util.Map;
9
10 import dev.wisebite.wisebite.domain.Restaurant;
11 import dev.wisebite.wisebite.firebase.FirebaseRepository;
12
13 /**
14  * Created by albert on 13/03/17.
15  * @author albert
16  */
17 public class RestaurantRepository extends FirebaseRepository<
       Restaurant> {
18
19     public static final String OBJECT_REFERENCE = "restaurant
           ";
20     public static final String NAME_REFERENCE = "name";
21     public static final String LOCATION_REFERENCE = "location
           ";
22     public static final String PHONE_REFERENCE = "phone";
23     public static final String DESCRIPTION_REFERENCE = "
           description";
24     public static final String WEBSITE_REFERENCE = "website";
25     public static final String NUMBER_OF_TABLES_REFERENCE = "
           numberOfTables";
26
27     public static final String OPEN_TIMES_REFERENCE = "
           openTimes";
28     public static final String MENUS_REFERENCE = "menus";
29     public static final String DISHES_REFERENCE = "dishes";
30     public static final String USERS_REFERENCE = "users";
31     public static final String EXTERNAL_ORDERS_REFERENCE = "
           externalOrders";
32     public static final String REVIEWS_REFERENCE = "reviews";
33
34     /**
35      * Constructor class
36      * @param context Repository's context
37      */
38     public RestaurantRepository(Context context) {
39         super(context);
40     }
41
```

```
42      @Override
43      protected Restaurant convert(DataSnapshot data) {
44          Restaurant restaurant = new Restaurant();
45          restaurant.setId(data.getKey());
46          for (DataSnapshot d : data.getChildren()) {
47              if (d.getKey().equals(NAME_REFERENCE)) {
48                  restaurant.setName(d.getValue(String.class));
49              } else if (d.getKey().equals(LOCATION_REFERENCE))
                    {
50                  restaurant.setLocation(d.getValue(String.
                        class));
51              } else if (d.getKey().equals(PHONE_REFERENCE)) {
52                  restaurant.setPhone(d.getValue(Integer.class)
                        );
53              } else if (d.getKey().equals(
                    DESCRIPTION_REFERENCE)) {
54                  restaurant.setDescription(d.getValue(String.
                        class));
55              } else if (d.getKey().equals(WEBSITE_REFERENCE))
                    {
56                  restaurant.setWebsite(d.getValue(String.class
                        ));
57              } else if (d.getKey().equals(
                    NUMBER_OF_TABLES_REFERENCE)) {
58                  restaurant.setNumberOfTables(d.getValue(
                        Integer.class));
59              } else if (d.getKey().equals(OPEN_TIMES_REFERENCE
                    )) {
60                  Map<String, Object> openTimes = new
                        LinkedHashMap<>();
61                  for (DataSnapshot openTime : d.getChildren())
                        {
62                      openTimes.put(openTime.getKey(), true);
63                  }
64                  restaurant.setOpenTimes(openTimes);
65              } else if (d.getKey().equals(MENUS_REFERENCE)) {
66                  Map<String, Object> menus = new LinkedHashMap
                        <>();
67                  for (DataSnapshot menu : d.getChildren()) {
68                      menus.put(menu.getKey(), true);
69                  }
70                  restaurant.setMenus(menus);
71              } else if (d.getKey().equals(DISHES_REFERENCE)) {
72                  Map<String, Object> dishes = new
                        LinkedHashMap<>();
73                  for (DataSnapshot dish : d.getChildren()) {
74                      dishes.put(dish.getKey(), true);
75                  }
76                  restaurant.setDishes(dishes);
77              } else if (d.getKey().equals(USERS_REFERENCE)) {
```

```
78                  Map<String, Object> users = new LinkedHashMap
                        <>();
79                  for (DataSnapshot dish : d.getChildren()) {
80                      users.put(dish.getKey(), true);
81                  }
82                  restaurant.setUsers(users);
83              } else if (d.getKey().equals(
                    EXTERNAL_ORDERS_REFERENCE)) {
84                  Map<String, Object> externalOrders = new
                        LinkedHashMap<>();
85                  for (DataSnapshot dish : d.getChildren()) {
86                      externalOrders.put(dish.getKey(), true);
87                  }
88                  restaurant.setExternalOrders(externalOrders);
89              } else if (d.getKey().equals(REVIEWS_REFERENCE))
                    {
90                  Map<String, Object> reviews = new
                        LinkedHashMap<>();
91                  for (DataSnapshot review : d.getChildren()) {
92                      reviews.put(review.getKey(), true);
93                  }
94                  restaurant.setReviews(reviews);
95              }
96          }
97          return restaurant;
98      }
99
100     @Override
101     public String getObjectReference() {
102         return OBJECT_REFERENCE;
103     }
104
105 }
```

A.4. Classe abstracta FirebaseRepository

```
1  package dev.wisebite.wisebite.firebase;
2
3  import android.content.Context;
4  import android.util.Log;
5
6  import com.google.firebase.database.DataSnapshot;
7  import com.google.firebase.database.DatabaseError;
8  import com.google.firebase.database.DatabaseReference;
9  import com.google.firebase.database.FirebaseDatabase;
10 import com.google.firebase.database.ValueEventListener;
11
12 import java.util.ArrayList;
13 import java.util.HashMap;
14 import java.util.LinkedHashMap;
15 import java.util.List;
16
17 import dev.wisebite.wisebite.utils.Entity;
18
19 /**
20  * Created by albert on 13/03/17.
21  * @author albert
22  */
23 public abstract class FirebaseRepository<T extends Entity>
       extends Repository<T> {
24
25     public static final String FIREBASE_URI = "https://
           wisebite-f7a53.firebaseio.com/";
26     private static final String TAG = FirebaseRepository.
           class.getSimpleName();
27     private final HashMap<String, T> map;
28     protected FirebaseDatabase firebase;
29     protected DatabaseReference database;
30
31     /**
32      * Constructor class
33      * @param context Repository's context
34      */
35     public FirebaseRepository(@SuppressWarnings("
           UnusedParameters") Context context) {
36         firebase = FirebaseDatabase.getInstance(FIREBASE_URI)
               ;
37         database = firebase.getReference().child(
               getObjectReference());
38         map = new LinkedHashMap<>();
39
40         database.addValueEventListener(new ValueEventListener
               () {
41             @Override
```

```
42              public void onDataChange(DataSnapshot
                    dataSnapshot) {
43                  for (DataSnapshot child : dataSnapshot.
                        getChildren()) {
44                      insertInternal(convert(child));
45                  }
46                  notifyChange(OnChangedListener.EventType.Full
                        );
47              }
48              @Override
49              public void onCancelled(DatabaseError
                    firebaseError) {
50
51              }
52          });
53      }
54
55      /**
56       * Insert some object in specific repository with random
             id.
57       * @param item Object that you want to insert.
58       * @return Inserted item
59       */
60      @Override
61      public T insert(T item) {
62          DatabaseReference ref = database.push();
63          ref.setValue(item);
64          item.setId(ref.getKey());
65          map.put(ref.getKey(), item);
66          return item;
67      }
68
69      /**
70       * Insert some object in specific repository with
             specific id.
71       * @param t   Object that you want to insert.
72       * @param key Key that new object will have.
73       * @return Inserted item
74       */
75      private T insertWithId(T t, String key) {
76          t.setId(key);
77          database.child(key).setValue(t);
78          map.put(key, t);
79          return t;
80      }
81
82      /**
83       * Delete some object of this repository by id.
84       * @param id Object key that you want to delete.
85       */
86      public void delete(String id) {
```

```
87          database.child(id).removeValue();
88          map.remove(id);
89      }
90
91      /**
92       * Update specific item of repository
93       * @param item item that you want to update
94       * @return Inserted item
95       */
96      public T update(T item) {
97          delete(item.getId());
98          return insertWithId(item, item.getId());
99      }
100
101     /**
102      * Get object reference URI in Firebase.
103      * @return Respective reference URI of object.
104      */
105     public abstract String getObjectReference();
106
107     /**
108      * Get specific item of this repository
109      * @param id key that identifies the item
110      * @return Item that you want to get
111      */
112     public T get(String id) {
113         return map.get(id);
114     }
115
116     /**
117      * Get if an specific item exists
118      * @param id key that identifies the item
119      * @return Item that you want to get
120      */
121     public boolean exists(String id){
122         return map.keySet().contains(id);
123     }
124
125     /**
126      * Get all repository
127      * @return a list that contains all values of this
           repository
128      */
129     public List<T> all() {
130         return new ArrayList<>(map.values());
131     }
132
133     /**
134      * Listener that controls when it is occurred an error
135      * @param firebaseError error
136      */
```

```
137     @Override
138     public void onCancelled(DatabaseError firebaseError) {
139         Log.e(TAG, firebaseError.getMessage(), firebaseError.
               toException());
140     }
141 
142     /**
143      * Insert some object internally
144      * @param item Object that you want to insert
145      * @return Inserted item
146      */
147     @Override
148     public T insertInternal(T item) {
149         map.put(item.getId(), item);
150         return item;
151     }
152 
153     /**
154      * Update some object internally
155      * @param item Object that you want to update
156      * @return Updated item
157      */
158     @Override
159     public T updateInternal(T item) {
160         return insertInternal(item);
161     }
162 
163     /**
164      * Delete some object internally
165      * @param id Object that you want to delete
166      */
167     @Override
168     public void deleteInternal(String id) {
169         map.remove(id);
170     }
171 
172     /**
173      * Convert to object model
174      * @param data Object that you want to convert
175      * @return Modeled item
176      */
177     @Override
178     protected T convert(DataSnapshot data) {
179         return null;
180     }
181 
182 }
```

A.5. Interfície Service

```
1  package dev.wisebite.wisebite.utils;
2
3  import java.util.List;
4
5  import dev.wisebite.wisebite.firebase.Repository;
6
7  /**
8   * Created by albert on 13/03/17.
9   * @author albert
10  */
11 public abstract class Service<T extends Entity> {
12
13     protected final Repository<T> repository;
14
15     public Service (Repository<T> repository){
16         this.repository = repository;
17     }
18
19     public T save(T item) {
20         if (repository.exists(item.getId())) return
               repository.update(item);
21         return repository.insert(item);
22     }
23
24     public T get(String key) {
25         return repository.get(key);
26     }
27
28     @SuppressWarnings("unused")
29     public void delete(String key){
30         repository.delete(key);
31     }
32
33     @SuppressWarnings("unused")
34     public int getAmount(){
35         return repository.all().size();
36     }
37
38     public List<T> getAll() {
39         return repository.all();
40     }
41
42     public void setOnChangedListener(Repository.
          OnChangedListener listener){
43         repository.setOnChangedListener(listener);
44     }
45
46 }
```

A.6. Singleton ServiceFactory

```
1  package dev.wisebite.wisebite.service;
2
3  import android.content.Context;
4
5  import dev.wisebite.wisebite.repository.DishRepository;
6  import dev.wisebite.wisebite.repository.ImageRepository;
7  import dev.wisebite.wisebite.repository.MenuRepository;
8  import dev.wisebite.wisebite.repository.OpenTimeRepository;
9  import dev.wisebite.wisebite.repository.OrderItemRepository;
10 import dev.wisebite.wisebite.repository.OrderRepository;
11 import dev.wisebite.wisebite.repository.RestaurantRepository;
12 import dev.wisebite.wisebite.repository.ReviewRepository;
13 import dev.wisebite.wisebite.repository.UserRepository;
14
15 /**
16  * Created by albert on 13/03/17.
17  * @author albert
18  */
19 public final class ServiceFactory {
20
21     private static DishService dishService;
22     private static ImageService imageService;
23     private static MenuService menuService;
24     private static OpenTimeService openTimeService;
25     private static OrderItemService orderItemService;
26     private static OrderService orderService;
27     private static RestaurantService restaurantService;
28     private static ReviewService reviewService;
29     private static UserService userService;
30
31     public static DishService getDishService(Context context)
           {
32         if (dishService == null)
33             dishService = new DishService(
34                     new DishRepository(context));
35         return dishService;
36     }
37
38     public static ImageService getImageService(Context
           context){
39         if (imageService == null)
40             imageService = new ImageService(
41                     new ImageRepository(context));
42         return imageService;
43     }
44
45     public static MenuService getMenuService(Context context)
           {
```

```
46          if (menuService == null)
47              menuService = new MenuService(
48                      new MenuRepository(context),
49                      new DishRepository(context));
50          return menuService;
51      }
52
53      public static OpenTimeService getOpenTimeService(Context
           context){
54          if (openTimeService == null)
55              openTimeService = new OpenTimeService(
56                      new OpenTimeRepository(context));
57          return openTimeService;
58      }
59
60      public static OrderItemService getOrderItemService(
           Context context){
61          if (orderItemService == null)
62              orderItemService = new OrderItemService(
63                      new OrderItemRepository(context),
64                      new DishRepository(context),
65                      new MenuRepository(context));
66          return orderItemService;
67      }
68
69      public static OrderService getOrderService(Context
           context){
70          if (orderService == null)
71              orderService = new OrderService(
72                      new OrderRepository(context),
73                      new OrderItemRepository(context),
74                      new DishRepository(context),
75                      new MenuRepository(context),
76                      new RestaurantRepository(context),
77                      new UserRepository(context));
78          return orderService;
79      }
80
81      public static RestaurantService getRestaurantService(
           Context context){
82          if (restaurantService == null)
83              restaurantService = new RestaurantService(
84                      new RestaurantRepository(context),
85                      new MenuRepository(context),
86                      new DishRepository(context),
87                      new ImageRepository(context),
88                      new OpenTimeRepository(context),
89                      new OrderRepository(context),
90                      new OrderItemRepository(context),
91                      new UserRepository(context),
92                      new ReviewRepository(context));
```

```
93          return restaurantService;
94      }
95
96      public static ReviewService getReviewService(Context
           context) {
97          if (reviewService == null) {
98              reviewService = new ReviewService(
99                      new ReviewRepository(context),
100                     new RestaurantRepository(context),
101                     new DishRepository(context),
102                     new MenuRepository(context),
103                     new UserRepository(context));
104         }
105         return reviewService;
106     }
107
108     public static UserService getUserService(Context context)
           {
109         if (userService == null)
110             userService = new UserService(
111                     new UserRepository(context),
112                     new ImageRepository(context),
113                     new RestaurantRepository(context),
114                     new OrderRepository(context),
115                     new OrderItemRepository(context));
116         return userService;
117     }
118
119     public static Integer getServiceCount() {
120         return 9;
121     }
122 }
```