NexttPOS Delivery App: Aplicació mòbil per a la restauració

Daniel Muñoz Vidal

Resum— NexttPOS Delivery App és una aplicació mòbil multiplataforma que va dirigida als usuaris del sector de l'hostaleria, en concret dels que fan ús del servei a domicili. L'aplicació està dividida en tres parts diferenciades, la del repartidor, la del client i la del gerent. En la part del repartidor, entre d'altres funcionalitats l'aplicació és capaç de guiar de manera òptima, mitjançant diverses heurístiques, a l'encarregat de fer el repartiment de comandes. En quant a la part del client, la funcionalitat principal consisteix en oferir la possibilitat de fer un seguiment en temps real de la ubicació GPS del repartidor assignat a la comanda. Per últim a la part del gerent, aquest podrà tenir accés a dades estadístiques dels diferents locals i repartidors que disposi. Darrere d'aquesta solució estan principalment Ionic, un framework per a crear aplicacions mòbils híbrides amb Angular, Firebase i SQL Server com a bases de dades i una API REST lleugera en Node.js encarregada d'oferir serveis web per obtenir i tractar dades.

Paraules clau— Servei a domicili, multiplataforma, GPS, Ionic, Angular, Firebase, SQL Server, Node.js.

Abstract— NexttPOS Delivery App is a multiplatform mobile application aimed at users in the food service sector, specifically those who use the delivery service. The application is divided into three different parts, the delivery man, the customer and the manager. In the part of the delivery man, among other functionalities, the application is able to guide optimally, through various heuristics, the person who deliver the orders. For the client part, the main functionality consists of offering the possibility of real-time tracking of the GPS location of the delivery man assigned to the order. Finally, in the part of the manager, he may have accés to statistical data of the diferent restaurants and delivery men. Behind this solution are mainly Ionic, a framework for creating hybrid mobile applications with Angular, Firebase and SQL Server as databases and a lightweight API REST in Node.js responsible for offering web services to obtain and process data.

Index Terms— Delivery, cross platform, GPS, Ionic, Angular, Firebase, SQL Server, Node.js.

1 Introducció

Aquest projecte forma part d'una proposta realitzada per l'empresa Next & Tech SL, en endavant Nextt.

La proposta neix a partir del creixent de demanda que estem vivint en quant al servei de menjar a domicili, i cada cop són més els negocis que decideixen oferir un servei d'aquest tipus. Next disposa d'un extens llistat de clients en l'àmbit de l'hostaleria i en les últimes dates nombrosos clients s'han posat en contacte amb l'empresa demanant solucions relacionades amb

Per tant, amb aquest projecte el que es pretén aconseguir és una solució per a les necessitats dels clients d'aquesta empresa, en concret als que fan servir un procés de "Delivery" en la seva lògica de negoci. Actualment, no hi ha mètodes de control dels lliuraments de comandes. Per aquesta raó és necessària la creació d'una aplicació que a més de facilitar la feina de repartidor, permeti als locals de restauració tenir un control en temps real del seu servei a més d'un mètode capaç d'enregistrar i tractar dades (feedback clients, temps d'entrega, eficiència de treballadors...) per tal de generar estadístiques que permetin millorar el seu servei. A més, molts dels clients tenen la necessitat d'oferir un servei diferent als seus comensals, un servei que els permeti tenir informació total sobre la seva comenda i on puguin deixar una valoració del servei.

- *E-mail de contacte: d.munoz@nextt.es*

- *Menció realitzada: Enginyeria del Software.*
- *Treball tutoritzat per: Lluís Gesa*
- *Curs 2018/19*

2 OBJECTIUS

En aquesta secció definim els objectius principals i secundaris de l'aplicació.

2.1 Objectius principals

L'objectiu principal d'aquest projecte és implementar una aplicació mòbil multiplataforma orientada a la restauració, concretament al servei de servei a domicili. Aquesta aplicació ha de satisfer les necessitats del client:

- Permetre als repartidors tenir un control total de les comandes que han de ser lliurades.
- Mitjançant l'aplicació, el gerent dels locals ha de poder tenir accés a informació d'interès sobre el servei que estan oferint amb aquesta app.
- Els comensals que facin comandes mitjançant l'aplicació han de sentir-se informats en tot moment de l'estat de la seva ordre.

2.2 Objectius secundaris

Aquest són altres del objectius que han d'assolir-se en la realització del projecte:

- Dissenyar una aplicació útil per a un sector en concret (restauració) que pugui ser fàcilment adaptable a altres sectors com el logístic.
- Aplicar els coneixements adquirits al llarg del anys de grau (ús de patrons de disseny, serveis web, gestió de projectes, test ...).
- Analitzar el mercat actual per trobar quins són els punts forts i dèbils d'altres aplicació que puguin tenir alguna funcionalitat en comú.
- Adquirir nous coneixements a l'hora d'implementar software amb noves tecnologies.

3 ESTAT DE L'ART

La creació d'una aplicació dedicada a la gestió del "delivery" en l'àmbit de la restauració ens obre camí en un mercat en el qual l'enfocament d'aquest tipus d'aplicacions sempre ha estat orientat al consumidor final dels nostres clients. D'aquesta manera, podem aprofitar la falta d'aplicacions enfocades als restaurants, als seus treballadors i als seus gerents per a obtenir un bon posicionament en el mercat. Oferint un producte innovador, que pugui guanyar-se el nom, aprofitant la tendència actual en la qual cada dia és més habitual demanar menjar a domicili. Per a descriure millor el context en què es troba NexttPOS Delivery App, és necessari parlar d'algunes de les aplicacions de la competència com Glovo, JustEat, o UberEat, aplicacions que ofereixen un servei de venta de menjar a domicili. Com es pot observar, totes aquestes aplicacions ofereixen un servei una mica diferent del que vol oferir NexttPOS Delivery App. Totes estan orientades al client final dels restaurants i a l'hora de fer el seguiment de la comanda la quantitat de informació de la que es disposa és molt limitada.

4 METODOLOGIA

La metodologia que s'ha fet servir a l'hora de desenvolupar el projecte ha sigut Kanban Personal [1]. La decisió d'escollir aquest mètode de treball àgil ve donada per la necessitat de poder gestionar el procés de realitzar el projecte en solitari.

4.1 Kanban Personal

Per dur a terme la metodologia Kanban s'ha de seguir un procés:

1. Identificar el procés: En primer lloc cal identificar quin serà el flux de treball a seguir. Es comença amb tres columnes que són: per fer, en procés i finalitzat. Aquest és un flux de treball senzill que és ideal per a la gestió de projectes on només hi treballa una persona.
2. Visualitza el treball: Un cop el procés a seguir ha estat definit, cal fer un llistat de les tasques a realitzar. En el cas de que les tasques siguin molt genèriques es poden simplificar en tasques més petites. Es poden fer servir colors en les tasques per diferenciar-les en diferents tipus. A aquestes tasques se'ls hi pot afegir descripcions, data límit, imatges, prioritats...
3. Limitar la feina en progrés: Cal definir un màxim de tasques que poden estar en la columna "per fer". Aquesta tècnica ens ajuda a no perdre concentració ni productivitat a l'hora de treballar, d'aquesta manera ens podem centrar en poques tasques en concret.
4. Utilitzar el taulell Kanban: Un cop s'ha construït el taulell i s'han definit totes les regles ja es pot començar a treballar.

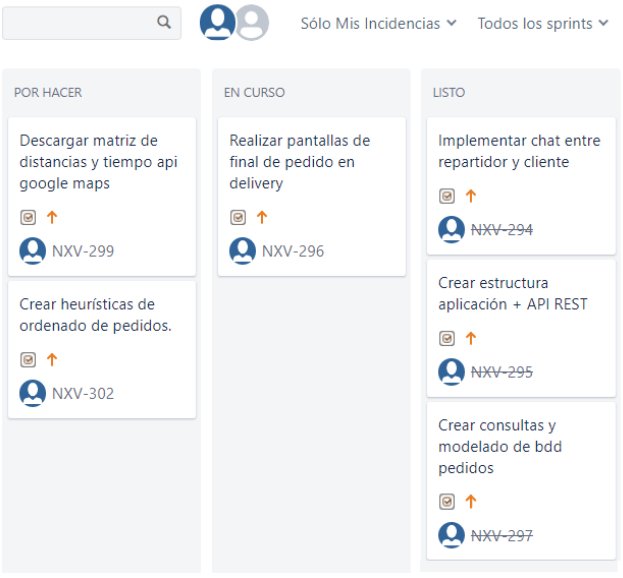

Figura 1: Taulell de tasques Kanban en Jira.

4.2 Control de versions

Per a que el desenvolupament del projecte de software sigues satisfactori, ha sigut necessari utilitzar un mètode de control de totes les versions generades durant el desenvolupament. En aquest projecte, s'ha utilitzat GIT, un software de control de versions pensat per a fer possible el desenvolupament i manteniment de codi.

En aquest projecte, principalment l'ús que se li ha donat, ha sigut el de poder recuperar antics canvis, d'aquesta manera reduïm l'impacte que pot tenir esborrar codi important, instal·lació de llibreries que posteriorment donen peu a errors de compilació, etc.
Concretament s'ha dut a terme el disseny "Gitflow Workflow"[2], un model de ramificació estricte dissenyat per a estructurar el control de versions de forma eficient amb GIT. Aquest model de ramificació consisteix en l'ús de dues branques anomenades "Màster" on trobem tots els commits de producció i "Develop" on hi hauran les versions del software que estan encara en desenvolupament i que en un futur formaran part de producció. A més d'aquestes dues branques, es proposen les següents branques auxiliars:

- Feature: neix de "Develop" i que està destinada a afegir noves característiques i funcionalitats al software.
- Release: utilitzada per incorporar el codi que sortirà a producció. Aquesta branca neix a "Develop" per a incorporar-se un cop finalitzada a "Màster".
- Hotfix: es fan servir per corregir errors existents a la branca "Master".

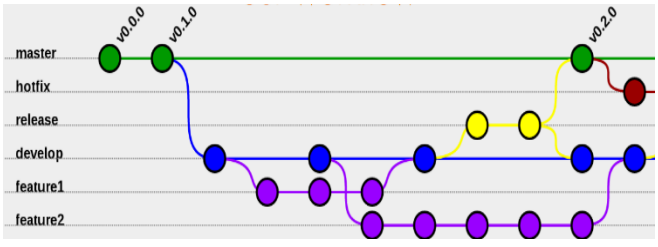

Figura 2: Exemple de flux amb GitFlow WorkFlow.

4.3 Model, View, Presenter

Es tracta d'un patró derivat del conegut patró MVC (Model Vista Controlador), és un dels patrons més populars que s'utilitza per a organitzar l'estructura de la capa de presentació en aplicacions mòbils.

El patró MVP[3] permet separar la capa de presentació de la lògica, perquè tot el que està relacionat amb el funcionaesmento de la UI sigui fidel a com es representa en pantalla. Perquè la nostra aplicació sigui fàcilment extensible i mantenible, hem de definir capes ben diferenciades. MVP fa que les nostres vistes siguin independents de les dades. Aquí es defineixen les diferents parts del patró

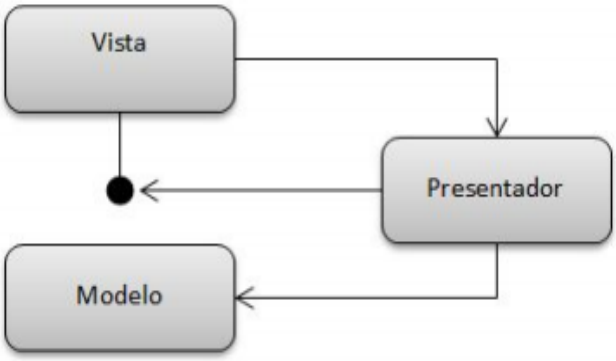

Figura 3: Funcionament del Model, View, Presenter

- Model: És una interfície responsable de controlar les dades. Les responsabilitats del model inclouen l'ús de APIs i el tractament de les dades i la base de dades.
- Presenter: És el responsable de prendre la informació del model i actualitzar la vista, reaccionant a les interaccions amb l'usuari i actualitzant el model.
- Vista: La seva única responsabilitat és presentar la informació que li arriba des del presenter.

5 DESENVOLUPAMENT

5.1 Requeriments i restriccions

Requeriments funcionals repartidor

REQF1: El repartidor ha de realitzar login contra el servidor mitjançant el domini (local o grup de locals), un usuari i contrasenya que anteriorment els tècnics de Nextt ja hagin facilitat al local. A més, s'ha de comprovar contra el servidor si el repartidor ha fet el marcatge d'entrada.

REQF2: Cada local disposa de X llicències d'ús per a repartidors. En el moment en què un repartidor fa login, una de les llicències queda bloquejada. En el moment en què finalitza la sessió, la llicència s'allibera. Per tant, no poden haver-hi més de X repartidors utilitzen-do l'aplicació al mateix temps. X ve determinat per el tipus de contracte entre l'empresa interessada i Nextt.

REQF3: El llistat de comandes a lliurar ha de presentar-se en fileres i ha de tenir la possibilitat de ser cercat mitjançant un cercador.

REQF4: Els repartidors poden consultar informació sobre la comanda en tot moment, per tant, poden tenir accés a les dades del client des de l'aplicació.

REQF5: Els repartidors han de poder configurar la manera de guiat de l'aplicació, això significa que podran triar entre diverses heurístiques de guiat (antiguitat de la comanda, mínima distància, mínim temps de trajecte i ordre lliure).

REQF6: El repartidor té control total sobre les comandes, per tant, si una comanda és lliurada la marcarà com lliurada, si existeix alguna incidència la informarà. Pot eliminar parades si ho considera.

REQF7: Cada vegada que una comanda hagi estat lliurada amb èxit o no, l'aplicació haurà de mostrar la ruta al següent punt de lliurament.

REQF8: El repartidor deu poder contactar mitjançant xat o realitzant una trucada al client en accedir a les dades d'aquest.

REQF9: L'aplicació ha d'enviar i rebre constantment in-formació al servidor, validació de comanda, notificació d'incidències, enviament d'ubicació en temps real…

REQF10: Quan un repartidor fa Logout de l'aplicació, s'ha d'alliberar una de les llicències bloquejades.

Requeriments funcionals client

REQF11: Per a que un client pugui realitzar el seguiment de la seva comanda, només ha d'introduir el número de telèfon.

REQF12: Des de la pantalla del client, aquest podrà consultar diversa informació: ubicació del repartidor, temps estimat de lliurament i detalls de la comanda. D'altra banda, ha de poder accedir a un històric de les seves comandes.

REQF13: El client té la possibilitat de contactar amb el repartidor només mitjançant el xat.

REQF14: En el moment en que la comanda ha sigut lliurada, aquesta passarà a formar part del històric de comandes.

REQF15: Un cop una comanda ha sigut lliurada, el client pot deixar feedback d'aquesta. Ha de poder escriure una valoració i deixar una nota de l'1 al 5 sobre el servei que ha rebut.

Requeriments funcionals gerent

Totes les funcionalitats que pot realitzar un gerent es realitzaran des d'una aplicació externa, ja existent, anomenada SigfridAPP. Per tant, totes les funcionalitats seran desenvolupades en un altre aplicació.

REQF16: El gerent pot decidir si veure informació sobre un local en concret o informació a nivell general de tots els seus locals.

REQF17: En el cas de voler veure informació i estadístiques d'un local en concret, la informació a la qual pot accedir serà:

- REQF17.1 Informació sobre repartidors (temps mitjans de lliurament, número de comandes realitzades, nombre d'incidències, seguiment en temps real...).
- REQF17.2 Informació en temps real sobre totes les comandes que s'estan realitzant en el local.
- REQF17.3 Estadístiques sobre l'històric de comandes del local.

REQF18: En el cas de voler veure informació sobre un conjunt de locals, podrà visualitzar estadístiques d'igual forma que en REQF17.3 però a nivell general.

Requeriments no funcionals

REQNF1 Eficiència: El sistema realitza càlculs contínuament i existeix un flux d'intercanvi d'informació constant entre client i servidor. A causa d'això és molt important que el programari realitzat compleixi mínims d'eficiència, tant en temps de resposta com en quantitat de peticions al servidor. Serà de vital importància que les consultes a bdd siguin el més òptimes possibles i que les peticions al servidor continguin tota la informació necessària per a així evitar peticions de més.

REQNF2 Seguretat: Totes les comunicacions entre client i servidor han d'estar protegides davant tercers. La informació que gestiona l'aplicació serà confidencial i per tant s'ha de garantir en tot moment que les dades estiguin fora de perill.

REQNF3 Comprovabilitat: El programari que es dissenyi ha de permetre realitzar proves en entorns tancats de forma senzilla. Per tant, és important programar de tal manera que creant un circuit de test intern es pugui reproduir el funcionament que podria tenir un local en producció.

REQNF4 Extensibilitat i escalabilitat: El programari desenvolupat ha de permetre que de forma senzilla es pugui estendre quant a funcionalitats o àmbits d'ús. Per tant, és tan important que en un moment donat sigui fàcil afegir noves funcionalitats, com que pugui ser utilitzat en comerços de retail.

REQNF5 Mantenibilitat: Tot el codi desenvolupat ha de permetre ser mantingut de forma senzilla, ja que mai se sap qui haurà de realitzar canvis o correccions en qualsevol moment donat. És important que no només el creador de l'aplicació pugui entendre i mantenir-la. La finalitat d'aquest requeriment és que sigui fàcil desenvolupar noves funcionalitats o requeriments, aïllar els defectes i les seves causes, corregir aquests defectes i atendre les demandes de l'entorn canviant.

REQNF6 Disponibilitat: El sistema ha de complir amb uns mínims de disponibilitat. En funcionar en un dispositiu mòbil, és molt usual que la connectivitat d'aquest es vegi afectada en algunes ocasions. Per tant, el sistema ha de

permetre continuar funcionant en la mesura que sigui possible en cas de falta de xarxa.

REQNF7 Usabilitat: L'aplicació ha de ser senzilla d'utilitzar, per tant, és important que la interfície sigui intuïtiva i senzilla (user friendly). D'altra banda, l'aplicació serà utilitzada com a GPS i en aquest cas, ha de ser imprescindible que el seu ús sigui únicament visual ja que la persona que la utilitzarà normalment es trobarà utilitzant un vehicle.

REQNF8 Multiplataforma: L'aplicació ha de ser multiplataforma. Per tant, ha de ser desenvolupada de tal manera que pugui funcionar tant en iOS com en Android.

Restriccions

A l'hora de desenvolupar el projecte només ha aparegut una restricció. En aquest cas, parlem de l'ús de SQL Server com a base de dades encarregada d'emmagatzemar la informació relacionada amb les comandes i la informació dels repartidors.

Al només tenir aquesta restricció s'ha pogut realitzar tota la resta del projecte amb total llibertat, sense importar llenguatge de programació, metodologia a seguir o framework de desenvolupament a utilitzar.

5.2 Tecnologies

Per implementar l'aplicació s'ha fet ús de diverses eines i tecnologies:

Visual Studio Code

Visual Studio Code[4] es tracta d'un editor de codi font sofisticat que admet moltes funcionalitats pràctiques a l'hora de treballar amb el codi. Algunes d'aquestes funcionalitats són:

- Multiplataforma: creat i dissenyat per funcionar tant en Windows, Linux com Mac OS.
- Plugins: possibilitat d'afegir una gran quantitat de plugins que ens faciliten la codificació.
- Open Source: podem trobar aquesta eina a GitHub, per tant, podem descarregar, analitzar i modificar el projecte.
- Intellisense: l'editor té la capacitat de preveure la instrucció que volem escriure i auto completar-la. Això ens fa més productius a més de reduir la possibilitat de tenir errors de sintaxis.
- Consola de comandes integrada que permet treballar amb més comoditat.

Firebase Realtime Database

Firebase Realtime Database[5] és un servei de Google que permet emmagatzemar i sincronitzar dades amb la nostra base de dades NoSQL allotjada en el núvol. Es tracta d'un servei de base de dades en temps real habilitat a la integració d'aplicacions mòbils tant iOS com Android. Per poder comunicar-se amb aquesta base de dades, es fa servir una REST API la qual serveix per crear connexions de HTTP per rebre notificacions push d'un servidor.

Sourcetree

Sourcetree[6] es un client GUI destinat a gestionar repositoris git i mercurial. Mitjançant aquesta eina podem realitzar totes les tasques de gestió de repositoris que faríem amb Git però d'una manera més simple i ràpida fent servir la interfície gràfica. Entre d'altres, aquestes son les funcions que podem realitzar:

- Crear i clonar repositoris allotjat a qualsevol lloc, ja sigui Git o Mercurial. A més es pot integrar perfectament amb Bitbucket o Github.
- Fer commit, push, pull y merge dels nostres fitxers.
- Detectar i resoldre conflictes en el nostre codi.
- Consultar l'historial de canvis del nostres repositoris.

Ionic

Ionic[7] és una eina, gratuïta i Open Source, per al desenvolupament d'aplicacions híbrides basades en HTML5, CSS3, Javascript i Typescript. Està construït amb Sass i Javascript i optimitzat amb Angular[8]. Les seves característiques principals són:

- Alt rendiment.
- Ús d'Angular amb la finalitat de crear un marc més adequat per a desenvolupar aplicacions riques i robustes.
- Centre Natiu, Ionic s'inspira en les SDK de desenvolupament mòbils natives més populars, per la qual cosa és fàcil d'entendre per a qualsevol persona que ha construït algun cop una aplicació nativa per a iOS o Android.

Figura 3: Arquitectura híbrida Ionic

SQL Server

Es tracta de un sistema de gestió de bases de dades relacionals de Microsoft que està dissenyat per a l'entorn empresarial. SQL Server[9] s'executa en T-SQL (Transact -SQL), un conjunt d'extensions de programació de Sybase i Microsoft que afegeixen diverses característiques a SQL estàndard, incloent control de transaccions, excepció i maneig d'errors, processament fila, així com variables declarades.

Jira

Jira[10] és una eina en línia que dona suport amb l'administració de tasques, seguiment d'erros i incidències d'un projecte. Disposa de taulell per gestionar tasques, un sistema de notificacions de canvis o modificacions via email i permet la realització d'informes amb informació referent temps dedicat a tasques, errors trobats, treball realitzat per cada integrant...

Servidor Node.js

Node.js[11] és un entorn de programació dissenyat per escriure aplicacions d'Internet escalables, en aquest cas un servidor web. Els programes estan escrits en JavaScript, utilitzant una arquitectura orientada a esdeveniments i entrada/sortida asíncrona per tal de minimitzar els temps de sistema i l'escalabilitat.

5.3 Mòduls

Login

En accedir a l'aplicació l'usuari haurà de seleccionar entre els dos possibles perfils, repartidor o client. Un cop escollit, ha d'autenticar-se mitjançant el seu domini (restaurant), codi de repartidor i contrasenya per al repartidor o número de telèfon en cas del client.

En el cas del repartidor, per autentificar l'usuari introduït es fa una crida al web service del servidor per comprovar que donat un codi de repartidor la contrasenya introduïda sigui la correcta. En cas afirmatiu, es genera un token mitjançant la llibreria JSON Web Tokens[12] amb el qual, posteriorment, podem fer validació de credencials al fer qualsevol altre crida al web service.

Llistat d'ordres

Si accedim al llistat d'ordres des de el perfil del repartidor trobarem un llistat de comandes associades al Codi de repartidor que prèviament ha fet login. En aquesta pantalla es poden fer diferents accions:

- Cercar comandes per codi de comanda.
- Fer un refresc de les comandes associades al Codi de repartidor o al telèfon del client.
- Accedir al detall de la comanda.
- Escollir el tipus heurística de navegació que volem seguir per a fer la navegació (només repartidor).
- Començar la navegació (només repartidor).

Per obtenir el llistat de comandes es fa una crida al web service del servidor node.js enviant com a paràmetre el codi de repartidor o el telèfon de client.

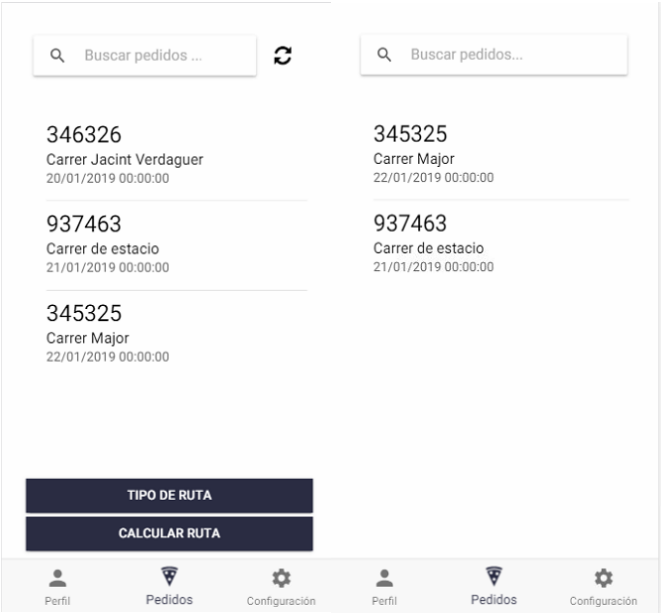

Figura 4: Disseny pantalla llistat d'ordres (repartidor i client).

Seguiment d'ordre

En accedir al detall d'una comanda des de el perfil de client podem veure el seguiment de la ruta per on va el nostre menjar. D'altra banda tenim l'opció de fer xat amb el repartidor, d'aquesta manera podem notificar-li de forma còmode qualsevol canvi o puntualització sobre la comanda.

A l'hora de mostrar el mapa de seguiment de la comanda, tenim per un costat el dispositiu del repartidor que cada 30 segons està enviant a una base de dades Firebase Realtime Database la seva posició (latitud i longitud). L'aplicació del client va a buscar aquesta informació i crea un mapa amb la ruta per on passarà la comanda i seguidament, va actualitzant un marker amb la posició del repartidor en temps real, tot gràcies a la API de Google Maps [13].

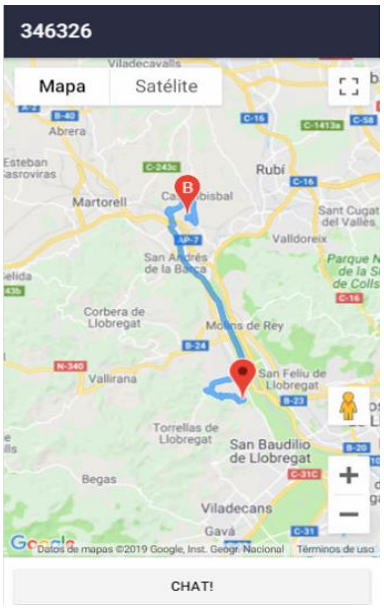

Figura 5: Disseny pantalla seguiment comanda client.

Navegació

Un cop decidit el tipus d'heurística que defineix quin serà l'ordre de parades en la navegació, accedim a la pantalla que guia al repartidor durant el lliurament.

A partir de les dades de cada comanda, es calcula la ruta i es carrega el mapa mitjançant la API de Google Maps. Des de aquesta pantalla podem accedir a la pantalla de detalls de la comanda i a la de Feedback de fi de comanda.

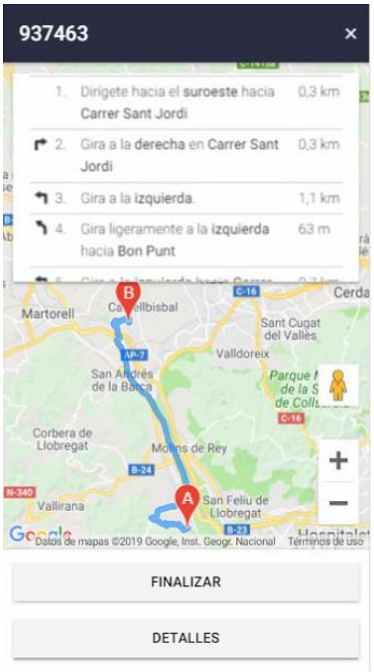

Figura 6: Disseny pantalla de navegació

Feedback fi de comanda

Per poder arribar a aquest mòdul s'ha de finalitzar una comanda.

En el cas del repartidor, un cop ha lliurat la comanda ha d'accedir a aquesta pantalla per poder continuar amb les següents. Pot indicar si ha sigut possible realitzar el lliurament, deixar un comentari de les incidències trobades i disposa d'una zona per fer signar al client com a confirmació de comanda lliurada. Un cop realitzades les accions pertinents, es fa una crida al web service del servidor, indicant el nou estat de la comanda, la signatura del client, les observacions i la data de lliurament per sobreescriure la comanda en qüestió.

En el cas del client, un cop la seva comanda ha sigut lliurada pot accedir a la informació d'aquesta per deixar un comentari i una valoració sobre el servei rebut. Enviat el feedback, es fa una crida a un servei web que guarda en base de dades el comentari i valoració del client.

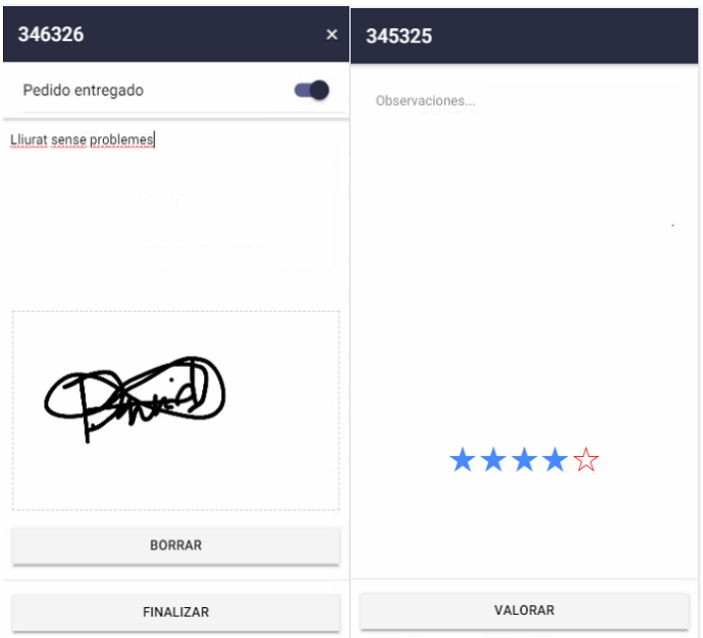

Figura 7: Disseny pantalla fi de comanda (repartidor i client).

Xat

Accedint al detall d'una comanda, tant el repartidor com el client poden accedir a la sala de xat on poden posar-se en contacte entre ells. Per dur a terme aquesta funcionalitat es fa servir Firebase Realtime Database[14]. En primer lloc es comprova si existeix una sala de xat amb el nom del codi únic de comanda, si no existeix la crea. Seguidament quan un usuari entra a la sala de xat, es crea un username per aquest, depenent de si es repartidor o client el username serà el codi de repartidor o el número de telèfon del client. A l'hora de xatejar entre les dues parts, els missatges es van

guardant a la base de dades i es van sincronitzant de forma contínua per a cada usuari.

5.4 Travelling Salesman Problem

El problema del Travelling Salesman Problem[15] és el problema d'optimització de trajectòries donat per l'enunciat següent: donat un conjunt de nodes, es tracta de trobar l'ordre de visites a seguir per tal de definir una trajectòria que passi un sol cop per a cada node i de manera que la distància total recorreguda sigui la més curta possible.

En el cas de l'aplicació, a l'hora de calcular l'ordre de les comandes per a la navegació, ens trobem amb un problema que respon a aquest algorisme, en concret en els casos de mínima distancia i mínim temps. Degut a que la distància entre les diferents destinacions no és la mateixa a la inversa (exemple: distancia de A a B no és igual que de B a A), podem confirmar que es tracta d'un TSP asimètric.

Per resoldre aquest problema en primer lloc obtenim les dades a tractar mitjançant una crida a la API Google Maps Matrix, un servei web de Google que retorna un Json amb totes les distàncies i temps entre un llistat d'orígens i destinacions. Tractem el Json obtingut de tal manera que podem crear un graf, indicant els nodes i els arcs que el comprenen.

```
[ [ 0, 20608, 20530, 19420, 18322 ],
  [ 19171, Infinity, 45, 839, 1072 ],
  [ 19126, 751, Infinity, 793, 1027 ],
  [ 17644, 4249, 4171, Infinity, 1962 ],
  [ 17864, 1837, 1759, 663, Infinity ] ]
```

Figura 8: Matriu adjacència comanda de 4 destinacions.

A continuació, un cop disposem del graf amb tots els nodes, arcs i costos corresponents, executem l'algoritme TSP per obtenir l'ordre de les parades que hauria de prendre el repartidor per realitzar la ruta de la forma més òptima. Aquest algoritme intenta totes les permutacions possibles i tria la de menor cost, per tant es tracta d'una cerca de força bruta. Això pot arribar a ser un problema, ja que en cas de tenir moltes destinacions, la complexitat creix fins a el punt de ser impracticable. Tot i aquesta restricció, en un entorn real de funcionament, serien poques les ocasions en que un repartidor hagués de lliurar més de 3 o 4 comandes i per tant el rendiment al calcular una solució no es veuria afectat.

5.5 Test i qualitat

Per poder donar com a bona una tasca realitzada, s'han de complir uns punts de qualitat que es poden trobar al document anomenat "Definition of Done". D'altra banda, s'ha de seguir el document anomenat "Document de criteris d'acceptació" que inclou els punts de qualitat que ha de tenir el producte (acordats amb l'interessat).

5 RESULTATS

En aquesta secció exposem els resultats extrets del desenvolupament de NexttPOS Delivery APP. Fem una descripció de les funcionalitats implementades per a cada perfil, descrivim els avantatges de la solució i destaquem les seves funcionalitats clau:

Perfil de repartidor:

S'ha aconseguit desenvolupar quasi tots els requeriments prèviament llistats per aquest perfil. Entre les funcionalitats principals d'aquest perfil podem trobar:

- Login funcional i segur, totes les crides al web service es fan mitjançant POST i a més al validar la autentificació d'usuari s'obté un token que posteriorment s'utilitza a l'hora de fer les altres crides al servidor.
- Navegació funcional, en un primer moment s'esperava realitzar una navegació interactiva que anés guiant al repartidor pas per pas. No ha sigut possible degut a les limitacions que la API de Google ens imposa. Per aconseguir un guiat pas a pas es necessitaria fer ús de la ampliació externa de Google Maps. Com a alternativa, es mostra una pantalla amb la ruta i els passos a seguir.
- Xat amb el client funcional, el repartidor pot obrir una finestra de xat amb el client en qualsevol moment per poder comunicar qualsevol incidència amb la comanda. Per aquesta funcionalitat, es va obviar l'ús de la base de dades que fa servir Nextt i es va optar per una més lleugera i eficaç, és a dir per Firebase Realtime Database.
- Gestió de llicencies no implementada, al dependre de altres projectes de l'empresa, aquesta funcionalitat no ha pogut ser realitzada. Es realitzarà més endavant.
- Optimitzar l'ordre de lliurament de comandes funcional. Depenent del moment de enregistrament en el sistema d'una comanda, mínim de temps o distancia o lliure elecció, s'ordenen les comandes amb l'objectiu d'oferir una navegació òptima depenent de les necessitats del repartidor o restaurant en cada determinat moment.
- Notificació de comanda lliurada funcional, en lliurar una comanda es guarden correctament en base de dades tota la informació d'interès per al local (signatura client, data de lliurament, observacions...).

Perfil de client:

- Login funcional, el client fa el seu login introduint el seu número de telèfon. Aquesta és una funcionalitat que es va determinar amb Nextt. Probablement en un futur es millori aquesta funcionalitat, en lo personal, crec que hauria d'existir alguna informació extra que fes aquesta funcionalitat més privada.
- Xat amb el repartidor funcional,.
- Feedback en comandes finalitzades funcional. El client pot deixar una valoració sobre la seva comanda que es guarda en base de dades.

- Seguiment en temps real de la comanda funcional. S'ha aconseguit, mitjançant Firebase Realtime Database, que l'aplicació del repartidor emmagatzemi la seva posició GPS cada 30 segons. D'aquesta manera l'aplicació del client recupera aquesta ubicació i pot veure en tot moment on es troba la seva comanda.

Perfil gerent

En aquest cas no s'ha pogut realitzar cap requeriment dels marcats al principi del projecte. Al ser un perfil menys prioritari, Nextt va decidir que aquest seria desenvolupat més endavant.

6 Conclusions

L'objectiu principal d'aquest projecte era crear una aplicació multiplataforma que s'ajustarà al que demanaven els interessats pel projecten cas de no finalitzar tots el requisits plantejats, l'idea era crear les bases suficients per poder continuar amb el desenvolupament del projecte més endavant. Degut al "handicap" d'haver de desenvolupar la part enfocada al gerent en una app ja existent, les limitacions trobades han fet que aquesta part no hagi pogut ser finalitzada. La principal raó d'això és a causa de la gran quantitat de temps que s'ha d'invertir en entendre i començar a desenvolupar amb facilitat en un llenguatge desconegut.

D'altra banda, al ser un projecte realitzat amb una empresa, en moltes ocasions reunir als interessats per fer consultes es tornava una tasca complicada i en moltes ocasions tasques molt trivials s'han anat deixant pendents de fer degut a la poca informació que en ocasions es disposava.

Per últim, s'ha de tenir en compte que al ser una aplicació que ha de treballar juntament amb altres, la interacció entre alguna de les altres aplicacions ha sigut simulada. Per exemple, les comandes que rep un repartidor havien de ser creades manualment ja que l'aplicació encarregada de ferho encara no disposava d'aquesta funcionalitat.

Algunes millores futures poden ser:

Afegir procés de compra

Actualment, si un client d'un restaurant vol fer una comanda, ho ha de fer via web o per telèfon. En un futur es podria realitzar una plantilla, on un cop seleccionat el restaurant es carreguessin el llistat de plats a demanar. Un cop la comanda ha sigut realitzada i pagada, l'aplicació hauria d'enviar una notificació al restaurant per a que puguin començar amb el procés de lliurament.

Notificacions push

L'usuari, tant client com repartidor, hauria de rebre una notificació al seu dispositiu, tingui o no oberta l'aplicació. Alguns exemples que invoquin l'enviament d'una alerta poden ser: alertes sobre canvis en la comanda, noves comandes a lliurar per el repartidor, alerta de xat, etc.

Millorar el disseny

Nextt disposa de d'un dissenyador interfícies gràfiques, seria interessant demanar-li que dissenyes logotips, pantalles i noves imatges de cara a la versió final del producte.

Crear alternatives al TSP

Si volem fer servir l'aplicació en altres àmbits com els de logística, el algoritme TSP no seria el més adient degut a l'alta complexitat que suposa al tenir una quantitat elevada de destinacions. Per tant, com a millora es podria plantejar la possibilitat de desenvolupar un altre algoritme de cerca de camins de mínim cost que no hagi d'intentar totes les permutacions i que tot i no oferir la millor solució pogués crear una aproximada.

Noves heurístiques d'ordenament de comandes

En algunes ocasions hi ha alguns plats els quals el temps de lliurament no afecta a la seva qualitat (productes freds). És per això que es podria plantejar una nova heurística la qual ordenes les comandes segons el tipus de plat que contenen. D'aquesta manera, la probabilitat de que el menjar arribi fred es veu reduïda.

8 Agraïments

Personalment vull agrair a tota persona que m'hagi ajudat a realitzar aquest projecte, tant en el moment de la realització com al abans de començar-lo. Donar les gràcies a l'empresa Nextt per oferir-me l'oportunitat de poder dur a terme aquesta proposta de projecte.

Per últim agrair a Ferran López, com a cap de projectes de Nextt, la confiança depositada en mi i a Xavi Pol, programador de Nextt, per donar-me opinió i consell al llarg de tota l'etapa de desenvolupament.

9 Bibliografia

[1] Javier Garzas (2011). ¿Qué es el método Kanban para la gestión de proyectos? [online] Consultat 9/2018 https://www.javier-garzas.com/2011/11/kanban.html

[2] AprendeGit, "Que es Git Flow?", 2013. Consultat 9/2018. Disponible: http://aprendegit.com/que-es-git-flow/

[3] Lars Gyrup Brink Nielsen (2018). Model-View-Presenter with Angular [online] Consultat: 9/ 2018 https://codelabs.developers.google.com/codelabs/firestoreweb/#0

[4] Documentation | Visual Studio Code [online] Consultat 9/2018 https://code.visualstudio.com/docs

[5] Firebase Database + Ionic. [online] Consultat: 10/2018 https://blog.ng-classroom.com/blog/ionic2/firebase-database-andionic/

[6] Documentation | Sourcetree [online] Consultat 9/2018 https://confluence.atlassian.com/get-started-with-sourcetree

[7] Documentation | Ionic Framework. [online] Consultat 9/2018 https://ionicframework.com/docs/

[8] Angular.io. (2018). Angular Docs. [online] Consultat 9/2018 https://angular.io/tutorial

[9] Documentation | SQL Server [online] Consultat 9/2018 https://docs.microsoft.com/es-es/sql/sql-server/sql-server-technical-documentation

[10] Documentation | Jira [online] Consultat 9/2018 https://confluence.atlassian.com/jira/jira-documentation-1556.html

[11] Documentation | Node.js [online] Consultat 10/2018 https://nodejs.org/es/docs/

[12] Documentation | JSON web Tokens [online] Consultat 10/2018 https://jwt.io/introduction/

[13] Beginning Google Maps API 3, Gabriel Svennerberg, 2010 Consultat 11/2018

[14] **D**idin J. Build Ionic 3, Angular 5 and Firebase Simple Chat (2018) Consultat 11/2018 https://www.djamware.com/post/5a629d9880aca7059c142976/build-ionic-3-angular-5-and-firebase-simple-chat-app

[15] Trekhleb Repositori GitHub amb exemples de algoritmes i estructuras de dades en Javascript. Consultat 12/2018 https://github.com/trekhleb/javascript-algorithms