PopFood

Nom Estudiant: Miquel Casals Pelegri

Màster Universitari en Desenvolupament d'Aplicacions per a Dispositius Mòbils

Nom Consultor/a: Albert Mata Guerra

Professor/a responsable de l'assignatura: Carles Garrigues Olivella

3 de Gener de 2018

FITXA DEL TREBALL FINAL

Títol del treball:	*PopFood*
Nom de l'autor:	*Miquel Casals Pelegri*
Nom del consultor/a:	*Albert Mata Guerra*
Nom del PRA:	*Carles Garrigues Olivella*
Data de lliurament:	*01/2018*
Titulació o programa:	*Màster Universitari en Desenvolupament d'Aplicacions per a Dispositius Mòbils*
Idioma del treball:	*Català*
Paraules clau	*Ionic 3, Firebase, Angular 2*

Resum del Treball:

El treball defineix el procés de creació d'una aplicació per a dispositius mòbils anomenada PopFood seguint el model de disseny centrat en l'usuari i implementada amb Ionic 3 i Firebase.

PopFood és una aplicació de tipus e-commerce que permet a tots els usuaris localitzar, contactar i comprar a tots els productors d'aliments artesanals pròxims basant-se en la ubicació de l'usuari. PopFood ofereix la possibilitat a tots els productors de construir i gestionar, totalment des del mòbil, un aparador virtual de tots els productes que disposa per vendre en temps real facilitant la venda dels seus productes directament al consumidor sense intermediaris.

Abstract:

This document defines the design and implementation process of PopFood, a mobile focused app. The design process follows the model of DCU (Design Centred in Users) and the app will be built on Ionic 3 and Firebase.

PopFood is an e-commerce application that allows users to locate, contact and reserve to local food producers based on their location. PopFood offers the opportunity to all producers to build and manage, fully through the mobile device, their own virtual store sharing at real time the products that they have available facilitating the own selling without any middleman in business.

Índex

Llista d'imatges

Llista de gràfics.

Llista de taules.

1. Introducció

1.1. Context i justificació del Treball

Com a fill de pagesos, estava habituat a disposar de productes frescos i de qualitat directament de l'hort. En el moment que vaig deixar de viure al meu poble per anar a una ciutat, aquests productes ja no eren al meu abast amb la mateixa facilitat. I, tot i viure en una ciutat rodejada de milers d'hectàrees de conreu, no és gens fàcil disposar de fruits i verdures de qualitat. Per què és tan difícil accedir als aliments dels productors situats al voltant del lloc on visc?

Són pocs els agricultors que distribueixin els seus productes directament al consumidor, ja sigui mitjançant una botiga física, virtual, o amb enviament a domicili sota comanda, entre altres mètodes. El motiu és que suposa un esforç, tan físic com en recursos que, habitualment, no es pot assumir.

La majoria d'agricultors comercialitzen els seus productes mitjançant cooperatives o distribuïdors privats on, del benefici que se n'obtindrà, se'n descomptarà els costos de distribució, manipulació, refrigeració, transport i la comissió que rebrà cada intermediari que hi hagi en la traça d'aquests. Molts d'aquests agricultors són petites empreses familiars amb capacitat per produir qualitat al detall però que no pot competir en producció i preu amb grans terratinents deixant com a resultat que en moltes ocasions el benefici obtingut és molt baix sinó inexistent.

Alguns estudis [1] ja indiquen que els consumidors tendeixen cada cop més a consumir productes de més qualitat perquè també valoren més la salut. Va en augment la tendència a comprar aliments per internet tal com es pot veure a la Taula 1, on es pregunta a l'enquestat si ha realitzat alguna compra d'aliments per internet, obtingut de l'informe del consum d'aliments a Espanya 2016 [2] (p. 224).

(%)	2004	2005	2006	2007	2008	2010	2011	2012	2013	2014	2015	2016
SÍ	2,7	3,5	4,3	5,4	4,7	7,8	8,5	9,5	10,4	10,5	10,8	**9,0**
NO	97,3	96,5	95,7	94,6	95,3	92,2	91,5	90,5	89,6	89,5	89,2	**91,0**
Bases	**8.000**	**8.000**	**8.018**	**3.007**	**4.012**	**2.402**	**2.600**	**1.500**	**1.500**	**1.500**	**1.500**	**1.500**

Taula 1. Evolució de la població que compra aliments per internet.

La ràpida evolució de la tecnologia mòbil i de comunicacions ha comportat canvis radicals en els hàbits d'ús d'aquests aparells. Actualment pots adquirir qualsevol cosa des d'on siguis i que te la portin a casa o contactar amb un altre usuari per comprar on el producte que ell ven i posar-se d'acord amb el preu i el lloc d'intercanvi. L'aparició d'aplicacions com *Wallapop* [3], on es posa en contacte usuaris de carrer facilitant compravenda de productes de segona mà, ha obert un nou model de venda.

PopFood pretén crear un canal de venda directa sense intermediaris entre productors i consumidors oferint els següents avantatges respecte al context actual:

- El preu del producte serà un preu just per al productor però al mateix temps, amb l'absència d'intermediaris, el cost serà més baix per al consumidor que en altres establiments.
- Els productes estaran en el millor punt de maduració o de frescor, ja que no hi haurà períodes de transport ni entrarà en cadena de fred si el producte ja no ho requereix.
- Consumir productes de proximitat redueix les emissions contaminants a l'atmosfera i beneficia a l'economia local.
- Els consumidors poden trobar productors artesanals en qualsevol lloc on estiguin. Ja sigui a l'entorn del lloc on resideixen o si es troben en un altre indret de vacances, per exemple.
- El productor guanyarà visibilitat als potencials clients i, gràcies al contacte directe amb els consumidors, pot dur a terme campanyes de publicitat i/o fidelitat a una franja de públic molt més específica.

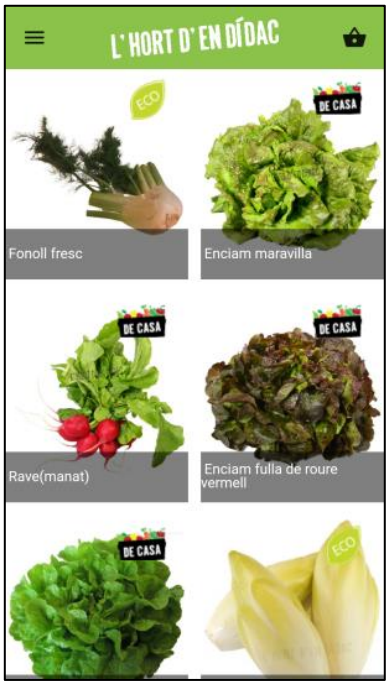
1.2. Estudi de competència

Després de fer una cerca per internet i en el mercat d'aplicacions Google Play Store, s'han identificat algunes aplicacions similars però que no resolen els problemes que pretén solucionar PopFood. Des d'aplicacions de botiga virtual de comerços físics de productes ecològics fins a aplicacions de cadenes de supermercats com Carrefour.

1.2.1.L'hort d'en Dídac [4]

Descripció:

L'Hort d'en Dídac és una aplicació on pots adquirir productes d'una única agrobotiga situada a Mataró. Aquesta agrobotiga es proveeix bàsicament d'una explotació familiar del Maresme però també disposa de productes de tercers. No es proporciona informació sobre els productes ni sobre els productors, solament dades de contacte, i únicament s'indica quins productes són de "la casa".



Imatge 1. App L'hort d'en Dídac, llista de productes.

Tipus d'aplicació:

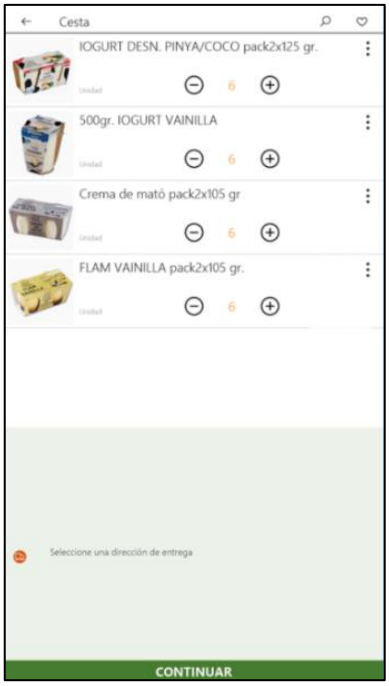
Aplicació per iOS i Android.

Punts positius:

- Aplicació bàsica que cobreix perfectament els objectius amb què segurament es va crear: Realitzar comanda, triar quan la vols i pagar.
- El sistema de distribució de les comandes és versàtil i fàcil d'entendre. S'ofereix una llista de punts on es pot recollir i horaris de forma que es pot triar el lloc en funció de quan ho pots recollir.

Punts negatius:

- No s'ofereix cap tipus de descripció del producte. Només foto, preu i quantitat a triar.
- Dona servei a una regió pròxima a la seva ubicació i prou.
- No hi ha possibilitat d'incorporar nous productors. Es tracta d'una aplicació de venda en línia d'una botiga.



Imatge 2. App Agrofresc. Llista de productes amb filtres aplicats.

1.2.2.Agrofresc [5]

Descripció:

Aplicació orientada a comerç al detall que permet adquirir productes làctics d'un grup d'empreses productores.

Tipus d'aplicació:

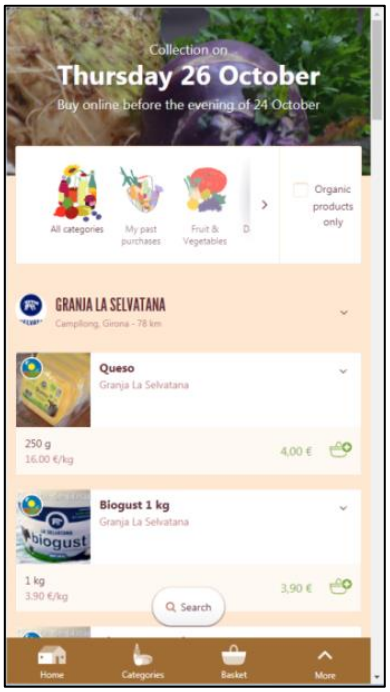
Per l'experiència d'ús i el disseny de la interfície sembla haver estat creada com a aplicació híbrida.

Punts forts:

- S'hi ofereix productes artesans de qualitat de diferents petits productors.

Punts dèbils:

- L'aplicació serveix com a servei logístic majorista.
- Baix rendiment de l'aplicació. Les transicions són lentes i en algun moment es bloqueja temporalment la carrega d'imatges o canvi d'interfície.



Imatge 3. Pàgina web de "La colmena dice sí".

1.2.3.The food Assembly [6]

Descripció:

The food assembly és un moviment d'abast europeu, originari de França i que arriba a Espanya el 2014 amb el nom "La colmena dice sí", que pretén facilitar el consum de productes de proximitat mitjançant agrupacions de consumidors on diferents productors subministren productes.

Tipus d'aplicació:

The Food Assembly disposa d'una web totalment *responsive*

perfectament accessible i amb bona experiència d'usuari amb dispositiu mòbil i ordinador.

Punts forts:

- Els objectius de l'aplicació són similars als de PopFood.
- Els productes oferts estan ordenats per productor.
- Es pot contactar amb el productor directament.
- Entorn visual de la web *responsive*, agradable i usable.

Punts dèbils:

- La distribució de productes es realitza únicament en un punt de recollida.
- El procés per poder proveir una agrupació com a productor requereix la participació de l'administrador d'aquesta.
- Cada agrupació disposa d'un administrador

Imatge 4. App CerroViejo pàgina principal.

1.2.4.Cerro Viejo [7]

Descripció:

Es tracta d'una organització que es defineix com a sense ànim de lucre, de comerç just i que ofereix la possibilitat d'adquirir productes de proximitat i ecològics a través de venda en línia amb el mínim d'intermediaris possible. Es tracta d'una comunitat tancada on has de ser soci per adquirir productes.

Tipus d'aplicació:

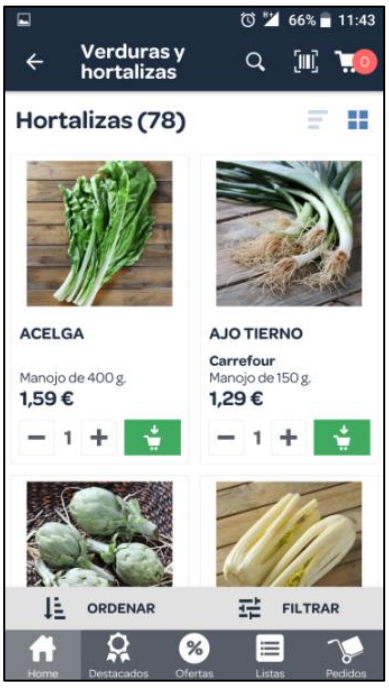
Aplicació nativa per Android

Punts positius:

- Productes de qualitat sense intermediaris.
- Procés de comanda senzilla.

Punts negatius:

- Interfície de l'aplicació poc optimitzada.
- No hi ha possibilitat d'incorporar nous productors de forma ràpida i senzilla.
- Has de ser soci per poder accedir a la botiga.



Imatge 5. Carrefour app, llista de productes aplicant filtres.

1.2.5.Carrefour [8]

Descripció:

Es tracta d'una aplicació de venda en línia clàssica de supermercat que proporciona una experiència correcta.

Tipus d'aplicació:

Sembla ser una aplicació híbrida per Android i iOS.

Punts forts:

- Fàcil d'utilitzar.
- Aplicació àgil i bona experiència d'usuari.
- Descripció detallada dels productes.
- Ofereix la possibilitat de guardar una llista de productes que necessites per la pròxima compra.

Punts dèbils:

- No encaixa amb els objectius de PopFood, ja que només distribueix d'un únic proveïdor.
- L'única aportació de l'aplicació és que pots realitzar la compra sense desplaçar-te a la botiga.

1.3. Objectius del Treball

L'objectiu del treball és fer el seguiment del procés de creació d'una aplicació de tipus hibrida desenvolupada amb tecnologia web fent ús dels diferents *frameworks* que existeixen actualment. L'aplicació es desenvoluparà des de zero i s'anomena *PopFood*. Aquesta aplicació consta de dues parts diferents que corresponen als perfils d'usuari.

Consumidor.

Una part de tipus *e-commerce* que li oferirà la possibilitat a l'usuari de buscar productors a partir de la seva ubicació podent configurar el buscador per tipus, preferits, distància de cerca, etc. i contactar amb ells.

Productor.

Es tracta d'un aparador virtual dels productes que comercialitza l'usuari. La gestió de la botiga es realitzarà íntegrament pel dispositiu mòbil amb possibilitat d'incorporar accés web en futurs actualitzacions.

Totes les dades dels usuaris s'emmagatzemaran en el servidor per tant estaran disponibles en tots els dispositius.

Amb motiu del limitat termini de desenvolupament del treball no s'implementarà l'aplicació completa sinó que es focalitzarà en la part corresponent al Productor que consta de les següents funcionalitats:

- Autenticació d'usuari realitzant la validació en servidor (Per tots els perfils d'usuari)

- Gestió / alta de perfil d'usuari (Per tots els perfils d'usuari).
- Gestió de la botiga i dels productes a la venda des del dispositiu mòbil. (només perfil Productor)
    - Edició de les dades de la botiga (Nom, logotip, localització, text de presentació, horaris de sol·licitud i entrega de comandes).
    - Inserció de nous productes (Preu, Descripció, imatges, etc).
    - Eliminació de productes.
    - Modificació de productes.
    - Pujada d'imatges obtingudes via càmera del dispositiu mòbil i emmagatzemades al servidor.
- Sincronització de les dades al núvol.

Però, per tal que es pugui entendre els objectius de *PopFood* i el seu funcionament, la fase de disseny es contemplarà amb l'aplicació completa.

1.4. Enfocament i mètode seguit

La intenció és crear una aplicació nova seguint un procés de disseny centrat en l'usuari. L'aplicació pot tenir similituds amb altres aplicacions existents per la qual cosa s'ha realitzat un petit estudi d'aquestes en l'apartat Estudi de competència d'aquest document. PopFood es basa en un escenari real i per poder estudiar els requisits de forma correcta es duran a terme les següents tasques:

- Definició dels perfils d'usuari.
- Escenaris actuals i futurs.
- Definició dels casos d'ús principals.
- Definició de les funcionalitats que ha de tenir l'aplicació.
- Definició de flux de navegació.
- Definició d'interfícies i prototipat de baix nivell.

Un cop es tingui el disseny definit es passa a la part d'implementació, que es realitzarà amb metodologia Àgil. S'intentarà crear l'aplicació per fases de forma que es preveu tenir una part operativa com més aviat millor per poder millorar-la i incorporar altres funcions. Es continuarà amb els següents passos:

- Implementació de l'aplicació bàsica sense persistència.
    - Definició de l'estructura de dades
    - Creació d'interfícies.
    - Implementació de lògica de controladors.
- Integració de la persistència en Firebase i emmagatzematge local.

- Implementació de l'autenticació en servidor i emmagatzematge local de les dades d'autenticació si es vol mantenir connectat.
- Implementació de lògica d'accés a base de dades de servidor.
- Implementació de lògica d'emmagatzematge d'imatges en servidor.

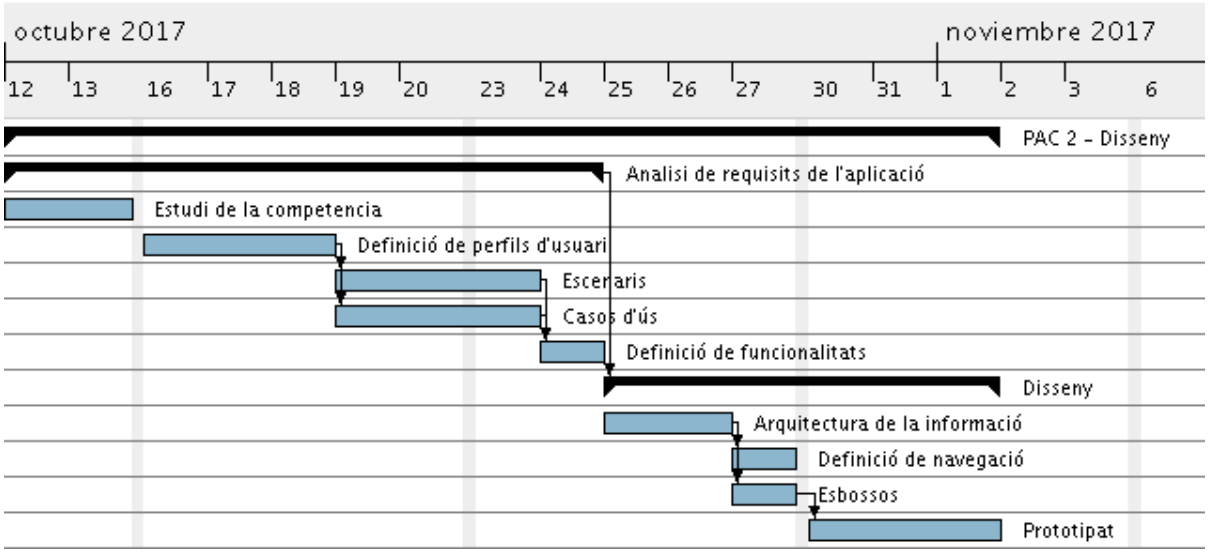
- Implementació de funcions natives.
  - Captura d'imatges mitjançant la càmera del dispositiu.

1.5. Planificació del Treball

Les tasques previstes i la seva inversió en hores és la que es pot consultar en les següents taules i diagrames. La previsió es realitza sobre un treball de 3 hores diàries de mitja sense tenir en compte dies no laborals.

ID	Nombre	Fecha de ini...	Fecha de ...	Antecesores	Duración
8	PAC 2 - Disseny	12/10/17	1/11/17		15
6	Analisi de requisits de l'aplicació	12/10/17	24/10/17		9
16	Estudi de la competencia	12/10/17	13/10/17		2
0	Definició de perfils d'usuari	16/10/17	18/10/17		3
2	Escenaris	19/10/17	23/10/17	0	3
15	Casos d'ús	19/10/17	23/10/17	0	3
14	Definició de funcionalitats	24/10/17	24/10/17	2,15	1
17	Disseny	25/10/17	1/11/17	6	6
23	Arquitectura de la informació	25/10/17	26/10/17		2
25	Definició de navegació	27/10/17	27/10/17	23	1
26	Esbossos	27/10/17	27/10/17	23	1
27	Prototipat	30/10/17	1/11/17	26	3

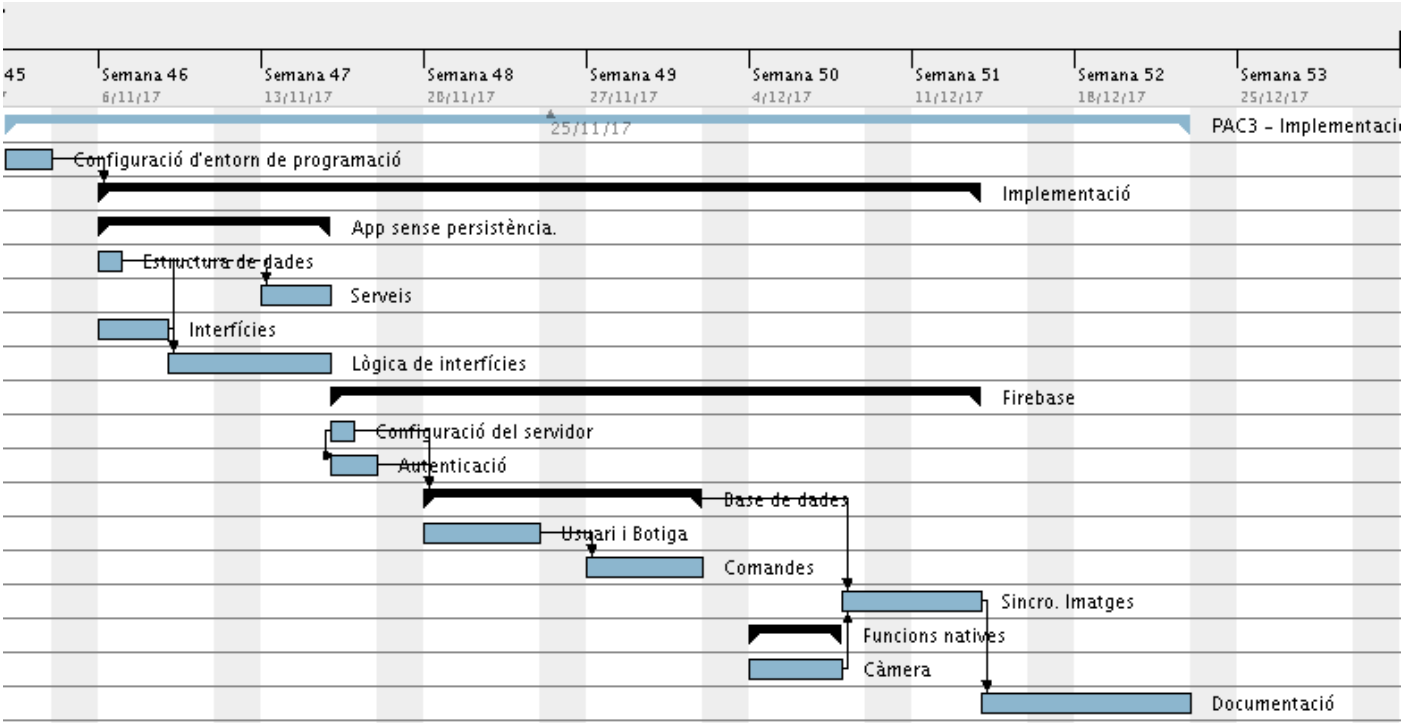
Taula 2. Previsió de hores de treball per tasca de la PAC2 Disseny.



Gràfic 1. Diagrama de GANTT de la planificació de la PAC2 Disseny.

ID	Nombre	Fecha de inicio	Fecha de fin	Antecesores	Duración
0	PAC3 - Implementació	2/11/17	23/12/17		37
16	Configuració d'entorn de programació	2/11/17	3/11/17		2
13	Implementació	6/11/17	13/12/17	16	28
28	App sense persistència.	6/11/17	15/11/17		8
7	Estructura de dades	6/11/17	6/11/17		1
6	Serveis	13/11/17	15/11/17	7	3
17	Interfícies	6/11/17	8/11/17		3
4	Lògica de interfícies	9/11/17	15/11/17	7,17	5
30	Firebase	16/11/17	13/12/17		20
8	Configuració del servidor	16/11/17	16/11/17		1
9	Autenticació	16/11/17	17/11/17	8	2
10	Base de dades	20/11/17	1/12/17	8,9	10
11	Usuari i Botiga	20/11/17	24/11/17		5
12	Comandes	27/11/17	1/12/17	11	5
19	Sincro. Imatges	8/12/17	13/12/17	10,18	4
32	Funcions natives	4/12/17	7/12/17		4
18	Càmera	4/12/17	7/12/17		4
14	Documentació	14/12/17	23/12/17	19	7

Taula 3. Previsió de hores de treball per tasca de la PAC3 Disseny.



Gràfic 2. Diagrama de GANTT de la planificació de la PAC3 Implementació

1.6. Breu sumari de productes obtinguts

Un cop finalitzat el procés del treball que es descriu en aquest document s'obtindrà els següents elements:

Binari de l'aplicació per Android.

Paquet d'instal·lació .apk per a dispositius mòbils Android.

Codi font.

Carpeta completa del codi font de l'aplicació.

Prototip de baix nivell.

Documentació corresponent al prototip de baix nivell.

1.7. Breu descripció de les tecnologies i materials emprats.

1.7.1.Equipament.

Ordinador portàtil **Macbook Pro** de finals del 2011 amb processador Intel Core i7 2.8Ghz i 16Gb de RAM. Per la realització de la documentació i implementació de l'aplicació serà necessari.

Mòbil **BQ Aquaris M5** amb sistema operatiu Android 6.0.1 per proves amb la versió final del treball.

1.7.2.Software de documentació.

Microsoft office for Mac 2011 (Word) per realitzar la memòria del treball.

Gantt Project per la planificació temporal del treball.

Draw.io per la generació de diagrames.

1.7.3.Prototipat.

Balsamiq per la realització del prototip de baix nivell.

1.7.4.Programació.

Microsoft Visual Studio Code com a IDE principal d'edició del codi web.

Git v.2.14.3 per control de versions i GitHub per tenir el codi disponible en un repositori remot. Això facilitarà poder treballar de dues localitzacions diferents.

Ionic 3.18.0 com a *framework* principal que ens permetrà també generar el paquet PKI que s'instal·larà per proves.

FireBase 4.7 com a *framework* per crear la base de dades, servei d'autenticació i emmagatzematge de les fotos proporcionades.

Node.js v.8.9.1 realitzarà la tasca de servidor local per realitzar proves en l'aplicació.

Npm v.5.5.1 serà el servidor de paquets que s'utilitzarà per gestionar les dependències.

1.8. Breu descripció dels altres capítols de la memòria

1.8.1. Anàlisi de requisits.

En aquest punt es determina els possibles usuaris que utilitzaran l'aplicació, en quin context i com la utilitzaran per a definir les funcionalitats. El capítol 2, corresponent a l'anàlisi de requisits, es divideix en 5 seccions:

Definició d'usuaris. Descripció dels usuaris tipus amb una definició de caràcter general i una més detallada amb la fitxa de persona que ens permet contextualitzar millor la seva situació.

Contexts d'ús. Descripció d'en quin context utilitzarà l'aplicació per tal de marcar algunes línies de treball en el disseny d'interfícies.

Escenaris. Descripció de casos on l'aplicació aportarà millores en el context dels usuaris.

Casos d'ús. Definició de què podrà fer cada usuari amb l'aplicació i quins passos es preveu que hagi de realitzar per dur a terme les tasques.

Definició de funcionalitats. Llista de funcions que ha de cobrir l'aplicació que ens aportarà més dades a l'hora de dissenyar les interfícies.

1.8.2. Disseny

El punt 3 del document es centra en el disseny visual de l'aplicació.

Arquitectura de la Informació. Definició de quines dades han de mostrar-se en cada interfície per tal de tenir-ho en compte més tard en el prototipat.

Arbre de navegació. Definició de com l'aplicació permetrà als usuaris moure's d'una interfície a una altra dins d'aquesta.

Definició d'interfícies i esbossos. Definició de com seran les interfícies i quins detalls s'han de tenir en compte amb els primers esbossos de paper amb una aproximació al que després serà el prototip de baix nivell.

Prototipat. Disseny d'un prototip de baix nivell que permet definir amb més exactitud les interfícies que posteriorment s'implementaran en el capítol d'implementació.

1.8.3. Implementació

El punt 4 del treball mostra els punts importants de la implementació de l'aplicació.

Configuració de l'entorn de programació. Indicació dels punts necessaris per preparar l'entorn de treball.

Estructura de dades. Definició del model de dades que gestionarà PopFood.

Interfícies Punts importants de l'estructura i la lògica de la part frontal de l'aplicació.

Serveis Explicació dels punts importants de la lògica de serveis de l'aplicació.

Firebase Configuració i utilització de Firebase.

2. Anàlisi de requisits.

2.1. Definició d'usuaris

2.1.1. Audiència

L'aplicació està dirigida a persones de qualsevol sexe i que viuen de forma autònoma. Per la banda dels usuaris amb l'aplicació bàsica de compra, es tracta de persones preocupades pel medi ambient i la salut dels seus però, també es poden veure motivats per aficions culinàries i per obtenir productes d'alta qualitat que no es poden trobar en botigues convencionals. Aquests usuaris estan disposats a invertir una mica més d'esforç en la realització de la compra, ja sigui recollint la comanda per un lloc diferent del que realitzes la compra, pagant unes despeses d'enviament, o recollint-la per casa del client.

Per altra banda, els usuaris amb l'aplicació Prèmium, són persones que treballen en el sector primari produint aliments de forma artesanal. Es tracta de petites empreses o explotacions familiars amb un model de treball menys intensiu i amb baixos volums de producció que permeten la venda al detall. Aquestes busquen nous canals de venda directa sense intermediaris, així com disposar d'una borsa de clients estable.

2.1.2. Rols

Venedor

El venedor disposa de la possibilitat de crear i gestionar la botiga virtual, que serà visible al mapa i llista de cerca.

Comprador

El comprador pot cercar venedors al mapa i veure el contingut del seu aparador, consultar les seves dades (contacte, horari de venda, modes d'enviament, etc) i posar-se en contacte amb ell.

El comprador pot configurar el cercador per filtrar els venedors visibles per tipus de producte, distancia a què es troben, així com poden guardar els venedors i productes preferits.

2.1.3. Perfils d'usuari

Inicialment es preveu que hi hagi únicament dos perfils d'usuari, consumidors i productors. La definició dels perfils es basa en informació obtinguda d'estudis públics realitzats per diferents entitats.

Productor

El productor és un home d'entre 35 i 60 anys amb nivell d'estudis mitjà – baix i amb coneixement de sistemes informàtics bàsic. Utilitza el mòbil principalment per trucades però també utilitza altres aplicacions ja sigui per comunicar-se, com pot ser WhatsApp, o per ús professional, com aplicacions d'informació climàtica o gestió de conreus.

Viu en un entorn rural i treballa al sector primari al camp o amb contacte amb animals. L'horari laboral del perfil de productor és flexible i, depenent del sector on treballa, varia en funció de l'època de l'any.

Fitxa Persona:

Albert Casals.

L'Albert és un agricultor de 53 anys que viu en un poble petit als afores de Lleida. Gestiona una explotació familiar de fruiters juntament amb la seva dona i un dels seus fills. L'Albert ha realitzat els estudis bàsics i va heretar la feina dels seus pares.

Fa ús del mòbil diàriament sobretot per trucar. Però també l'utilitza per missatgeria, correu electrònic i disposa d'algunes aplicacions útils de meteorologia i gestió dels conreus.

Dedica moltes hores a la feina però, per sort, durant els mesos que té que de recol·lectar la collita, disposa de temps suficient per fer el cafè al bar amb els companys i fer activitats amb la família. A l'estiu, en canvi, la càrrega de feina és major i aquests plaers són més limitats.

La major part de la seva producció la comercialitza a través de la cooperativa del poble però coneix un parell de botiguers de Lleida que en temporada li demanen material per la seva botiga. La cooperativa no li efectua el pagament de tots els productes aportats fins a finals de campanya (Setembre – Novembre) i la venda de petites quantitats de forma directa al comerciant l'ajuda en despeses habituals del dia a dia tot i que tampoc són prou estables per dedicar-hi més temps.

Consumidor

El perfil de consumidor és d'una persona adulta de qualsevol sexe amb estudis mitja – alts i mitjà – alt domini de sistemes informàtics i dispositius mòbils. Donat l'àmplia gamma d'usuaris diferents que entren en aquest perfil, la utilització del mòbil és molt variada però les principals utilitzacions són missatgeria instantània, cerca d'informació, jocs i com a navegador GPS.

Dins de tot el ventall de consumidors potencials, el perfil dels *Early Adopter* és un consumidor que valora el medi ambient, compromès socialment amb l'entorn i valora especialment la qualitat dels aliments i el menjar saludable per sobre del preu. Aquest usuari està disposat a anar a més d'una botiga per realitzar la compra, pagar una despesa extra d'enviament o desplaçar-se a buscar la cistella al lloc de producció per adquirir el producte que compleix les seves exigències.

Fitxa Persona:

Rosa Ferrer

La Rosa és una noia de 37 anys que viu amb la seva parella des de fa uns anys. Viu a Lleida, una ciutat de més de 100.000 habitants envoltada de petits pobles que, en general, es dediquen al sector primari. Ella treballa a l'administració en una plaça com a interina i disposa d'uns ingressos anuals per sobre dels 25.000 €.

La Rosa dona molta importància a la salut, per la qual cosa, procura anar sempre caminant a la feina i va periòdicament al gimnàs, almenys un parell de cops per setmana. També procura evitar menjar aliments processats i cuinar a casa sempre que li és possible. Sap que les matèries primeres que utilitza per cuinar són part important i intenta comprar productes de bona qualitat, a poder ser, de productors KM0.

Li agrada llegir articles i revistes sobre salut i cada cop més ho fa des del mòbil en lloc de publicacions en paper tradicional. Altres hobbies que té són trobar-se amb les amigues i viatjar, ja que li agrada molt conèixer noves cultures i menjars.

Actualment adquireix una cistella de productes ecològics d'uns agricultors de pobles veïns, però l'encàrrec es realitza per correu electrònic i s'ha de recollir els divendres a partir de les 19 h, cosa que molts cops li ha suposat un problema. L'altre inconvenient que hi veu és que els agricultors passen la llista de productes disponibles cada setmana però ella no sap l'estat d'aquests o que quan no hi ha del producte que volia, segons el producte, no el pot trobar enlloc més.

Relació rol – perfil d'usuari

Tots els usuaris, per defecte, poden consultar la llista de venedors i comprar productes, però el rol de Venedor, en un futur, s'adquirirà amb model de subscripció i no tots els usuaris disposen d'aquestes funcions.

Perfil d'usuari	Rol
Consumidor	Comprador
Productor	Comprador, Venedor

Taula 4. Relació rols - perfils d'usuari.

2.2. Contexts d'ús

2.2.1.Comprador

El consumidor utilitza l'aplicació de forma similar a la recopilació de la llista de la compra. Calcula els menús setmanals, de què es disposa i que falta. Habitualment, aquestes accions es duen a terme a casa o a la feina en moments de descans. Altrament, pot donar-se també el cas que, ja sigui perquè s'ha oblidat d'un producte o

per canvi de plans, etc., no disposi d'aquest temps i hagi de fer una cerca d'algun producte de forma urgent.

Es pot considerar que l'ús de PopFood per part del rol de comprador es realitzarà en estats de certa concentració i tranquil·litat però, en cas d'urgències es podria habilitar un registre dels productors preferits per tal d'accedir ràpidament al producte desitjat.

2.2.2. Venedor

El productor ja disposa d'una agenda de treball plena i no acostuma a disposar de llargues estones per gestionar les vendes per PopFood que, a més, poden arribar en qualsevol moment del dia. Però sí que disposa de petits moments al llarg del dia que li permeten consultar el mòbil, contestar algun correu o trucada o portar material a un punt de recollida.

El productor pot dedicar estones a PopFood de forma intermitent per la qual cosa seria convenient disposar d'un control de l'estat de les comandes de forma de consulta i modificació fàcil.

2.3. Escenaris

2.3.1. Consumidor

Escenari 1 – De vacances

La Rosa ha sortit de cap de setmana amb la seva parella. Aquest cop, han decidit anar al nord d'Espanya, per la zona de Cantàbria. Ja havia estat pel País Basc però encara no havia anat més enllà.

Passats els dos dies és hora de tornar cap a casa, ja que l'endemà torna a la feina. Els ha encantat el menjar de la zona i han pensat que podrien comprar alguna cosa per casa. Treu el mòbil i comprova que a la zona on són hi ha bastants ramaders i agricultors. Filtra la llista per productors de làctics i veu que en té un a poca distància on pot comprar formatge Cabrales i també sidra.

Com que han viatjat amb cotxe propi poden desplaçar-s'hi sense cap inconvenient. Segons la descripció, indica que en diumenge no hi ha hores de venda però decideix trucar-hi i aquest li diu que poden passar sense cap problema que ell és a casa tota la tarda.

D'aquesta manera poden marxar cap a casa amb un parell de productes de la terra per allargar l'experiència uns dies més a casa.

Escenari 2 – Menú setmanal

Aquest dissabte la Rosa té planejat un sopar amb unes amigues de tota la vida. Ja tenia previst el menú que sabia que les deixaria sense paraules però li ha sorgit un problema: El productor on compra habitualment no disposa del producte estrella del seu menú.

En un altre temps, no hagues tingut més remei que canviar el menú, ja que és un producte que no sol haver-ne en supermercats i, encara que coneixia un altre pagès que comercialitza productes ecològics, no disposa del telèfon ni sap quins dies es pot fer comanda.

Però gràcies a PopFood, localitzar un altre agricultor amb un bon producte no li ha estat gaire complicat. Com que encara hi és a temps, farà la comanda per recollir-la el divendres a un local que està prop del gimnàs on va. D'aquesta manera aprofitarà el viatge.

2.3.2.Productor

Escenari 1 – Nou producte a la venda.

Avui comença a collir el prèssec blanc. De bon matí, tot just han començat, ha agafat el mòbil i l'ha afegit a la llista de productes disponibles a PopFood. Ha afegit un parell d'imatges perquè es vegi que és un producte excel·lent, llàstima que no es pugui tastar a través del mòbil. Ha enviat una notificació als contactes que li han comprat algun cop per indicar l'actualització de la llista.

A l'hora de dinar veu que té pendents dues comandes de prèssec. Una la recollirà aquesta tarda en plegar i l'altra prefereix que li sigui entregada a la botiga del seu barri, una carnisseria que li fa el servei per un petit import entrega.

Un cop entregats els dos encàrrecs, rebrà els diners en l'intercanvi. No haver d'esperar a final d'any a rebre tots els diners, disposar d'una part dels ingressos en temps real i a millor preu li permet portar millor les despeses de l'explotació.

Escenari 2 – Preparació d'encàrrecs.

Avui l'Albert ha plegat més aviat de l'habitual. És dijous i ha de repartir encàrrecs pels diferents punts de recollida. Acostuma a preparar-los el mateix dia per tal d'assegurar-se que el producte que entrega es troba en perfectes condicions.

Ha de preparar dos encàrrecs per un dels punts i per dos punts més n'ha d'entregar un a cadascun. A més ha de dur la càrrega setmanal a una de les botigues la qual li compra fruita.

Per estalviar temps, prepara els encàrrecs simultàniament en sèrie producte a producte. Gràcies a l'aplicació, pot anar marcant quins productes ha preparat en cada encàrrec i saber que falta en cadascun.

Anteriorment, les botigues li enviaven la llista tant per correu electrònic com per telèfon i no era estrany que algun cop hi hagués errors, fos perquè s'equivocava el botiguer o l'Albert mateix. Amb *PopFood*, té el control clar de què hi ha a la llista, que ha preparat i que li falta.

Un cop entregats els encàrrecs als punts de recollida els marca com "Entregats" a l'aplicació per tal que el comprador sàpiga que pot passar a recollir-ho.

2.4. Casos d'ús

2.4.1.Consumidor

Cas d'ús	Autenticar-se
Actors	Usuari
Pre-condicions	L'usuari disposa de perfil vàlid de l'aplicació i coneix les credencials d'accés i es troba a la pantalla de login.
Disparador	Quan l'usuari ho decideixi.
Flux	1. Inicia l'aplicació. 2. Introdueix usuari i contrasenya. 3. Toca el botó de entrada.
Post-condicions	L'usuari ja ha entrat a l'aplicació i es troba a la pantalla de cerca de productors.

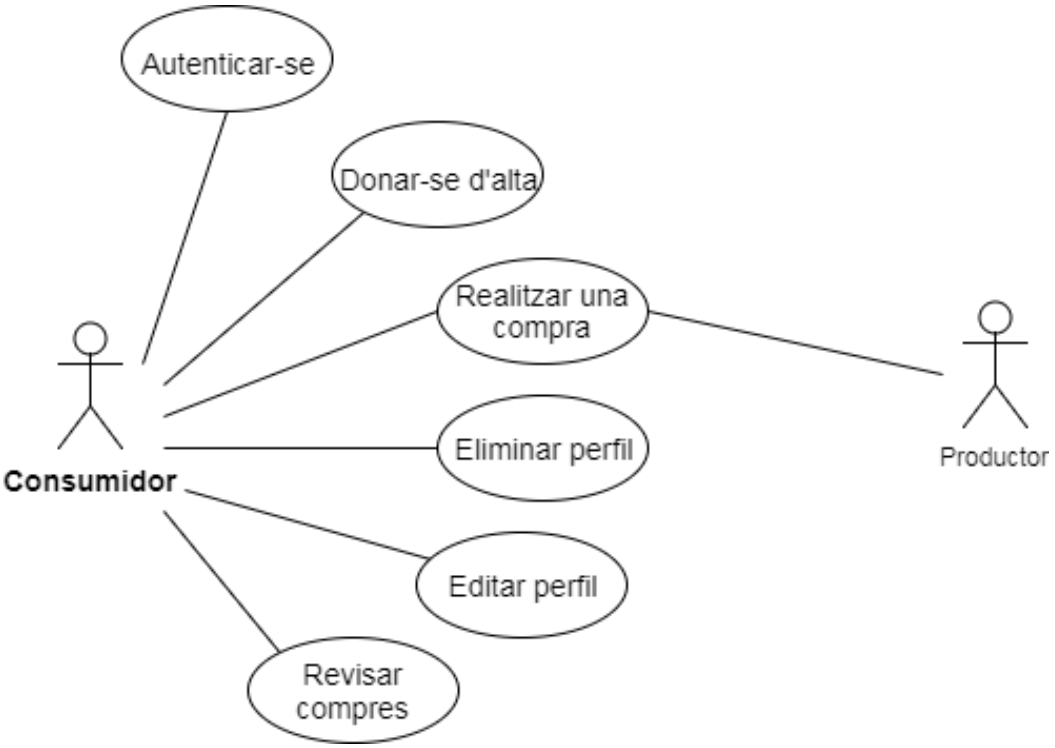
Cas d'ús	Donar-se d'alta
Actors	Usuari
Pre-condicions	L'usuari no ha de disposar de perfil a l'aplicació.
Disparador	Quan l'usuari ho decideixi.
Flux	1. Inicia l'aplicació. 2. Toca l'enllaç per donar-se d'alta. 3. Introdueix les dades. 4. Introdueix imatge. 5. Guardar.
Post-condicions	L'usuari disposa de perfil vàlid per l'aplicació.

Cas d'ús	Realitzar compra
Actors	Usuari i Productor
Pre-condicions	L'usuari s'ha autenticat i es troba a la pantalla de cerca de productors.
Disparador	Quan l'usuari ho decideix.
Flux	1. Busca un productor 2. Toca el productor i entra a la botiga d'aquest. 3. Afegeix productes a la cistella 4. Toca la cistella de la barra d'eines. 5. Comprova que la llista es correcta i toca el botó de triar hora. 6. Selecciona el mètode d'entrega i l'hora i toca el botó Continuar. 7. Revisa les dades de la compra i toca el botó finalitzar la comanda. 8. Acudeix al lloc i hora d'entrega acordat, recull la comanda.
Post-condicions	L'usuari disposa dels productes que ha comprat.

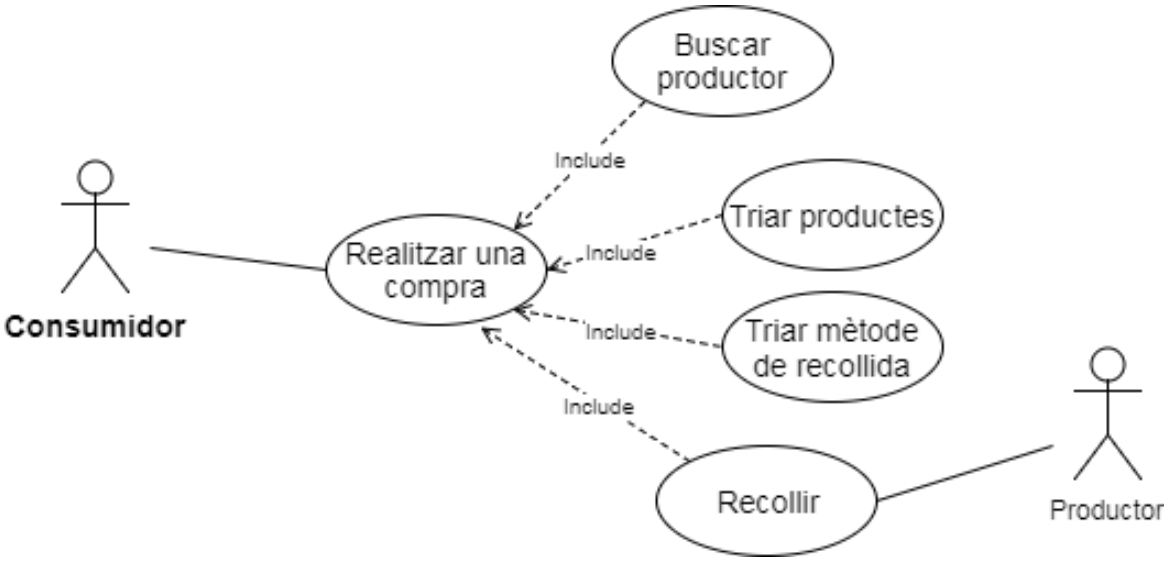
Cas d'ús	Eliminar perfil
Actors	Usuari
Pre-condicions	L'usuari ha de disposar d'un perfil vàlid i ha d'estar autenticat.
Disparador	Quan l'usuari ho decideixi.
Flux	1. Toca el botó de menú. 2. Toca l'enllaç a Usuari. 3. Toca el botó d'edició del perfil d'usuari de la barra d'eines. 4. Toca el botó d'eliminar perfil de la barra d'eines. 5. Confirma petició.
Post-condicions	El perfil de l'usuari ja no existeix i les credencials d'aquest no són vàlides.

Cas d'ús	Editar perfil d'usuari
Actors	Usuari
Pre-condicions	L'usuari ha de disposar d'un perfil vàlid i ha d'estar autenticat.
Disparador	Quan l'usuari ho decideixi
Flux	1. Toca el botó de menú. 2. Toca l'enllaç a Usuari. 3. Toca el botó d'edició del perfil d'usuari de la barra d'eines. 4. Modifica les dades o imatge que desitgi. 5. Toca el botó Guardar de la barra d'eines.
Post-condicions	L'usuari ha canviat les dades del seu perfil.

Cas d'ús	Revisar compra
Actors	Usuari
Precondicions	L'usuari ha de disposar d'un perfil vàlid i ha d'estar autenticat.
Disparador	Quan l'usuari ho decideixi
Flux	1. Toca el botó de menú. 2. Toca l'enllaç de compres. 3. Toca la compra que desitja revisar.
Postcondicions	L'usuari ja disposa en pantalla les dades de la compra que volia revisar.



Gràfic 3. Casos d'ús del perfil Consumidor



Gràfic 4. Cas d'ús Realitzar venda.

2.4.2.Productor

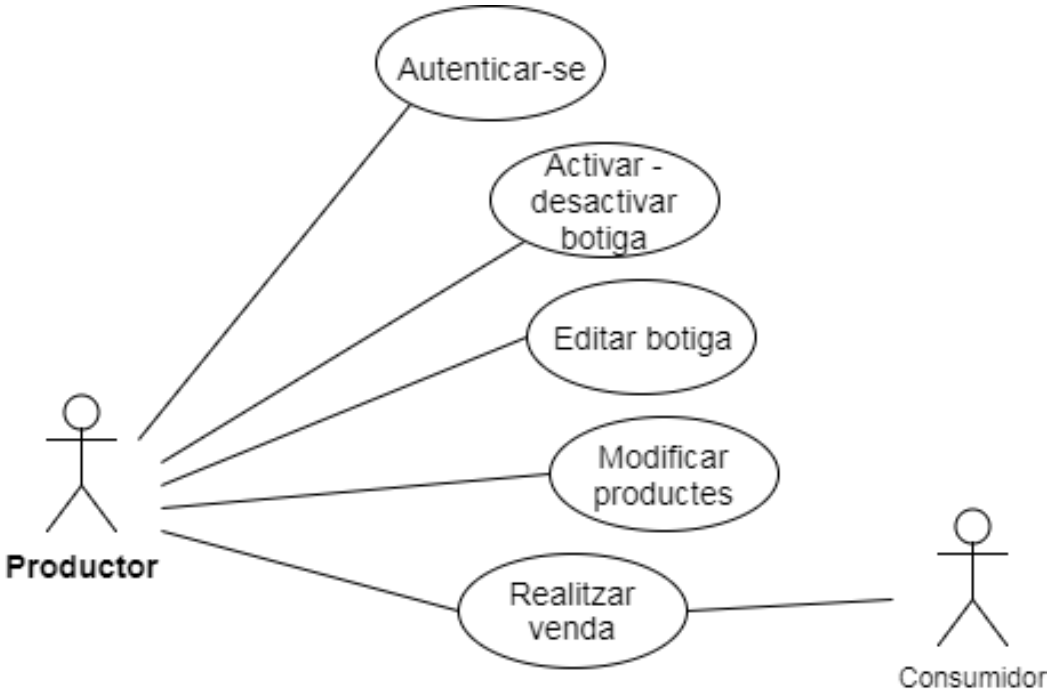
Cas d'ús	Autenticar-se
Actors	Usuari
Pre-condicions	L'usuari disposa de perfil vàlid de l'aplicació i coneix les credencials d'accés i es troba a la pantalla de login.
Disparador	Quan l'usuari ho decideixi.
Flux	1. Inicia l'aplicació. 2. Introdueix usuari i contrasenya. 3. Toca el botó de entrada.
Post-condicions	L'usuari ja ha entrat a l'aplicació i es troba a la pantalla de cerca de productors.

Cas d'ús	Activar – desactivar botiga
Actors	Usuari
Pre-condicions	L'usuari ha de disposar d'un perfil vàlid de productor i ha d'estar autenticat.
Disparador	Quan l'usuari ho decideixi
Flux	1. Toca el botó de menú. 2. Toca l'enllaç d'Usuari. 3. Toca la botó d'Editar de la barra d'eines. 4. Toca l'interruptor Activar botiga per activar o desactivar la visibilitat de la botiga. 5. Toca el botó de guardar
Post-condicions	L'usuari ha modificat la visibilitat de la botiga a la resta d'usuaris.

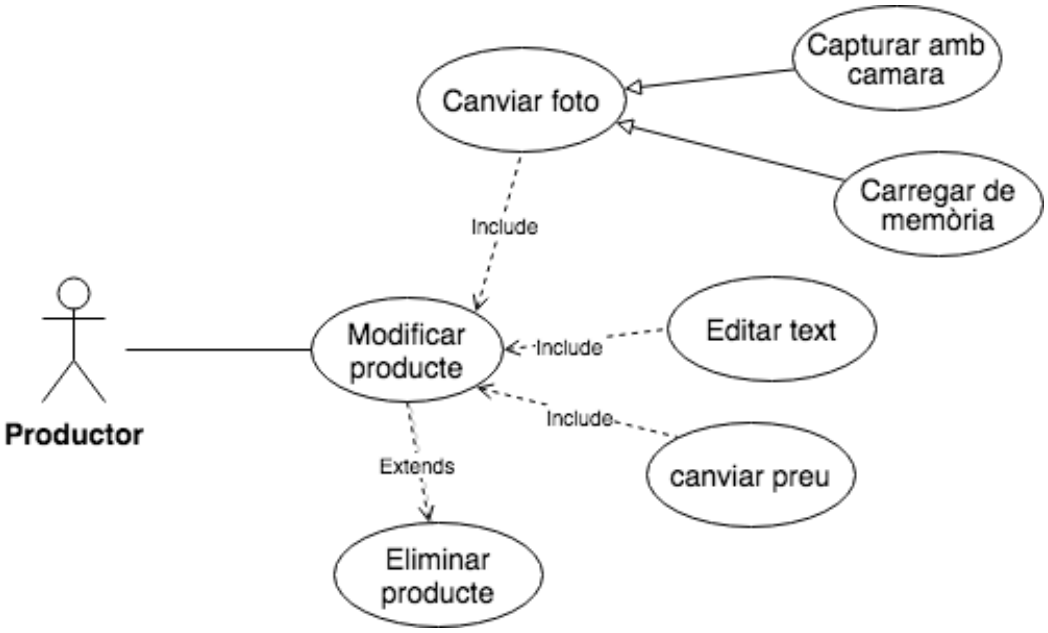
Cas d'ús	Editar botiga
Actors	Usuari
Pre-condicions	L'usuari ha de disposar d'un perfil vàlid de productor i ha d'estar autenticat.
Disparador	Quan l'usuari ho decideixi
Flux	1. Toca el botó de menú. 2. Toca l'enllaç de Botiga. 3. Toca el botó d'edició de la barra d'eines. 4. Modifica les dades desitjades. 5. Toca el botó de guardar de la barra d'eines.
Post-condicions	L'usuari ha modificat les dades públiques de la seva botiga.

Cas d'ús	Modificar productes
Actors	Usuari
Pre-condicions	L'usuari ha de disposar d'un perfil vàlid de productor i ha d'estar autenticat.
Disparador	Quan l'usuari ho decideixi
Flux **Flux**	1. Toca el botó de menú. 2. Toca l'enllaç de Botiga. 3. Toca la el botó d'edició del producte desitjat si es vol modificar un producte o el botó + per afegir producte. 4. S'introdueixen les dades desitjades. 5. Es toca la imatge si es vol substituir o el botó + per afegir més imatges. 6. Es tria el mètode de captura de la imatge i s'obté la imatge. 7. Es toca el botó Guardar de la barra d'eines.
Post-condicions	L'usuari ha afegit o modificat un producte de la seva botiga pública.

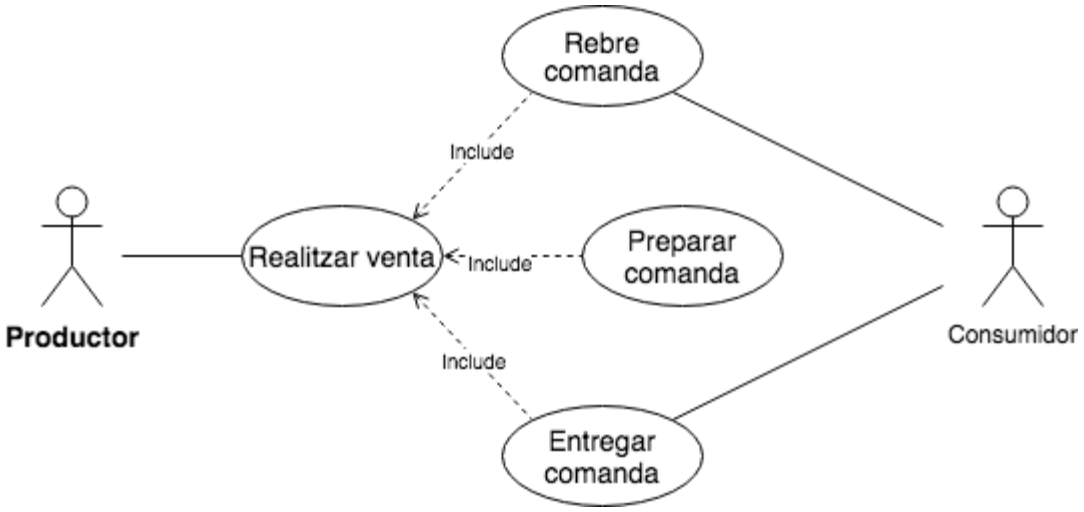
Cas d'ús	Realitzar venda
Actors	Usuari i Consumidor
Pre-condicions	L'usuari ha de disposar d'un perfil vàlid de productor, ha d'estar autenticat i ha de tenir la botiga activa.
Disparador	Quan l'usuari ho decideixi
Flux	1. L'usuari rep una notificació de nova comana o entra a partir del menú a l'enllaç Vendes. 2. Toca la venda rebuda. 3. Prepara els productes indicats marcant cada producte de la llista en el moment que l'ha preparat. 4. Un cop tots els productes estan preparats i sigui l'hora de l'entrega es dirigeix al lloc. 5. Entrega l'encàrrec.
Post-condicions	L'usuari ha entregat els productes de la llista.



Gràfic 5. Casos d'ús Productor



Gràfic 6. Cas d'ús Modificar producte.



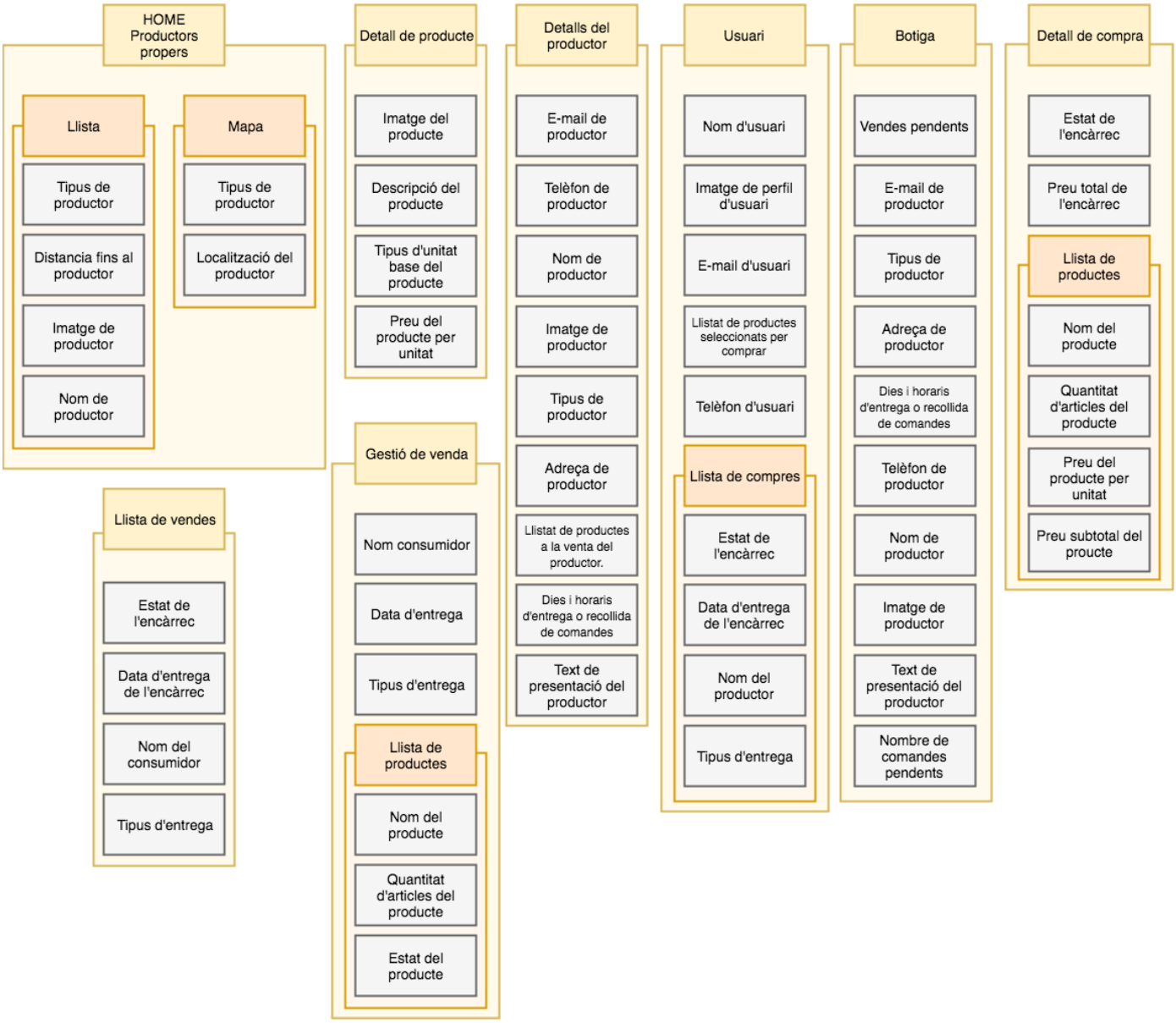
Gràfic 7. Cas d'ús Realitzar venda

2.5. Definició de funcionalitats

- Autenticació
- Donar-se d'alta com a usuari
- Modificar les dades del perfil d'usuari
- Donar-se de baixa com a usuari.
- Activació de la botiga virtual.
- Desactivació de la botiga virtual.
- Cerca de productors d'acord amb la localització del dispositiu.
- Possibilitat de filtrar els productors mostrats en pantalla per distància, tipus de producte i preferits.
- Visualització de les dades i productes disponibles del productor seleccionat.
- Realització de comanda.
- Visualització de les compres realitzades.
- Edició de dades de la botiga.
- Inserció d'imatge corporativa des de la càmera o des de la memòria del dispositiu.
- Inserció de productes a la venda.
- Inserció d'imatges als productes a la venda des de la càmera o des de la memòria del dispositiu.
- Edició de les dades dels productes a la venda.
- Visualització de les comandes rebudes.

3. Disseny

3.1. Arquitectura de la Informació



Gràfic 8. Distribució d'informació per interfícies

3.1.1.Punts d'interès

Separació de gestió de perfils.

Es preveu que els usuaris configurin dades de contacte per tal que en cas que realitzin alguna compra, el productor pugui contactar amb ells. Però, per mantenir la privacitat, s'ha optat per separar les interfícies de gestió del perfil d'usuari i de la botiga virtual

podent configurar dades de contacte diferents si s'escau. El perfil de botiga serà visible a tothom i el privat es considera millor que no.

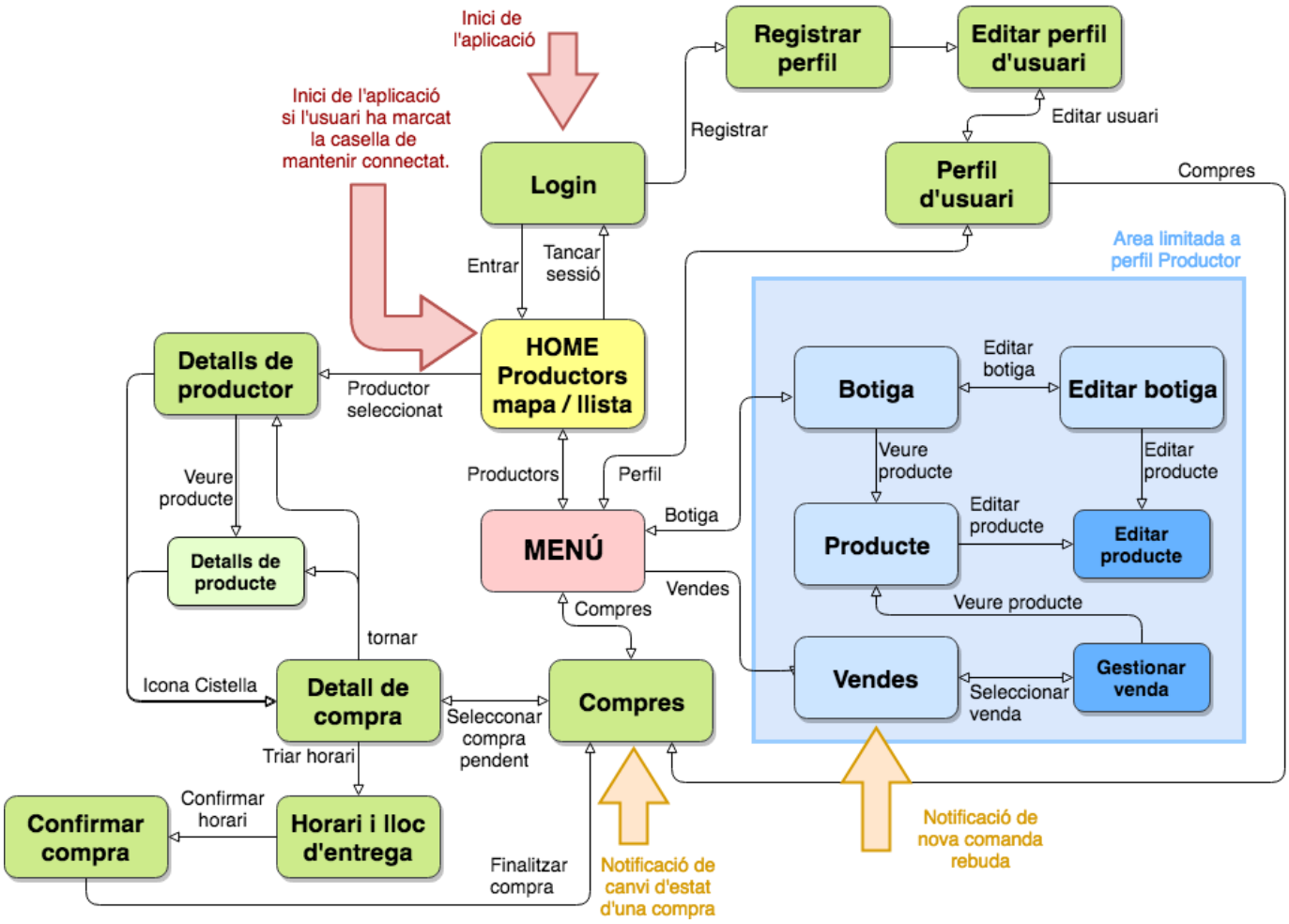
Llistat de compres de consumidors

S'ha optat per mantenir la llista de les últimes compres realitzades per l'usuari i habilitar una interfície també per veure la llista sencera. S'ha considerat que la probabilitat que un usuari tingui molts encàrrecs pendents d'entregar no és poc probable que es doni.

Modes de visualització de productors

La interfície de cerca de productors permet la visualització per llista i per mapa mostrant dades diferents en cada mode.

3.2. Arbre de navegació



Gràfic 9. Arbre de navegació de *PopFood*.

3.3. Definició d'interfícies i esbossos

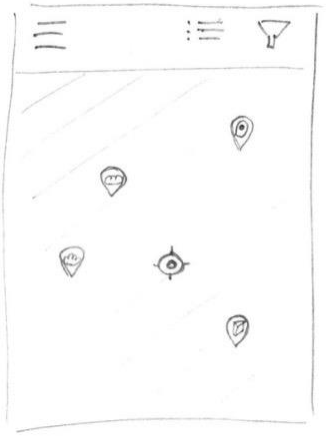
3.3.1.Estructura general de l'aplicació.

L'aplicació constarà d'un menú lateral desplegable amb la possibilitat d'accedir a totes les seccions principals de forma ràpida i clara. Paral·lelament alguns accessos importants es mostraran a les interfícies en forma d'icona segons l'interès en aquella secció.

Imatge 6. Prototip del menú desplegable

3.3.2.Cerca de productors.

Pantalla inicial de l'aplicació. Aquesta interfície ens mostrarà els productors propers en forma de llista o situats en el mapa, depenent del tipus de vista seleccionada. A la barra de tasques es mostrarà les icones del canvi de vista i de filtres (distància, preferits i tipus de productor). La informació mostrada en cada format és diferent i ens proporcionarà les dades necessàries per a aquella vista.



Imatge 7. Prototip del mapa de productors

Mapa.

Es mostrarà el mapa amb la ubicació actual de l'usuari al centre. En el mapa es mostraran també els diferents productors, amb la icona que correspon al tipus de producte que produeix, a la seva localització.

Llista.

La llista s'ordenarà per proximitat i per cada productor es mostrarà la imatge de perfil, el tipus de producte, la distancia, el nom del productor o empresa, tipus d'entrega de comanda i horaris de la pròxima entrega d'aquesta.

Imatge 8. Prototip de la llista de productors.

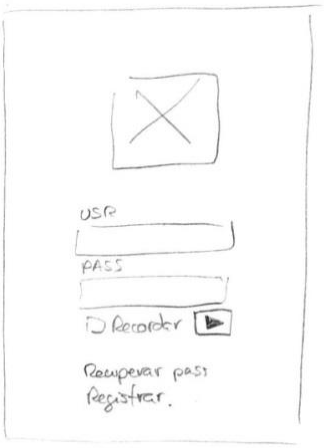
3.3.3.Botiga.

La interfície de botiga permet al propietari d'aquesta la comprovació de les dades que s'hi mostren. Aquí es mostra les mateixes dades que en la interfície de Productor amb la diferència que disposa d'accessos a l'edició de les dades i a la modificació dels articles a la venda. S'indica també el nombre de comandes pendents de preparar que té.

Imatge 9. Prototip de la vista de Botiga per part del productor

3.3.4.Login.

La interfície de *login* està formada pels elements habituals. Imatge corporativa, dos camps de text per introduir usuari i la contrasenya, *checkbox* per mantenir iniciat el compte, enllaç per donar-se d'alta i enllaç per recuperar la contrasenya.



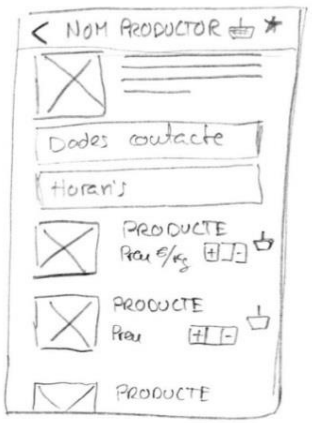
Imatge 10. Prototip de la vista Login.

3.3.5.Producte.

La interfície de detall del producte permet a l'usuari llegir la descripció d'aquest i veure les imatges disponibles. Les dades visibles seran el nom del producte, la descripció el preu i la unitat de mesura. També es podrà afegir a la cistella el nombre d'articles desitjats des d'aquesta interfície.

3.3.6.Productor.

En aquesta interfície s'ha de poder veure totes les dades del productor (imatge de perfil, nom de la botiga, telèfon, e-mail, adreça i text de presentació), localització, horari i un llistat dels productes dels quals disposa amb el preu, així com la possibilitat d'afegir a la cistella el nombre d'unitats indicada.



Imatge 11. Prototip de la vista de gestió del perfil d'usuari.

3.3.7.Perfil d'usuari.

La interfície de perfil d'usuari, comuna per tots els perfils, es mostrarà la imatge adjuntada, el nom, telèfon i e-mail. També contindrà un indicador de si disposa de botiga o no. Es mostrarà la llista de compres que l'usuari ha realitzat indicant el productor, la data i l'estat d'aquesta.

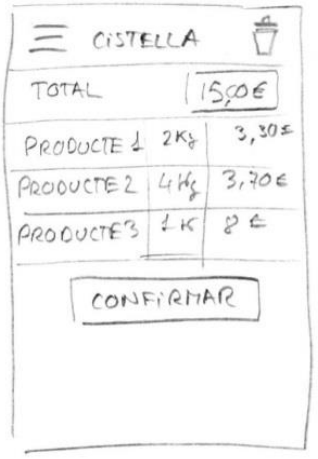
L'eliminació de perfil, modificació de dades i activació o desactivació de la botiga, tot i estar plasmats en l'esbós, es realitza en el mode d'edició de perfil.

Imatge 12. Prototip de la vista de la botiga per part del consumidor.

3.3.8.Cistella o Compra.

L'usuari pot accedir a aquesta interfície per comprovar els elements que hi ha dins una compra determinada. Serà visible el nom del Productor a qui va dirigida, el preu total, les dades d'enviament, si és que la compra s'ha formalitzat i estan definides (mètode d'entrega i data), i la llista d'articles que la componen indicant el nom, quantitat, preu unitari i total per articles.

En cas que la compra no estigui formalitzada, en el lloc de les dades d'entrega s'indicarà l'estat de l'encàrrec i hi haurà habilitats un botó Afegir articles per accedir directament a la botiga del productor i un altre per continuar amb la compra i triar hora i lloc d'entrega.



Imatge 13. Prototip de cistella

3.3.9.Selecció de tipus d'entrega

En aquesta vista es mostraran tots els mètodes d'entrega de la comanda per tal de seleccionar-ne un. Cada mètode d'entrega o horari diferent tindrà la seva pròpia entrada.

3.3.10. Llistat de vendes (productor)

Aquesta interfície mostrarà la llista de comandes que ha rebut el productor indicant el nom del comprador, l'estat (pendent, en preparació, enviat o entregat), la data límit d'entrega i el mètode. Es pot filtrar les comandes per data, estat i nom.

Imatge 14. Prototip de selecció d'horari

3.3.11. Gestió de venda

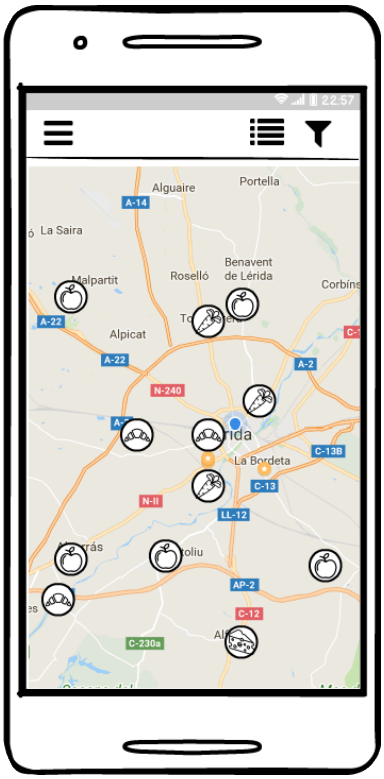
Aquesta interfície és similar a la que disposa el consumidor per revisar els articles de l'encàrrec. Disposa de la llista d'articles de la venda indicant el nom, la quantitat i l'estat que es troba (Es tracta d'una llista to-do on, en tocar l'element, canviarà d'estat automàticament). També serà visible el nom del comprador i les dades d'enviament.

Imatge 15. Prototip de la llista de Comandes

3.4. Prototipat

3.4.1.Interfícies

Interfície de *Login.*



Mapa de productors.

Llista de productors.

Menú desplegable.

Filtre de productors.

Vista d'un productor.

Detall de producte.

Edició de producte.

Llistat d'encàrrecs rebuts pel productor.

Vista de la botiga per part del propietari.

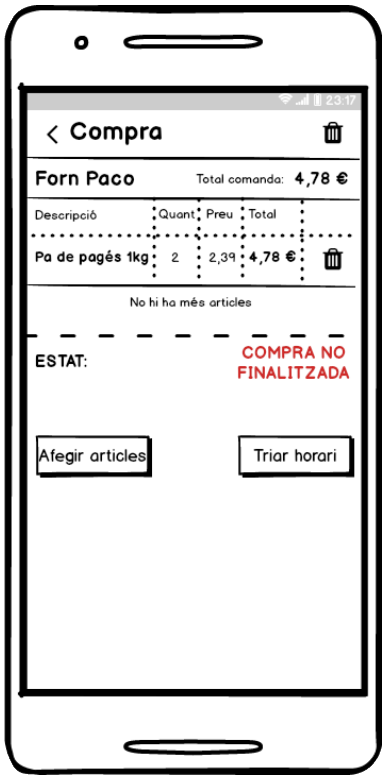
1[a] part de la interfície d'edició de la botiga.

2[a] part de la interfície d'edició de la botiga.

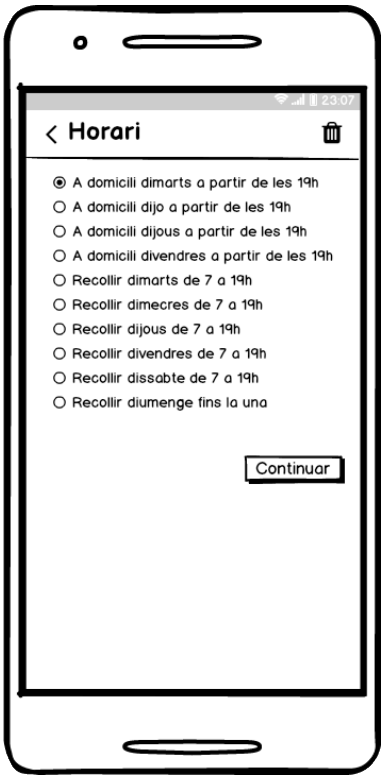
Vista del perfil d'usuari.

Edició del perfil d'usuari.

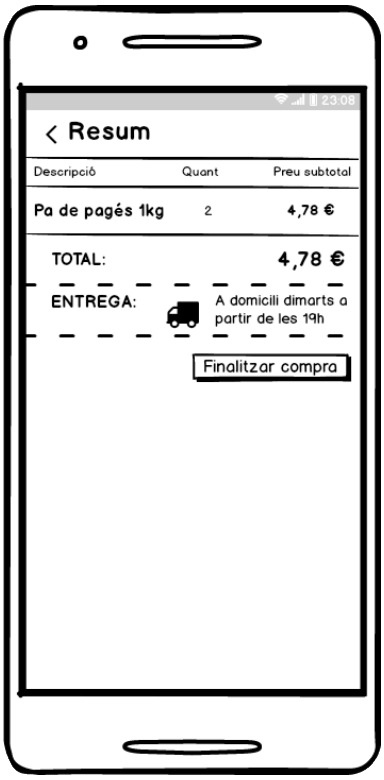
Nou usuari.



Detall de compra. En aquest cas no s'ha finalitzat la tramitació.



Selecció d'horari i mètode d'entrega.



Confirmació de compra.

Gestió d'encàrrec.

4. Implementació

4.1. Configuració de l'entorn de programació.

L'aplicació es realitza amb el framework Ionic per tal de simplificar la implementació i la generació del paquet .apk que es podrà instal·lar en dispositius Android.

Per tal de poder utilitzar Ionic s'ha d'haver instal·lat el servidor **Node.js** [9] que es pot descàrregar gratuïtament de la mateixa pàgina web. Aquest servidor és accessible des de la mateixa línia de comandes i du integrat el servidor de paquets **npm** [10]. És amb aquest últim que es realitzarà la descarrega del paquet de Ionic que ens permetrà començar a treballar amb ell.

Ionic [11] és un framework basat en **Angular 2** [12] i **Cordova** [13], que incorpora funcionalitats natives per dispositius **Android** i **iOS**. Ionic proporciona una capa d'estil similar a Bootstrap amb els mateixos conceptes de columnes i de classes predefinides. També ofereix diferents plantilles de projectes per crear una aplicació amb l'estructura de carpetes Ionic. A la mateixa pàgina web es proporciona documentació de referència i indica els primers passos a seguir.

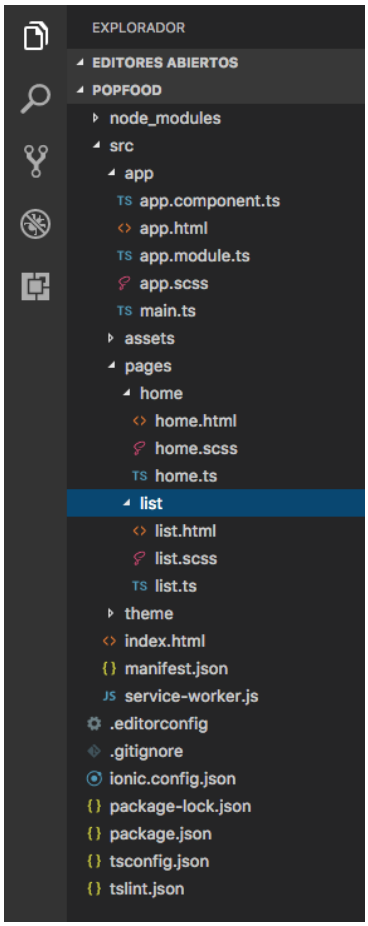
1. Amb Node.js ja instal·lat, es descarrega el mòdul de Ionic.

```
npm install -g cordova ionic
```

2. Un cop el mòdul està disponible ja es pot accedir a línia de comandes de Ionic. Pel desenvolupament de PopFood s'ha creat el projecte partint de la plantilla "sidemenu".

```
ionic start popFood sidemenu
```

3. Un cop finalitzat el procés de creació s'obté l'estructura de carpetes de l'aplicació que es pot veure a la Imatge 16.



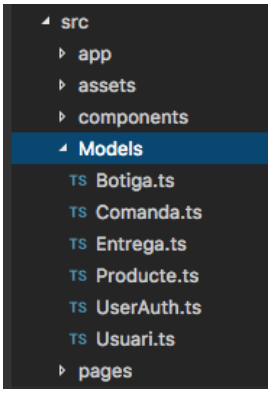
Imatge 16. Estructura de carpetes de l'aplicació creada amb la plantilla *sidemenu* de Ionic.

4.2. Estructura de dades.

Tal com s'ha definit a l'apartat d'Arquitectura de la Informació, es contempla entitats separades entre Usuari i Botiga per tal de preservar la privacitat de les dades dels

usuaris. A part d'aquestes dues, es crea les entitats Producte, Comanda, Entrega i Article. Posteriorment, es crea UserAuth que s'utilitzarà per l'autenticació amb Firebase [14].

Usuari. Inclou les dades bàsiques de contacte (nom, adreça, telèfon i correu electrònic), imatge de perfil i un llistat amb les compres realitzades (*array* d'objectes classe Comanda). No es crea cap propietat identificador, ja que s'utilitzarà l'identificador UID obtingut en registrar-se al servei d'autenticació de Firebase.



Imatge 17. Classes de la carpeta Models

Botiga. Inclou les dades bàsiques de contacte (nom, adreça, telèfon i correu electrònic), imatge de perfil, indicador de botiga activa, llistat de productes disponibles (*array* de Producte) i un altre llistat de vendes (*array* de Comanda). Tot i utilitzar com a identificador el mateix UID que per Usuari, en aquest cas sí que s'emmagatzema en una propietat que serà necessària en l'accés a les dades de la botiga per part dels compradors.

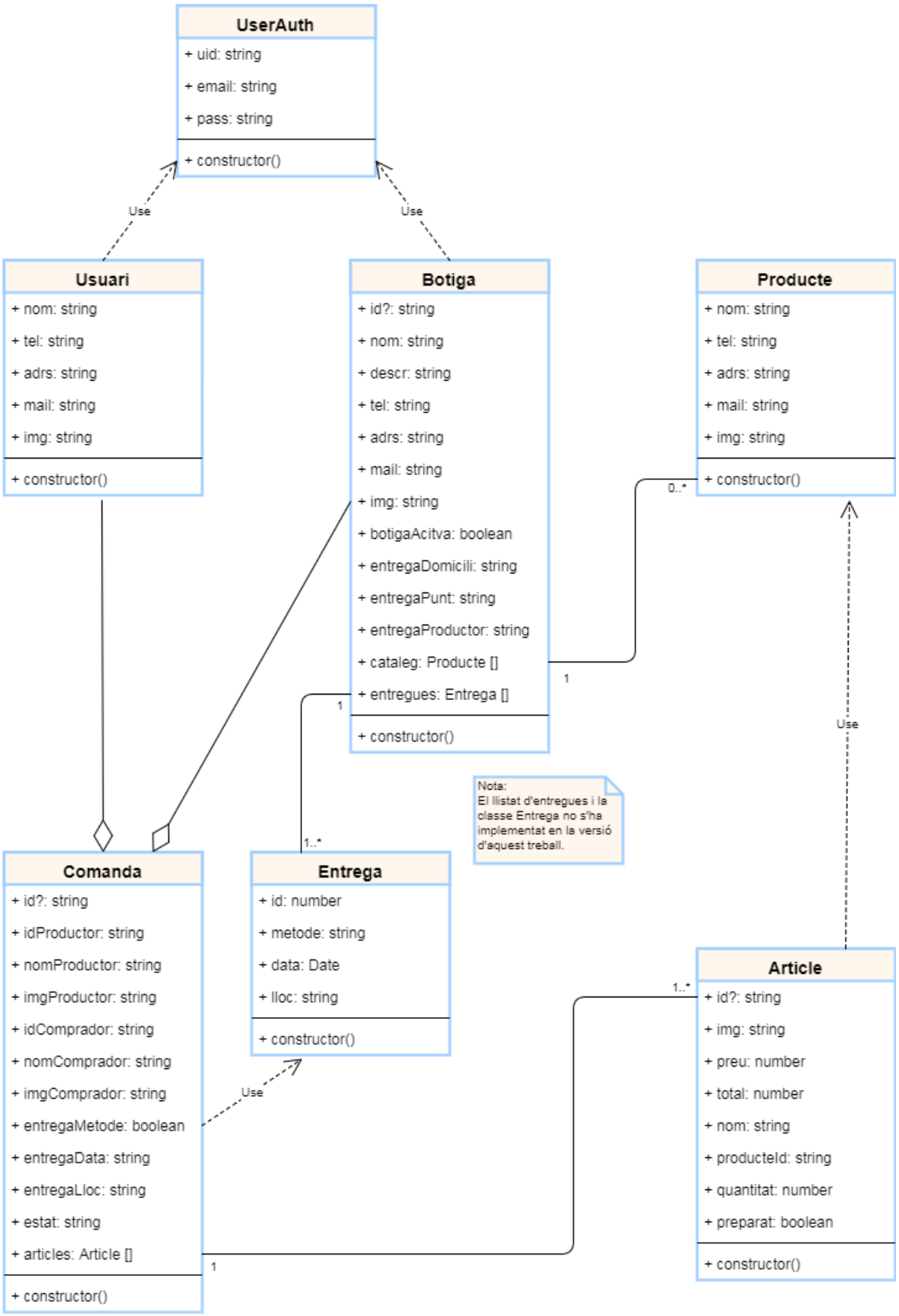
Comanda. La comanda és un objecte que enllaça les dues entitats anteriors, per tant ha de guardar informació referent als dos. A part ha de contenir també el llistat d'articles i les dades d'entrega.

Producte. El producte guarda el nom, descripció, preu i tipus d'unitat de venda del producte. També ha de contenir almenys una imatge d'aquest.

Article. Article és una classe que es crea inicialment per afegir, a la classe producte, la propietat `quantitat` però posteriorment, motivat per com es guarden les dades a Firebase, es manté com a classe replicant algunes dades obtingudes de la classe Producte.

Entrega. Es crea la classe Entrega, on es defineix el mètode d'entrega (recollir a casa del productor, punt d'entrega o domicili), el dia i l'hora, per tal que el propietari d'una botiga pugui definir les finestres temporals que el consumidor pot triar per fer l'entrega de l'encàrrec.

UserAuth. És una classe emprada per separar el perfil d'autenticació del perfil d'usuari. Conté solament correu electrònic, contrasenya i guarda el UID.



Gràfic 10. Diagrama de l'estructura de dades.

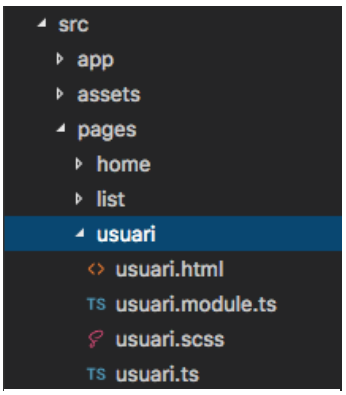
4.3. Interfícies i lògica.

4.3.1.Pages

Una pàgina en Ionic correspon a una interfície de l'aplicació i disposa del seu controlador, estil i vista. La generació de vistes a través de Ionic CLI [15] es realitza amb la següent comanda:

```
Ionic generate page Usuari
```

Un cop s'ha executat, s'ha generat la carpeta **Usuari** dins del directori **src/pages** amb quatre fitxers: **usuari.html**, **usuari.scss**, **usuari.ts** i **usuari.module.ts**. El fitxer **.module.ts** no és necessari realitzar-hi cap canvi. En el fitxer **.html** es definirà els elements que contindrà la pàgina i l'estructura que tindrà. En **usuari.scss** es definirien els estils que sol afectarien aquesta vista. En el codi desenvolupat en aquest treball s'ha prioritzat la funcionalitat de l'aplicació i no s'ha personalitzat, tret d'alguns petits detalls, cap estil i es manté el bàsic de Ionic. El fitxer **.ts** és el controlador de la vista codificat en Typescript. Aquest controlador contindrà tota la lògica que requerirà la vista.



Imatge 18. Arxius creats automàticament per Ionic CLI en generar una pàgina nova.

Per poder fer ús de la pàgina generada s'ha d'afegir al fitxer **app.module.ts** com s'indica a continuació:

```
import {UsuariPage} from '../pages/usuari/usuari;
@NgModule({
   Declarations: [
      UsuariPage,
      ...
   ],
   ...
   entryComponents: [
      UsuariPage,
      ...
  ]
})
```

Menú

La plantilla **sidemenu,** amb la qual s'ha generat el projecte PopFood, incorpora un menú lateral desplegable amb la icona d'hamburguesa per obrir-lo. Les entrades que consta aquest menú es defineixen a l'arxiu **.html** arrel (**app.html**) de l'aplicació, ja que el menú serà disponible en totes les vistes. El botó d'hamburguesa, en canvi, s'ha d'incorporar a cada vista a l'apartat **ion-header**.

Imatge 19. Menú lateral desplegable

Es pot trobar informació de com s'implementa el menú a la pàgina web de Ionic a *Components > Menu* [16].

En el cas de PopFood s'ha preferit crear un menú amb icones i això implicava crear una estructura de llista amb icona i text, tal i com es pot apreciar a la Imatge 19.

Ja que la vista s'implementa a **`app.html`**, la lògica corresponent també s'implementa al controlador arran de l'aplicació, **`app.component.ts`**. Si es toca un dels elements del menú (excepte Tancar sessió i Sortir) es crida la funció **`openPage(...)`** i es carregarà la pàgina seleccionada. Si l'opció seleccionada és Tancar sessió, la funció executada és **`tancarSessio(...)`** i si és Sortir, serà **`exitApp(...)`.**

Vistes Usuari, Botiga i Productor.

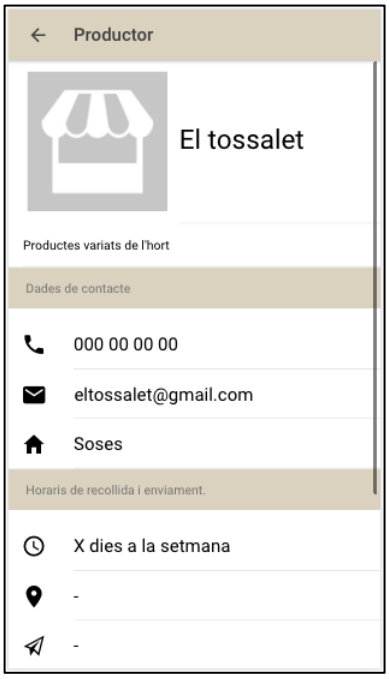
Les tres vistes es creen amb igual estructura però amb petites variacions. Tota l'aplicació es crea amb estructura base de llista amb l'etiqueta **`<ion-list>`** on cada fila correspon a un element **`<ion-item>`.** Algunes seccions s'identifiquen amb un títol que correspon a l'etiqueta **`<ion-list-header>`.**

La interfície de perfil d'usuari conté un element en la qual s'indicarà l'estat de la botiga. Botiga i productor mostren un element que mostrarà el text de descripció i d'una secció on s'indicarà resumidament els mètodes d'entrega que ofereix la botiga.

Les interfícies de Botiga Personal i Productor son iguals visualment però al integrar la persistència a Firebase l'obtenció de dades passa a ser asíncrona i, al tractar-se d'una versió en construcció, el component utilitzat per la llista de productes és el mateix (**`fila-producte`**). La intenció però, era crear un component diferent que incorporés els botons per gestionar l'element en la cistella.

Imatge 20. Vista Perfil d'usuari.

Imatge 21. Vista de gestió de la botiga.



Imatge 22. Vista de la botiga des del punt de vista del client.

Les dades mostrades en Perfil d'Usuari i de Botiga són dades pròpies i, per millorar el rendiment, estan disponibles al servei des del moment que s'inicia l'aplicació i es mostren directament.

```
<!-- IMATGE I NOM DE BOTGA -->
<ion-item>
  <img col-5 src="{{ botigaStorage.botiga.img }}" item-start>
    <h1 text-wrap>{{ botigaStorage.botiga.nom }}</h1>
</ion-item>
<ion-item text-wrap>
  {{ botigaStorage.botiga.descr }}
</ion-item>
```

En canvi, les dades de la vista de Productor es reben de forma parcialment asíncrona com s'ha comentat. Les dades de la botiga es reben per paràmetre **`navParam`** des de la vista pare i s'hi accedeix igual que el cas de Botiga, però no així el catàleg que s'ha d'obtenir directament del servidor de forma asíncrona afegint " **`| async`**" a la lectura de l'objecte **`Observable<Producte[]>`**.

```
<ion-item *ngIf="catalegRebut">
    <fila-producte-botiga
      *ngFor="let producte of catalegRebut | async"
      (click)="veureProducte(producte)" [createIcon]="false"
      [nom]="producte.nom" [img]="producte.img" [preu]="producte.preu"
      [unitat]="producte.unitat">
    </fila-producte-botiga>
</ion-item>
```

Producte i Producte-detall.

La vista del producte consta d'una imatge gran a la part superior que permeti al client apreciar detalls d'aquest. Per tal que la imatge aprofités l'ample de la pantalla es defineix l'element **`img`** amb la propietat **`col-12`**.

Les altres dades visibles en format de llista són el nom del producte, la descripció, el preu i la unitat de venda.

Producte-detall es crea per ser accessible pel públic i constarà del component que permet seleccionar la quantitat d'articles i afegir-los a la cistella mitjançant la funció **`afegirArticle(...)`** del servei.

El preu del producte es modifica per tal de mostrar la moneda i limitar els decimals a dues xifres amb l'aplicació de filtres al valor de la propietat.

```
{{ producte.preu | currency: 'EUR':true:'.2-2' }}
```

Les dades mostrades en aquestes vistes s'obtenen de l'objecte producte del controlador que sempre es rep per paràmetre a l'iniciar la vista.

Imatge 23. Vista de producte des del punt de vista del client.

Compres i Vendes

Aquestes dues interfícies únicament contenen una llista de les comandes rebudes del servidor corresponents a l'usuari / botiga. Per mostrar les dades s'utilitza el component Fila-venda que es detallarà en la secció 4.3.2 Components d'aquest document.

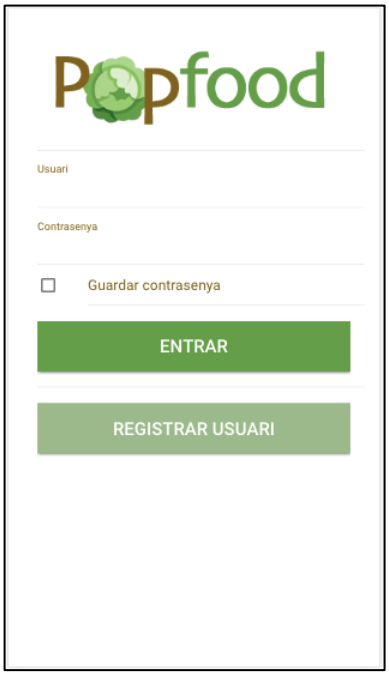
List i Home.

Són les interfícies de cerca de productors però per limitacions en el termini de la realització del treball es quedaran com a treball futur. Solament s'ha implementat una versió bàsica a la pàgina List on es mostra una llista de les botigues indicant el nom per poder testejar les comandes i la visualització de les interfícies.

L'obtenció de les dades es realitza de forma asíncrona i si es toca algun dels elements de la llista, es crida la funció **`itemTapped(...)`** que obre la interfície Productor passant la botiga seleccionada per paràmetre.

Login i Registrar

Les interfícies Login i Registre consten de dos camps de text editables (*input*) amb etiqueta (*label*) corresponents a correu electrònic i contrasenya, un *checkbox* per recordar les credencials en propers inicis i els botons Entrar i, en el cas de la interfície de login, Registrar per a la vista de registre de nous usuaris.



Imatge 24. Pàgina de Login.

Els camps del formulari s'enllacen amb la variable **`userAuth`** del servei **`AuthProvider`** mitjançant la propietat **`[(ngModel)]`** que proporciona Angular. El *checkbox* s'enllaça amb una variable local **`mantenirConnectat`**, ja que no s'empra enlloc més.

El botó Entrar executa la funció **`checkAuth()`** que s'encarrega de realitzar la crida a la funció del servei encarregada de connectar amb el servidor Firebase i, de forma asíncrona, gestiona la resposta.

No s'ha implementat cap tipus de validació en els camps d'entrada, únicament s'implementa un *Toast* indicant l'error rebut de Firebase o del mateix formulari.

Compra.

Aquesta interfície mostra les dades de la compra seleccionada a la llista de Compres o cistella si s'hi ha afegit algun article. S'ha construït amb uns elements permanents a la part superior on es mostra el nom del

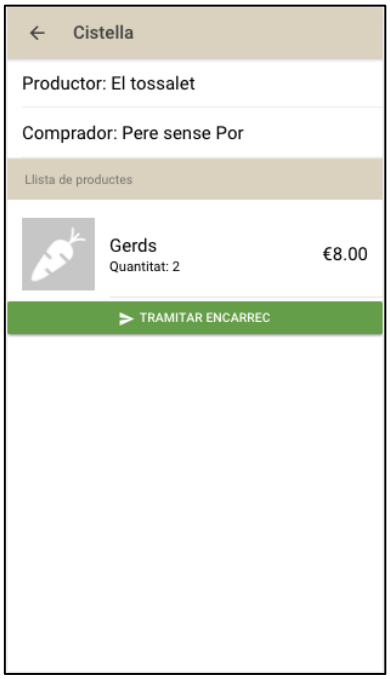
Productor, el nom del Consumidor i, en funció de si es mostra un encàrrec registrat o la cistella, les dades de l'entrega (tipus d'enviament, direcció, dia i hora) i una part dinàmica generada amb un **ngFor** on es visualitza els articles.

El booleà **enCistella** determina quan es tracta de la cistella. En el moment que s'afegeix la Comanda a Firebase es genera automàticament l'**id**, en el qual es basa el càlcul del valor d'**enCistella**.

En cas que s'estigui mostrant la cistella s'obtindran les dades dels articles directament del servei **ComandesProvider**, s'inclourà un botó per continuar la tramitació de la compra i, donat que encara no s'han definit, no es mostraran les dades d'entrega de la comanda.

En canvi, si es tracta d'una compra registrada a Firebase, s'ha d'obtenir els articles de forma asíncrona i no es mostra el botó

Per seleccionar la vista corresponent en cada cas s'utilitza la funció **ngIf** d'Angular com es pot observar en les següents imatges.



Imatge 25. Cistella

```
<!-- LLISTA DE FIREBASE -->
<ion-list *ngIf="!enCistella">
  <ion-item  *ngFor="let article of articlesRebuts | async" >
```

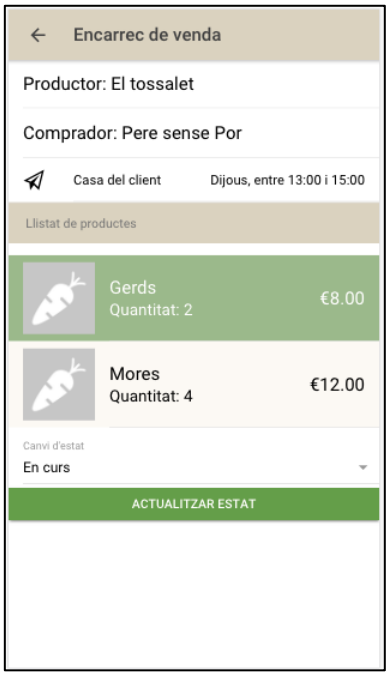
```
<!-- LLISTA DE CISTELLA -->
<ion-list *ngIf="enCistella">
  <ion-item *ngFor="let article of comanda.articles" >
```

Venda

La interfície Encàrrec de venda mostra les dades d'una venda seleccionada a la llista de Vendes. Es mostra el nom del productor, el nom del consumidor, les dades d'entrega, la llista d'articles i un desplegable on seleccionar el nou estat juntament amb un botó per confirmar el canvi.

En aquesta interfície és on el Productor ha de fer els canvis en la comanda que després el consumidor veurà. A mesura que el productor té llestos els articles ha de tocar-lo a la pantalla i aquest es marca com a preparat. Els articles no es poden marcar si la comanda no es troba en estat "EN CURS" i, tot i que no s'ha pogut implementar encara, no es pot canviar d'estat si queda algun article pendent de marcar.

Les dades de la comanda es reben de la vista anterior amb **navParams** però els articles s'han de demanar a



Imatge 26. Vista de gestió de venda.

Firebase amb la funció del servei **`ComandesProvider`** **`recuperarArticlesComandaFirebase(...)`** i es mostren de forma asíncrona.

La llista d'articles obtenen el valor de la propietat color dinàmicament executant la funció **`getColor(...)`**. Depenent del valor (booleà) de la propietat **`preparat`** de l'article mostra un color diferent. Si es toca l'article, s'executa la funció **`articlePreparat(...)`** que actualitza l'article a *Firebase* que automàticament es modifica a la vista.

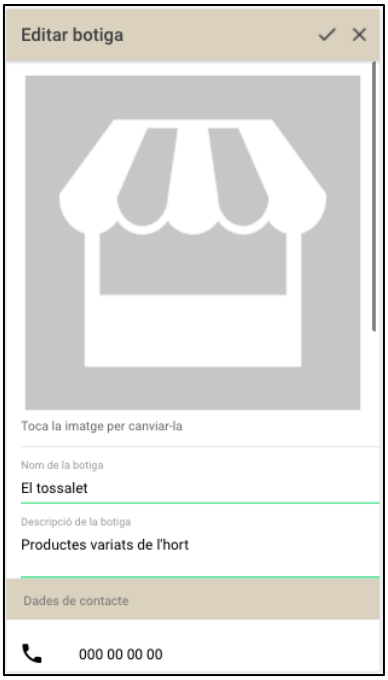
Vistes d'edició.

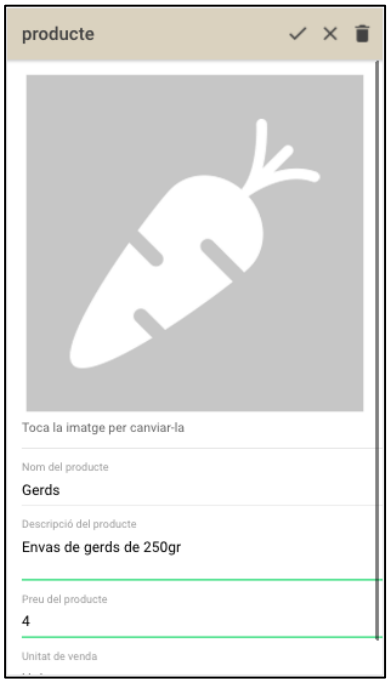
Les vistes d'edició han mantingut en tots els casos la mateixa estructura modificant els elements on es mostra el text per elements editables. **`<ion-input>`**, **`<ion-toggle>`**, **`<ion-select>`**, etc.

En el cas d'Usuari-Editar i Botiga-Editar, es realitza una còpia de l'objecte guardat en el servei corresponent i es realitzen les modificacions en aquest. Si l'usuari decideix declinar els canvis, es torna a la vista anterior sense haver modificat res però si els accepta, es modifica els objectes afectats dels serveis i també a Firebase.

En la pàgina d'edició de l'usuari s'ha desactivat la possibilitat de modificar el correu electrònic perquè és el mateix amb el qual l'usuari s'autentica. En futures actualitzacions es preveu poder modificar aquest camp realitzant el canvi a la base de dades d'autenticació també.

En el cas de Producte-Editar s'hi pot accedir des de dues vistes, Producte i Botiga-Editar. En ambdós casos es passa per paràmetre a la vista d'edició el Producte i la pàgina d'on es realitza la crida per tal de conèixer l'origen un cop estiguem a la nova vista. Si es realitza la crida a la vista amb el botó de crear nou producte de la interfície d'edició de la botiga, el producte passat per paràmetre és un producte en blanc.





Imatge 28. Vista d’edició de la botiga.

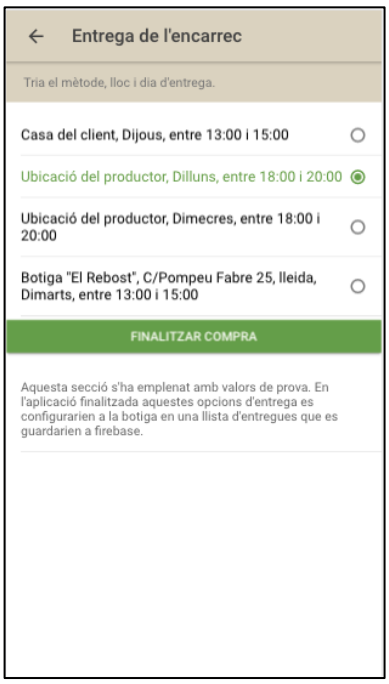
Imatge 27. Vista d'edició de producte

Entrega-select

La interfície **`Entrega-select`** forma part el procés de realització d'una comanda. Un cop el consumidor ha ficat tots els productes a la cistella i continua la tramitació, es carrega aquesta vista per tal de triar quina finestra prefereix per rebre els productes.

Aquesta és una interfície que quedava fora de la previsió d'implementació per aquest treball però per tal de poder realitzar comandes de prova s'ha implementat una llista d'opcions fictícia.

En condicions reals, l'usuari ho configuraria a la Botiga. A part dels resums mostrats al consumidor que la visiti, s'hauria de fer una llista de finestres disponibles en el període d'una setmana.



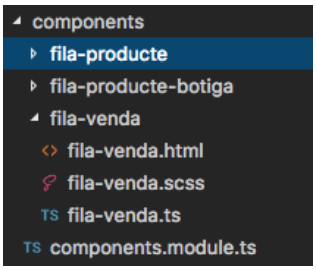
Imatge 29. Selecció d'entrega en la realització d'una comanda.

4.3.2.Components

Un component és una part de codi d'interfície que podem reutilitzar. A l'igual que les pàgines, els components disposen del seu controlador, estil i vista. Els components en Ionic es poden generar per línia de comandes amb la següent sentència:

```
Ionic generate component nomComponent
```

Aquesta generarà, dins de la carpeta Components, una carpeta amb el nom del component i amb els fitxers habituals **`.html`**, **`.scss`** i **`.ts`**. En aquest cas, es genera un mòdul dedicat dins de la carpeta Components on es registrarà automàticament els components que generem.



Imatge 30. Estructura de la carpeta components

Per tal de poder passar dades de la classe pare cap al component, es declara, a la definició del controlador, la llista de propietats que el controlador rep des de fora:

```
@Component({
  selector: 'component-exemple',
  inputs: ['variable1','variable2', ... ],
  templateUrl: 'component-exemple.html'
})
export class ComponentExemple { ...
```

Un cop s'utilitza el component, es passa els valors que es vol que tinguin aquestes propietats:

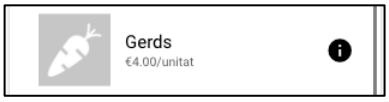
```
<component-exemple
         [variable1]= "valor1" [variable2]= "valor2" ... >
</component-exemple>
```

Per poder utilitzar el component en l'aplicació s'ha de registrar al fitxer **app.module.ts** a la llista de Components.

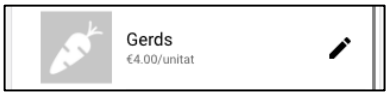
```
import {ComponentExemple} from
         '../components/component-exemple/component-exemple;
@NgModule({
   Declarations: [
      ComponentExemple,
      ...
   ],
   ...
   entryComponents: [
      ComponentExemple,
      ...
  ]
})
```

Fila-producte

Element utilitzat per mostrar els productes del catàleg de Botiga i Botiga-Editar. Mostra la imatge, el nom del producte, el preu i unitat de venda. Es passa un input indicant si la icona a afegir al final de la fila és una icona d'edició o d'informació, ja que a la vista Botiga-Editar, tocar el component significa editar-lo i, a la vista Botiga sols es mostra la vista detall del producte.



Imatge 32. Component fila-producte amb icona de informació



Imatge 31. Component fila-producte amb icona d'edició.

Fila-producte-botiga

El component fila-producte-botiga es crea amb la intenció d'afegir a l'element l'opció d'afegir aquell producte a la cistella sense necessitat d'entrar al detall del producte. Aquest punt queda com a treball futur, ja que no ha estat possible implementar-ho a temps i en aquest moment és exactament igual que el component anterior fila-producte.

Fila-venda

Element utilitzat en les llistes de comandes Vendes i Compres. Mostra el nom, imatge, la data d'entrega, el lloc i l'estat de la comanda. Quan s'utilitza per compres es passa el nom i la imatge del Productor i quan és per vendes es passa nom i imatge del Consumidor.

Imatge 33. Component fila-venda.

4.4. Serveis

La creació d'un servei o proveïdor en Ionic CLI es genera amb la següent sentencia:

```
ionic generate provider Exemple
```

En el cas dels Providers, Ionic genera únicament l'arxiu .ts, ja que per un servei actua com a proveïdor de dades i no compta amb interfície visual. Per tal de poder actuar com a Provider, la classe ha de dur l'anotació **`@Injectable()`** i importar el mòdul **`HttpClient`** de **`@angular/common/http`**.

S'ha donat el cas que la generació via CLI del servei no generava tot el codi necessari i en executar l'aplicació es rebia un error de **`staticInjection[HttpClient].`** Després de comprovar la documentació es detecta que manca la injecció del mòdul **`HttpClientModule`** a l'arxiu **`app.module.ts`**. Un cop generat el servei s'ha de registrar a la llista de dependències del projecte a **`app.module.ts`** amb el següent codi:

```
import {ExempleProvider} from '../providers/exemple/exemple;
@NgModule({
  ...
  providers: [
    ExempleProvider,
    ...
  ]
})
```

4.4.1.StorageProvider

`StorageProvider` és el servei dedicat a gestionar les dades del perfil d'usuari. El servei manté una instància del model **`Usuari`** de forma que les dades d'usuari adoptin un format *singletone*. A part, s'ha implementat tres mètodes de connexió amb Firebase i una altre de conversió de dades d'**`Observable<Usuari>`** a **`Usuari`**.

`crearUsuariFirebase()` crea un usuari en blanc a la base de dades de Firebase. Es cridada únicament quan es registra un usuari nou a la pàgina **`Registrar`**.

`updateUsuariFirebase(Usuari)` actualitza l'usuari guardat a Firebase amb les dades de l'**`Usuari`** rebut per paràmetre.

`recuperarUsuariFirebase()` obté de la base de dades Firebase les dades de l'usuari emmagatzemat retornant un objecte asíncron **`Observable<Usuari>`**.

`recuperarDadesPerfil(Observable<Usuari>)` assigna les dades de l'Observable a la instancia d'**`Usuari`** del servei. Aquest canvi es realitza transformant l'objecte tipus **`Observable`** a **`Promise`** i guardant les dades a l'Usuari, un cop completada la promesa. La funció **`first()`** hi és perquè l'observable rebut és en realitat una llista però que sempre serà d'un únic element.

```
async recuperarDadesPerfil(usuariRebut: Observable<Usuari>){
   try {
      usuariRebut.first().toPromise().then(
        value => {
          this.usuari = value;
```

```
            console.log('Dades recuperades');
          }, error => {
            this.usuari = new Usuari;
          }
        );
      } catch (err) {
        console.error(err);
      }
    }
```

`capturarFoto()` Crida la funció de captura d'imatge nativa i assigna la imatge rebuda al camp img de la instància d'usuari del servei.

`guardarImatgeFirebase(referència)` Guarda la imatge a Firebase sobre la referència rebuda per paràmetre i retorna la resposta al controlador.

Les funcions **`recuperarUsuariFirebase()`** i **`recuperarDadesPerfil()`** hagueren pogut implementar-se en una única funció però atès els canvis que s'ha anat fent al codi s'ha optat per no modificar-les i utilitzar les dues funcions juntes. Inicialment, les dades de l'usuari es rebien i es mostraven directament de forma asíncrona en totes les interfícies. Una dificultat que va sorgir va ser que per editar les dades d'usuari a **`Usuari-Editar`**, els camps del formulari s'han d'enllaçar a un objecte **`Usuari`** del controlador per després actualitzar-ho a Firebase. El problema és que no es poden mostrar les dades rebudes al formulari per ser un Observable.

La primera solució adoptada va ser crear la funció **`recuperarDadesPerfil`** cridada en el moment que l'usuari accedia a la pàgina Usuari. Un cop s'accedís a la pàgina d'edició del perfil, el servei disposaria de les dades al *Provider* i es mostrarien al formulari.

Finalment es decideix realitzar l'obtenció de les dades del perfil un cop l'usuari s'autentica a l'aplicació cridant les dues funcions juntes per dos motius. Un dels motius va ser perquè es necessiten les dades de l'usuari per crear una comanda, cosa que pot fer-se sense haver accedit abans a les dades del perfil, i un altre motiu era perquè es milloraria el rendiment de la interfície Perfil d'usuari.

4.4.2.BotigaProvider

Aquest servei s'encarrega de gestionar les dades corresponents a botigues. Les dades de la botiga personal es gestionen de forma molt similar a les de usuari, explicades a la secció anterior StorageProvider, però també es crida les funcions d'aquest servei per mostrar dades d'altres botigues quan es visita la botiga d'un altre usuari per realitzar una comanda.

Els mètodes **`crearBotigaFirebase`**, **`updateBotigaFirebase`**, **`recuperarBotigaFirebase,`** **`recuperarCatalegFirebase`** i **`recuperarDadesBotiga`** realitzen les funcions equivalents de StorageProvider amb el detall que aquesta última realitza la conversió tant de les dades de botiga com del catàleg de productes d'aquesta.

Tal com s'explica en la secció

Base de dades Firestore, un document només permet guardar llistes de tipus bàsics, no d'objectes. La manera que s'ha utilitzat per guardar el llistat de productes és afegir una altra col·lecció dins del document de la botiga. Per simplificar el codi, s'opta per

separar l'obtenció de les dades de botiga i del catàleg. Aquest fet també ens millorarà el rendiment a l'obtenció del llistat de botigues visibles a l'usuari (**`ListPage`**), ja que no es descarregaran dades del catàleg fins que es seleccioni una botiga.

`obtenirBotiguesFirebase()` retorna la llista completa de botigues que hi ha a Firebase i que estan actives (**`botigaAcitva`**).

`updateBotigaActivaFirebase()` modifica l'indicador **`botigaActiva`** de la botiga.

`afegirProducteCatalegFirebase(Producte)` afegeix el producte rebut per paràmetre a la col·lecció catàleg de la botiga.

`eliminarProducteDeCatalegFirebase(Producte)` elimina el producte rebut per paràmetre de la col·lecció catàleg de la botiga.

`actualitzarProducteDeCatalegFirebase(Producte)` actualitza el producte amb **`id`** igual al rebut per paràmetre amb les dades d'aquest.

`capturarFoto()` Crida la funció de captura d'imatge nativa i assigna la imatge rebuda al camp img de la instància d'usuari del servei.

`guardarImatgeFirebase(referencia, imatge)` Guarda la imatge a Firebase sobre la referència rebuda per paràmetre i retorna la resposta. A diferència del servei de gestió de l'usuari, es passa per paràmetre la imatge que es vol emmagatzemar perquè s'utilitza la mateixa funció per Botiga i per Producte. El servei de Usuari obté la imatge directament de la instància d'usuari del d'aquest, però en aquest cas se l'hi ha d'indicar quina és la imatge a emmagatzemar.

4.4.3. ComandesProvider

El proveïdor de dades de comandes consta de tres seccions importants: comandes, articles i Firebase. A diferència dels altres serveis, la gestió de la cistella implica major contingut de lògica "local" i no es limita a gestionar Firebase. Només la cistella es gestiona de forma síncrona, tota la resta de vistes reben les dades de forma asíncrona.

`afegirArticle(Producte,quantitat,Botiga)` es crida quan l'usuari toca el botó Afegir de la pàgina Producte-detall. En aquesta funció es comprova que la Comanda guardada a cistella existeix i no és d'una botiga diferent de la que s'ha visitat. En cas que no sigui així, es crea una Comanda buida. Posteriorment, s'afegeix l'article (creat a partir del producte rebut), si no existia ja a la comanda, o s'actualitza la quantitat, en cas contrari.

`obtenirQuantitatArticle(id)` i **`articleExisteix(id)`** retornen respectivament la quantitat indicada a cistella i si existeix l'Article corresponent a l'**`id`** rebut per paràmetre.

`creaArticle(Producte,quantitat)` crea un objecte Article a partir dels paràmetres rebuts.

`crearCistellaBuida(Botiga)` crea una nova Comanda amb les dades de la botiga rebuda i les d'usuari. Les dades d'usuari sempre són les mateixes però la botiga pot ser qualsevol. Per aquest motiu s'ha de passar la botiga per paràmetre.

afegirComandaFirebase(Comanda) insereix a Firestore l'objecte Comanda rebut. S'ha de tenir en compte que guarda el document de la comanda i la col·lecció d'articles. Per facilitar la gestió es genera un **id** abans de realitzar la inserció.

recuperarComandesBotigaFirebase() i **recuperarComandesUsuariFirebase()** obtenen la llista de comandes necessària en cada cas en un **Observable<Comanda[]>**. Les comandes d'Usuari es mostren a la pàgina Compres i es filtren per **idComprador** i les de Botiga es mostren a la pàgina Vendes i es filtren per **idProductor**. Tots dos **id**'s es comparen amb el UID de l'usuari.

recuperarArticlesComandaFirebase(Comanda) retorna la llista d'articles que formen part de la comanda rebuda per paràmetre en un **Observable<Article[]>**. Es recupera els articles en una funció a part perquè no es necessita les dades si no s'accedeix al detall d'una compra o una venda en concret.

actualitzarArticleComanda(Article,Comanda) inverteix l'indicador preparat de l'article a Firebase. Es crida de la pàgina Venda quan l'usuari toca un article i el marca.

actualitzarEstatComandaFirebase(Comanda,estat) canvia l'estat de la comanda de Firebase que es passa al nou estat rebut.

4.4.4. AuthProvider

El servei d'autenticació gestiona l'accés a Firebase Auth per registrar nous perfils i autenticar-se i a l'emmagatzematge local que guarda una còpia de l'usuari UserAuth autenticat si s'ha marcat la casella mantenir connectat. Ionic proporciona Storage [17], un paquet per gestionar l'emmagatzematge local documentat a la pròpia pàgina web.

El formulari de la vista (LoginPage i RegistrePage) s'enllacen a un objecte UserAuth guardat al servei.

Login() i **register()** envien les dades del formulari d'autenticació a Firebase, per comprovar les credencials o guardar-les respectivament, i retorna la resposta per ser gestionada a la mateixa pàgina.

clearUserAuth() i **validateFields()** actuen sobre el formulari per netejar els camps i per comprovar que s'ha introduït alguna cosa en ells respectivament.

getStoredUserAuth(), **setStoredUserAuth()** i **removeStoredUserAuth()** s'encarreguen de la gestió de l'usuari emmagatzemat a memòria del dispositiu.

4.4.5. UtilProvider

Aquest servei s'ha creat per guardar-hi algunes funcions que s'utilitzen en diferents llocs i que suposen vàries línies de codi com és el cas de llençar un *Toast*.

4.5. Firebase

Firebase és un *framework* que proporciona emmagatzematge remot al núvol mitjançant unes API dedicades a diferents funcionalitats. En el projecte de PopFood s'ha previst utilitzar els serveis d'autenticació, base de dades i Cloud Storage.

Com es pot deduir, el servei d'autenticació proporciona diferents mètodes d'autenticació, una base de dades dedicada i una llibreria de mètodes i propietats per registrar-se, autenticar-se, etc. El servei de base de dades ens proporciona emmagatzematge a una base de dades no relacional de tipus JSON (clau – valor). Recentment Firebase ha iniciat una nova versió del servei de base de dades anomenat Cloud Firestore i, tot i trobar-se en fase beta, s'ha optat per implementar l'emmagatzematge amb aquesta versió. El servei de Cloud Storage ens permet emmagatzemar imatges i/o vídeos amb mètodes molt similars a l'API de base de dades.

Firebase proporciona API's per diferents llenguatges. En el cas de PopFood s'hauria d'utilitzar l'API Web però està més dedicada a html i javascript. Per integrar les funcionalitats d'Autenticació i base de dades de Firebase s'ha utilitzat el paquet AngularFire2 [18] versió 5.0 que adapta les funcions de Firebase a Angular i permet importar-lo com a mòdul igual que la resta de llibreries.

Per fer-ho, primer hem de disposar de perfil d'usuari compatible, com pot ser Google, i s'ha de crear un projecte a Firebase. El procés de creació del projecte a firebase és molt senzill e intuïtiu. Només cal anar a la consola de Firebase, crear nou projecte, on es demana nom i país d'aquest. Per incorporar el projecte Firebase a la App Ionic que s'ha creat s'ha de definir la variable de connexió amb les dades del projecte Firebase i instal·lar el paquet AngularFire2.

Per instal·lar el paquet AngularFire2 s'utilitza *npm* amb la següent ordre per terminal.

```
npm install firebase angularfire2 --save
```

També és necessari registrar el paquet a **`app.module.ts`** i afegir-lo a Imports executant la funció **`initializeApp(FIREBASE_CONFIG)`**.

```
import { AngularFireModule } from 'angularfire2';
import { FIREBASE_CONFIG } from './firebaseConfig';
@NgModule({
   imports: [
       ...
       AngularFireModule.initializeApp(FIREBASE_CONFIG),
       ...
   ],
   ...
})
```

La variable FIREBASE_CONFIG es guarda en l'arxiu **`/app/firebaseConfig.ts`** amb les dades obtingudes de la consola de Firebase.

4.5.1. Autenticació Firebase

Per poder utilitzar les funcions d'autenticació del paquet AngularFire2 s'ha d'importar **`AngularFireAuthModule`** a la secció de Imports de **`app.module.ts`**, **`AngularFireAuth`** a **`AuthProvider`** i s'ha d'inicialitzar al constructor del controlador la instancia.

```
import { AngularFireAuth } from 'angularfire2/auth';
...
constructor(
    ...
    private angularFireAuth: AngularFireAuth,
  ) {
...
```

Un altre punt on s'ha d'actuar és en el servei d'autenticació de Firebase. S'ha d'indicar quin tipus d'autenticació es vol utilitzar.

Imatge 34. Configuració del mètoe d'inici de sessió accptat per Firebase Authentication.

Tocant a sobre d'un dels proveïdors s'obre una finestra que et permet activar-lo. Cada servei utilitza diferents funcions. En el cas de PopFood s'ha habilitat la autenticació per correu electrònic i contrasenya.

En el servei d'autenticació s'utilitzen dues funcions: **`signInWithEmailAndPassword`**, que comprova les credencials introduïdes i respon amb una promesa, i **`createUserWithEmailAndPassword`**, que insereix a la base de dades d'autenticació el correu electrònic i contrasenya indicats.

```
// Realitza la autenticació contra firebase auth i retorna la resposta.
login(){
  return this.angularFireAuth.auth.signInWithEmailAndPassword(this.userAuth.email,this.userAuth.pass);
}
// Registra un nou usuari a firebase auth i retorna la resposta.
register(){
  return this.angularFireAuth.auth.createUserWithEmailAndPassword(this.userAuth.email,this.userAuth.pass);
}
```

Imatge 35. Mètodes utilitzats per gestionar la autenticació amb Firebase.

Aquestes funcions únicament realitzen l'enviament de les dades al servei d'autenticació de firebase i retornen la resposta al sol·licitant. Això implica que s'ha de

gestionar aquesta resposta en la classe que crida les funcions **`login`** i **`register`** del servei AuthProvider.

Es crida la funció **`login`** a **`LoginPage`** quan un usuari introdueix les credencials i a **`app.component.ts`**, el controlador arrel, quan s'inicia l'aplicació i hi ha un **`UserAuth`** guardat a memòria local. La funció **`register`** solament es crida des de **`RegisterPage`** quan es vol donar d'alta un nou usuari. La gestió de la resposta es realitza en tots els casos de forma similar:

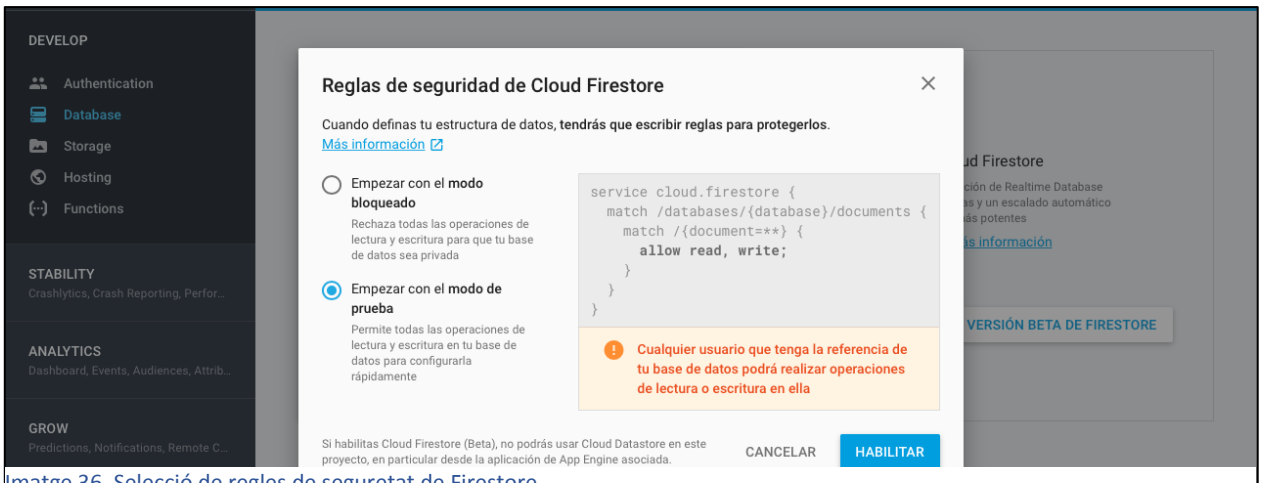
```
//Es crida la funció del Provider
this.auth.login().then( authData => {
        // resposta OK

        // Es guarda el UID rebut al UserAuth del servei.

        // Si s'ha marcat guarda conexió s'emmagatzema les dades.

        // S'obtenen les dades de Usuari i Botiga als seus serveis.

        // Passem a la pantalla d'inici de l'aplicació.

   }, error => {
        // ERROR. No s'ha pogut autenticar per algun motiu.

        // Es mostra a l'usuari el motiu de no haver autenticat.
   })
```

4.5.2.Base de dades Firestore

Per poder utilitzar les funcions d'emmagatzematge de dades del paquet AngularFire2 s'ha d'importar **`AngularFirestoreModule`** a la secció de Imports de **`app.module.ts`**, **`AngularFirestore`** als **`Providers`** que el necessiten (**`StorageProvider`**, **`BotigaProvider`** i **`ComandesProvider`**) i s'ha d'inicialitzar al constructor del controlador la instancia.

```
import { AngularFirestore } from 'angularfire2/firestore';
...
constructor(
    ...
    private angularFirestore: AngularFirestore,
  ) {
...
```

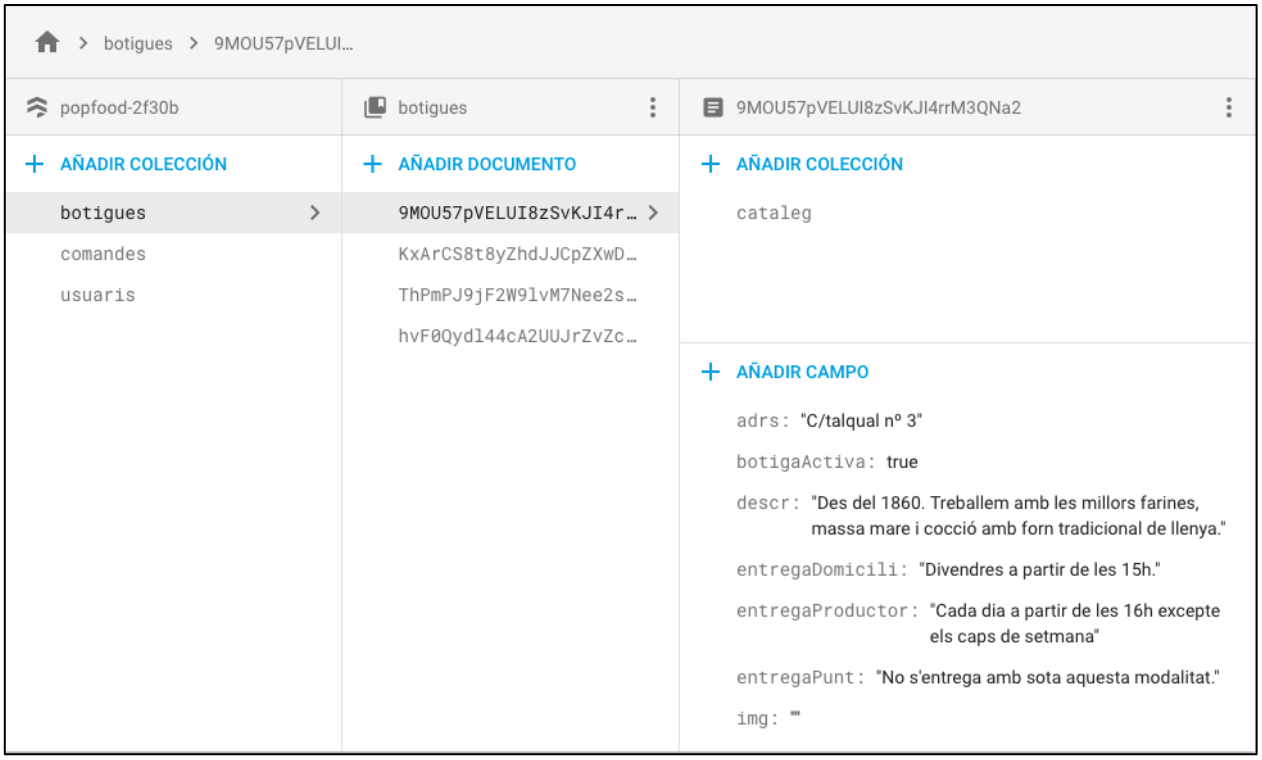
Quan s'accedeix a la secció de *Database* de la consola del projecte de Firebase es permet escollir la versió normal de base de dades o la versió beta. Si s'escull la versió



Imatge 36. Selecció de regles de seguretat de Firestore

beta *Cloud Storag*e et facilita la configuració de les regles d'escriptura. De moment s'ha marcat la versió de prova que permet que tothom pugui escriure i llegir la base de dades si es disposa de les dades de connexió.

Aquesta nova versió de base de dades consta de dos tipus d'elements: Col·leccions i Documents. Un document equivaldria a una parella clau-Objecte sempre i quan l'objecte sigui de tipus de dades bàsics i una col·lecció conté documents.



Imatge 37. Visualització de la base de dades a la pàgina de Firebase.

Les dades emmagatzemades per PopFood es dividiran en tres col·leccions: usuaris, botigues i comandes. Com es pot veure a la Imatge 37, un document pot contenir a la vegada una altra col·lecció cosa que ens permet incloure el catàleg a cada botiga. Les col·leccions són únicament contenidors, no tenen propietats. Per tant, una col·lecció ha de contenir almenys un document.

El **`id`** del document pot generar-se automàticament o pot introduir-se'n un de propi. Per PopFood s'ha utilitzat com a identificador per les col·leccions Usuaris i Botigues el UID obtingut del registre en el servei d'autenticació.

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
test4@test.com	✉	30 nov. 2017	9 dic. 2017	9MOU57pVELUI8zSvKJI4rrM3QNa2
test3@test.com	✉	30 nov. 2017	9 dic. 2017	KxArCS8t8yZhdJJCpZXwDT2f32l1

Imatge 38. Exemples de contes d'usuari de la base de dades d'autenticació.

Escriptura de dades

En el moment que s'introdueix un document s'ha de indicar obligatòriament la col·lecció on s'emmagatzema però, depenent de si l'identificador és automàtic o no, s'utilitza la funció **`collection(...).add( data )`** o **`collection(...).doc(id).set( data )`**. La funció **`set( ... )`**, com és habitual, insereix les dades si no existien o, si existien, les substitueix.

L'exemple més clar es troba a la funció **`afegirComandaFirebase`** del servei **`ComandesProvider`** a la següent captura.

```
// S'obté l'id abans per tal de poder afegir després els articles
let id = this.angularFirestore.createId();

//Es crea la comanda a firebase sense articles perque sino ho afegeix com un array.
this.angularFirestore
  .collection('comandes')
  .doc<Comanda>(id)
  .set(Object.assign({},comanda));

// S'afegeixen els articles a la comanda com a collection.
articles.forEach(article => {
  this.angularFirestore
  .collection('comandes')
  .doc(id)
  .collection('articles')
  .add(Object.assign({},article));
})
```

Imatge 39. Exemple d'ús de les funcions *add* i *set* per escriure dades a la base de dades

Interessa definir l'identificador de la Comanda que s'introdueix a la col·lecció comandes ja que després s'ha de guardar els articles en una col·lecció dins de la comanda. Per tant, s'utilitza **`.set(data).`**

En canvi, en el moment de guardar els articles no fa falta saber l'identificador. El crearà automàticament i quan es faci la lectura d'aquests ja es disposarà d'ell si fa falta.

Lectura de dades

Hi ha dos mètodes de lectura de dades: **`ValueChanges( )`** i **`SnapshotChanges( )`**. La diferència bàsica que ens indica quan s'ha d'utilitzar l'un o l'altre és si es necessita meta-data, com el **`id`**, o només les dades. Si només es vol recuperar dades la gestió de la resposta és més senzilla ja que no s'ha de filtrar la meta-data.

La funció **recuperarUsuariFirebase** del servei **StorageProvider** serveix com a exemple de l´ús de **valueChanges**. Només interessa obtenir les dades de l'usuari, el identificador ja el sabem.

```
// Recupera les dades del usuari de Firebase i ho guarda com a observable.
recuperarUsuariFirebase(): Observable<Usuari>{
  return this.angularFirestore
    .collection('usuaris')
    .doc<Usuari>(this.auth.userAuth.uid)
    .valueChanges();
}
```

Imatge 40. Exemple d'utilització de la funció valueChanges per rebre dades de Firestore

Per altra banda, la funció **recuperarCatalegFirebase** de **BotigaProvider** mostra com la lògica per poder retornar un Observable és més complicada. El resultat de **snapshotChanges()** es canalitza a **.map(...)** que ens filtrarà una primera capa de dades per obtenir una llista de productes. Aquesta llista s'ha de canalitzar altre cop amb **.map(...)** per tal d'arribar a les dades buscades.

```
// Retorna la llista de productes guardades al cataleg d'una botiga en concret.
recuperarCatalegFirebase(uid: string): Observable<Producte[]>{
  return this.angularFirestore
    .collection('botigues')
    .doc<Botiga>(uid)
    .collection('cataleg')
    .snapshotChanges()
    .map( prods=> {
      return prods.map( prod => {
          const data = prod.payload.doc.data() as Producte;
          const id = prod.payload.doc.id;
          return { id, ...data };
        })
    });
}
```

Imatge 41. Exemple d'utilització de la funció *snapshotChanges* per rebre dades de Firestore

Una altra particularitat és que es pot simular consultes SQL a la crida a la base de dades filtrant en origen les dades que es descarregaran. És el que s'aplica en les crides realitzades a comandes, ja que depenent de si estem buscant les de part de botiga o d'usuari es filtraran per un camp diferent.

```
// Retorna la llista de comandes amb el id de productor de l'usuari.
recuperarComandesBotigaFirebase(): Observable<Comanda[]>{
  return this.angularFirestore
    .collection<Comanda>('comandes', ref => {
      return ref.where('idProductor', '==', this.auth.userAuth.uid)
    })
    .snapshotChanges().map(comandes => {
      return comandes.map( comanda => {
        const data = comanda.payload.doc.data() as Comanda;
        const id = comanda.payload.doc.id;
        return { id, ...data };
      })
    });
}
```

Imatge 42. Exemple d'ús de la funció *where* per filtrar les dades de Firestore.

Com es veu en l'exemple, amb la funció **`where()`** apliquem una condició a la cerca de dades. En aquest cas, com que es vol obtenir totes les comandes fetes a la botiga de l'usuari, es filtra les dades per comandes on el **`id`** del productor (Botiga) sigui el UID de l'usuari, que correspon al **`id`** de la seva Botiga. El tipus de clàusules que es poden crear son limitades però permet filtrar les dades en origen minimitzant el consum innecessari de dades mòbils de l'usuari.

4.5.3.Modificació de dades

Per actualitzar les dades de Firebase es pot utilitzar la funció **`set(objecte)`** per substituir el document complet o, si es desitja modificar solament alguns camps, la funció **`update({camp: nouValor})`**. Per exemple, en el proveïdor de comandes, la funció **`actualitzarEstatComandaFirebase(...)`** modifica únicament el valor del camp estat de la comanda.

```
actualitzarEstatComandaFirebase(comanda: Comanda, estat: number){
  return this.angularFirestore
    .collection('comandes')
    .doc(comanda.id)
    .update({estat: estat});
}
```

Imatge 43. Exemple d'utilització de la funció *update* de Firestore.

L'eliminació d'un document es realitza amb la funció **`.delete()`** sobre aquest. En la funció **`eliminarProducteDeCatalegFirebase(...)`** es pot veure com s'elimina de la base de dades el document amb l'identificador del producte rebut per paràmetre.

```
eliminarProducteDeCatalegFirebase(producte: Producte){
  console.log(producte.id);
  this.angularFirestore
    .collection('botigues')
    .doc<Botiga>(this.auth.userAuth.uid)
    .collection('cataleg')
    .doc<Producte>(producte.id)
    .delete();
}
```

Imatge 44. Exemple d'utilització de la funció *delete* de Firestore

4.5.4.Storage Firebase

Per operar amb Firebase Storage i poder emmagatzemar les imatges al núvol no es podia utilitzar les funcions proporcionades pel paquet AngularFire2 perquè l'API de Firebase Storage no s'hi ha implementat. En aquest cas s'ha hagut de recórrer a l'API Web proporcionada per Firebase i s'ha d'importar en cada servei que haurà de gestionar alguna imatge la llibreria corresponent.

```
import { storage } from 'firebase';
```

No ha estat necessari afegir cap paquet a **`app.module.ts`** ja que el projecte Firebase ja s'ha integrat a PopFood amb la funció **`.initializeApp(FIREBASE_CONFIG)`**.

S'utilitza en les tres pàgines de gestió del perfil de l'usuari: **`Usuari-Editar`**, **`Bogita-Editar`** i **`Producte-Editar`**. Les imatges capturades per la càmera del dispositiu (punt 4.6) s'envien a Firebase un cop s'accepten els canvis.

```
guardarImatgeFirebase(refString: string, imatge: any){
  return storage().ref(refString).putString(imatge, 'data_url');
}
```

Imatge 45. Emmagatzematge d'imatges a Firebase Storage

Aquesta funció retorna un objecte **`UploadTask`** que de tipus asíncron que serà gestionat al controlador origen. En la implementació realitzada solament es comprova quan s'ha finalitzat l'operació de pujada per capturar l'URL de la imatge guardada i assignar-li al camp **`img`** de l'objecte editat.

```
const uploadTask = this.botigaStorage.guardarImatgeFirebase(this.calcularRef(), this.botiga.img);
uploadTask.on(storage.TaskEvent.STATE_CHANGED, ()=> {
  if (uploadTask.snapshot.downloadURL){
    this.botiga.img = uploadTask.snapshot.downloadURL;
    this.guardarCanvis('Modificacions guardades');
  }
}, (error) => {
  this.util.presentToast('No s\'ha pogut guardar les modificacions');
});
```

Imatge 46. Gestió de la resposta a la pujada d'imatges a Firebase.

Les imatges emmagatzemades a Firebase s'han distribuït en carpetes, en un primer nivell per usuaris identificant-les amb l'UID i, dins d'aquestes, organitzades per botiga, usuari i producte. S'ha guardat una carpeta a part anomenada **`/default`** amb les imatges per defecte que s'assignen als diferents elements en crear-se.

Per tal d'evitar mantenir imatges antigues guardades cada cop que es canvia una imatge es sobreescriu la referència existent i tal com s'ha implementat la gestió de les imatges en l'estructura de dades hi ha un inconvenient. Tot i mantenir la referència, cada cop que actualitzem una imatge l'URL d'aquesta canvia. Això implica que les URL's guardades en comandes deixen de ser vàlides perquè no s'actualitzen. Una solució a aquest problema seria emmagatzemar la referència enlloc de l'URL i obtenir-la cada cop que es requereix. No s'ha disposat de temps suficient per realitzar el canvi i s'indicarà com una de les millores pendents a treballs futurs.

4.6. Captura d'imatges amb càmera.

Una de les virtuts de Ionic és que incorpora Cordova [13], el qual ens permet utilitzar funcions natives dels dispositius. En el marc d'aquest treball només es preveu generar l'instal·lable .APK per dispositius Android.

Per poder utilitzar-ho s'ha d'instal·lar els plugins natius.

```
$ ionic cordova plugin add cordova-plugin-camera
$ npm install --save @ionic-native/camera
```

La càmera passa a ser com un proveïdor de dades més i s'ha d'afegir a l'arxiu **`app.module.ts`** a l'apartat de *Providers*.

```
import { Camera } from '@ionic-native/camera';
...
@NgModule({
  ...
  providers: [
    ...
    Camera
  ]
  ...
})
export class AppModule { }
```

La crida a la funció nativa de la càmera s'utilitza en els serveis **`Storage`**, per la gestió de l'usuari, i **`Botiga`**. Inicialment s'havia previst crear un servei dedicat a la funció nativa però finalment s'opta per crear la funció de captura al servei corresponent per unificar la gestió de les estructures corresponents. A la funció **`capturarFoto()`** dels serveis es configura les opcions de càmera i es crida la funció **`getPicture(options)`** de la llibreria nativa de *Cordova*.

```
// Crida a la funció de camera i guarda a Firebase Storage la foto.
async capturarFoto(): Promise<any>{
  try {
    //Definició de les opcions de Camera
    const options: CameraOptions = {
      quality: 100,
      targetHeight: 600,
      targetWidth: 600,
      allowEdit: true,
      destinationType: this.camera.DestinationType.DATA_URL,
      encodingType: this.camera.EncodingType.JPEG,
      mediaType: this.camera.MediaType.PICTURE
    };

    //Crida a la funció de camera nativa.
    const result = await this.camera.getPicture(options);


    //Rebem la imatge de la camera.
    return `${result}`;

  } catch (e) {
    console.error(e);
  }

}
```

Imatge 47. Configuració de les opcions de càmera.

Un cop es rep el resultat de la funció **`getPicture(...)`** es retorna a la pàgina la imatge en el format **`jpg`** i aquesta serà assignada al camp **`img`** de l'objecte que s'està editant i es mostra a la interfície. Tal com s'indica a l'apartat Storage Firebase, aquesta imatge es puja a Firebase en el moment que l'usuari accepta els canvis. Si els canvis es descarten, la instància de l'objecte (**`Botiga`**, **`Usuari`** o **`Producte`**) del servei es recarrega de Firebase recuperant la imatge correcta.

Les opcions de càmera definides són les següents:

- **`quality`**: Indica la qualitat de la imatge. Es defineix 100 que és el màxim.

- **`targetHeight`** i **`targetWidth`**: Mida de la imatge. Es defineix 600x600 per obtenir una imatge quadrada.
- **`allowEdit`**: S'indica a *true* per activar l'opció d'enquadrar la imatge després de capturar-la. S'activa aquesta opció perquè la imatge obtinguda agafi les mides indicades.
- **`destinationType`**, **`encodingType`** i **`mediaType`**: Indiquen el format de retorn de la imatge, que es defineix com a DATA_URL, tipus de codificació, que es defineix JPEG, i el tipus de mitjà s'ha obtingut, que es defineix com a imatge.

5. Conclusions

La realització de PopFood ha estat principalment un repte per mi. La meva experiència en la disciplina de *front-end,* amb HTML, CSS i Javascript, és escassa i la realització d'aquest treball era una oportunitat per aprofundir en aquests llenguatges. És cert que treballar amb *frameworks* no és exactament treballar amb ells però l'ús de llibreries també està molt estes avui dia i sense elles no podria acabar l'aplicació a temps.

Per una banda, el resultat final em deixa satisfet, ja que s'ha dissenyat una aplicació e implementat gran part d'ella treballant amb connexions asíncrones i utilitzant funcions natives del dispositiu. Però per l'altra, la falta de temps m'ha limitat el poder polir detalls tant en documentació com en l'aplicació. Crec que no vaig plantejar-me correctament el tema i abast del treball a l'inici del semestre i, tot i dedicar-hi moltes hores, no he estat capaç d'assolir la qualitat de treball que m'hagués agradat.

La planificació que es realitza inicialment es fa en base al que s'haurà d'entregar cada PAC i, tot i que els temps invertits no han estat del tot els planificats, s'ha anat seguint tots els punts satisfactòriament. Evidentment, la falta d'experiència dificulta marcar uns terminis encertats o definir un ordre equivocat de treball.

Un exemple molt clar és com s'ha planificat la implementació de l'aplicació. El plantejament era desenvolupar l'aplicació sencera afegint posteriorment els punts complicats com eren l'emmagatzematge de dades i autenticació amb Firebase i la captura d'imatges amb funcions natives del mòbil. Això ha comportat implementar parts de codi que després s'havien de tornar a escriure, ja que treballar amb dades asíncrones implica canviar l'estructura i afegeix dificultats extra a l'hora de combinar tots dos conceptes.

En l'aplicació entregada s'implementa la part de l'aplicació corresponent a la gestió de la botiga personal. Tot i això han quedat pendents alguns detalls i d'altres que s'haurien de millorar:

- **Afegir el tipus de botiga** per tal que els usuaris puguin realitzar el filtrat per tipus i, el dia que s'implementi, indicar la icona corresponent al mapa.
- **Introduir la geo-localització de la botiga**. Es tractaria de substituir l'adreça que es mostra actual per la geo-localització poden mostrar-ho en latitud i longitud o en adreça habitual si existeix.
- **Eines de cerca en les llistes de compres i vendes**. Actualment es mostren dues icones que servirien per ordenar la llista pel camp indicat o cercar directament el nom però no s'ha implementat aquestes funcions.
- **Ampliar el nombre d'imatges per les botigues i els productes**. Actualment sol hi ha la possibilitat d'incloure una única imatge.
- **Millorar la gestió d'imatges de Firebase**. La solució implementada no s'ha pogut optimitzar com seria convenient i ens trobem amb errors com que si canviem la imatge d'un producte, les comandes realitzades amb aquest, contenen una URL invàlida, ja que, tot i mantenir la referència d'emmagatzematge a Firebase, la URL varia.

- **Modificar el component que mostra els elements del catàleg a la vista de Productor** per tal d'incorporar en el mateix element la possibilitat d'afegir el producte al cistell sense necessitat d'entrar al detall.

També faltaria implementar la banda del client com es detalla en la següent llista:

- **Implementar el mapa** centrat en la ubicació de l'usuari mostrant els productors propers.
- **Implementar la llista de productors**. Actualment hi ha implementada una llista on sol es mostren els noms de les botigues però és una solució provisional per testejar les comandes. La solució final hauria de ser la detallada en el prototip.

Altres exemples de treballs futurs interessants per millorar l'experiència d'usuari serien els següents;

- **Inclusió d'un sistema de pagament**. Actualment l'aplicació no contempla la part del pagament de comandes. Aquest punt implicaria revisar la legislació del país per tal de proporcionar tots els serveis obligatoris com la possibilitat d'extreure una llista de transaccions o l'aplicació d'impostos.
- **Preferències de l'usuari**. Disposar d'una llista de preferits o filtrar els tipus de botigues permanentment com a exemples.
- **Inclusió de missatgeria interna de l'aplicació**.

6. Glossari

Productor	Usuari corresponent al perfil d'usuari amb el mateix nom que disposa de funcionalitats afegides a la versió bàsica.
Consumidor	Usuari corresponent al perfil d'usuari amb el mateix nom. Disposa de les funcionalitats bàsiques
Punt de recollida	Lloc on el productor i el consumidor acorden realitzar l'entrega dels productes.
PopFood	Nom de l'aplicació que es desenvolupa en aquest treball.
Ionic CLI	*Command Line Interface*. Llistat de comandes de treball de Ionic sobre consola que permet accions com generar vistes, iniciar el servidor o generar un instal·lable per Android .apk.
Patró singleton	Es tracta d'un mètode de codificació d'una classe de forma que evites que es creï múltiples instancies d'aquest i sol n'existeix un.
JSON	Es una estructura de dades que es basa en el format de clau – valor amb la possibilitat que es pugui fer de forma recurrent i que una clau congingui al mateix temps vaires parelles clau – valor.

7. Bibliografía

[1] Nielsen, «Seis de cada diez españoles son total consumer: compaginan hacer la compra en tienda con online,» 2017. [En línea]. Available: http://www.nielsen.com/es/es/press-room/2017/nielsen-360-micro.html. [Último acceso: 4 10 2017].

[2] G. d. España., «Informe del consumo de alimentación en España 2016,» Ministerio de Agricultura y Pesca, Alimentación y Medio Ambiente, Madrid.

[3] Wallapop, [En línea]. Available: https://es.wallapop.com/.

[4] L'hort d'en Dídac, [En línea]. Available: https://play.google.com/store/apps/details?id=com.initias.tdapp.

[5] Agrofresc, [En línea]. Available: https://play.google.com/store/apps/details?id=com.farandsoft.agrofresc .

[6] The food assembly, [En línea]. Available: https://lacolmenaquedicesi.es/ca/p/index .

[7] Cerroviejo, [En línea]. Available: https://play.google.com/store/apps/details?id=org.cerroviejo.android&rdid=org.cerroviejo.android .

[8] Carrefour, [En línea]. Available: https://play.google.com/store/apps/details?id=com.mo2o.carrefour.alimentacion.

[9] «Node.js,» Node.js, [En línea]. Available: https://nodejs.org/es/. [Último acceso: 12 12 2017].

[10] npm. [En línea]. Available: https://www.npmjs.com/. [Último acceso: 2017].

[11] Ionic, «Build amazing apps in one codebase, for any platform, with the web.,» [En línea]. Available: https://ionicframework.com/. [Último acceso: 12 12 2017].

[12] Angular. [En línea]. Available: https://angular.io/. [Último acceso: 12 12 2017].

[13] Apache, «Apache Cordova,» [En línea]. Available: https://cordova.apache.org/. [Último acceso: 12 12 2017].

[14] Google, «Firebase,» [En línea]. Available: https://firebase.google.com. [Último acceso: 12 12 2017].

[15] I. CLI. [En línea]. Available: https://ionicframework.com/docs/cli/. [Último acceso: 12 12 2017].

[16] Ionic, «Component Menú de Ionic,» Ionic, 8 Desembre 2017. [En línea]. Available: https://ionicframework.com/docs/components/#menus.

[17] «Ionic Storage,» [En línea]. Available: https://ionicframework.com/docs/storage/. [Último acceso: 12 12 2017].

[18] «AngularFire2 The official library for Firebase and Angular,» [En línea]. Available: https://github.com/angular/angularfire2. [Último acceso: 12 12 2017].

[19] Ministerio de Agricultura y Pesca, Alimentación y Medio Ambiente, «Informe del consumo de alimentos en España 2016,» Taller del Centro de Publicaciones del MAPAMA, 2017.

8. Annex

8.1. Annex I

Codi guardat al repositori GitHub:

https://github.com/miqasals/popFood.git

Base de dades Firebase:

https://console.firebase.google.com/u/0/project/popfood-2f30b/overview

Per accedir al projecte Firebase es requereix donar d'alta l'usuari amb un correu elctrònic de Google.