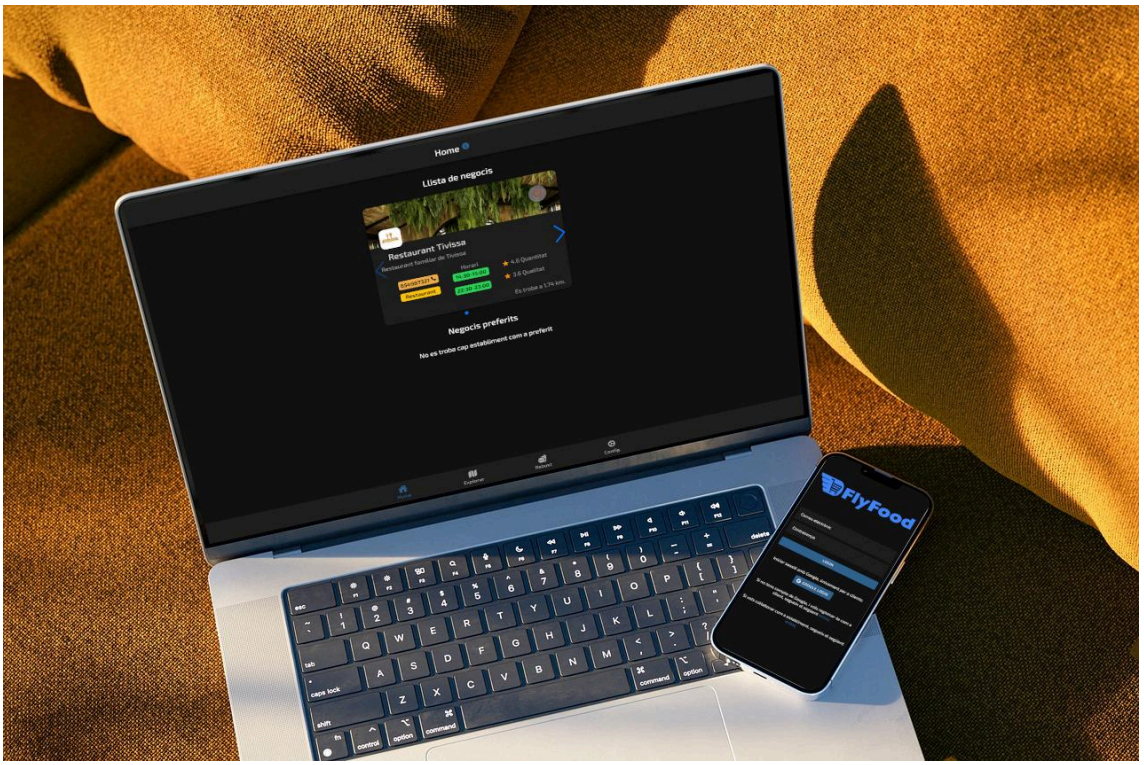


Desenvolupament d'una aplicació multiplataforma per a reduir el malbaratament d'aliments

Martí Masot Serrano
Enginyeria informàtica
Desenvolupament multiplataforma d'aplicacions mòbils

Xavier Velàsquez Melenciano, Jordi Almirall López, Carles Garrigues Olivella

06/2024

Aquesta obra està subjecta a una llicència de [Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons]

FITXA DEL TREBALL FINAL

Títol del treball:	*Desenvolupament d'una aplicació multiplataforma per a reduir el malbaratament d'aliments*
Nom de l'autor:	*Martí Masot Serrano*
Nom del consultor/a:	*Xavier Velàsquez Melenciano,Jordi Almirall López*
Nom del PRA:	*Carles Garrigues Olivella*
Data de lliurament (mm/aaaa):	*06/2024*
Titulació o programa:	*Enginyeria Informàtica*
Àrea del Treball Final:	*Desenvolupament multiplataforma d'aplicacions mòbils*
Idioma del treball:	*Català*
Paraules clau	*"DCU", "MVC", "Ionic"*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Actualment, el malbaratament d'aliments és un problema molt present en la nostra societat. De fet, ja existeixen aplicacions com Too Good To Go que ja presenten solucions per a reduir el malbaratament d'aliments en els establiments. Tot i això, el seu ús no està estès encara en tot el territori català i no ofereix solucions perquè l'usuari pugui portar un registre de tots els aliments del seu rebost i la data de caducitat.

Per aquest motiu es decideix portar a terme aquest projecte on primerament se n'especifica l'abast, els objectius i la planificació a seguir. Posteriorment, es duen a terme tècniques de DCU que van permetre implementar un prototip d'alta fidelitat amb el qual es van fer tests amb usuaris reals i corregir certs aspectes abans d'implementar-lo. Com que es desenvolupava una aplicació multiplataforma, també va caldre dissenyar l'arquitectura general del sistema i la base de dades.

Finalment, les conclusions del treball reflecteixen l'eficàcia del procés seguit durant tot el projecte i la implementació, pel fet que s'aconsegueixen dues aplicacions finals. Una web i l'altra Android, que permeten oferir i comprar aliments a punt de caducar, explorar ofertes a través d'un mapa interactiu i gestionar els aliments del rebost de l'usuari, permetent afegir els aliments a partir de la fotografia d'un tiquet i ser notificats quan algun aliment està a punt de caducar, entre d'altres.

Abstract (in English, 250 words or less):

Food waste is a very present problem in our society. In fact, there are already applications like Too Good To Go that offer solutions to reduce food waste in establishments. However, its use is not yet widespread throughout Catalonia, and it does not provide solutions for users to keep track of all the food in their pantry and the expiration dates.

For this reason, the decision was made to carry out this project, which first specifies the scope, objectives, and planning to follow. Subsequently, UID (User Interface Design) techniques are implemented to develop a high-fidelity prototype, with which tests are conducted with real users to correct certain critical aspects before implementation. Since a cross-platform application is being developed, it was also necessary to design the general architecture of the entire system and the database design.

Finally, the conclusions of the work reflect the effectiveness of the process followed throughout the project and its implementation, as two final applications were achieved. One web application and the other Android, which allow offering and buying food about to expire, exploring offers through an interactive map, and managing the user's pantry. This includes adding food by photographing a receipt and being notified when any food is about to expire, among other features.

Índex

Llista de figures

1. Introducció

1.1. Context i justificació del Treball

En un planeta on l'any 2021 hi havia 828 milions de persones amb problemes per a aconseguir aliment [1 *WPF. 2022*], hi ha certes regions del planeta on es malbaraten una gran quantitat d'aliments per diferents motius, per sobre estoc o sobreproducció, per falta de compliment en estàndards de qualitat, a causa de compres excessives per part de consumidors... Per a posar un exemple, únicament a Espanya l'any 2022 es calcula que es van malbaratar 1364 milions de kg d'aliments. Dels quals s'estima que un 20% dels aliments es trobaven en bon estat i aptes per al consum en el moment de malbaratar-los. [2 *Bolinches, Cristina G. 2022*]

Tot i això, Espanya no és l'únic país que malgasta aliments. Segons la notícia [3 *EPE. 2022*] es menciona que a la UE es van malbaratar 153,5 milions de tones d'aliments aquest l'any 2022. A més a més, en l'article es posa èmfasi en el fet que la UE va malbaratar més aliments que aliments importats d'altres països. Això podria fer arribar a la conclusió que realment no seria necessari importar aliments d'altres regions del planeta si fóssim capaços d'aprofitar tot aquest rebuig alimentari. A més a més, si no calgués importar aliments d'altres regions del planeta, es contribuiria a la reducció de la crema de combustibles fòssils i de gasos d'efecte hivernacle. D'altra banda, tots aquests aliments que no s'importarien d'altres regions podrien destinar-se a alimentar altres poblacions que es troben en situacions de fam o properes a la inanició.

Actualment, existeixen aplicacions com Too Good To Go, que han estat dissenyades per a reduir el malbaratament d'aliments en supermercats i tendes. El punt fort d'aquesta aplicació és sens dubte el de poder comprar un article a preu reduït just abans que la tenda el llença. Tot i això, aquestes aplicacions no tenen en compte el malbaratament d'aliments que es produeixen en les llars un cop ja s'ha fet la compra.

Amb aquest projecte es busca crear un aplicatiu multiplataforma que pugui ser accessible des de la majoria dels dispositius disponibles al mercat. Ja siguin mòbils (únicament telèfons i tauletes, no s'inclouen ni rellotges ni televisors) com d'escriptori (mitjançant un navegador), desenvolupant una única aplicació (amb un codi font) compatible amb totes aquestes plataformes. **Amb aquesta app es vol obtenir una eina que permeti als negocis oferir productes amb una pròxima data de caducitat a un preu reduït.** D'aquesta manera, els usuaris puguin aprofitar-los a la vegada que es beneficien del descompte. Finalment, tots els aliments que han estat oferts mitjançant la plataforma seran donats a una ONG o menjador social proper a l'establiment.

Per una altra banda, per a facilitar la reducció del malbaratament d'aliments a les llars, s'implementarà una funcionalitat on cada usuari podrà portar un

registre dels aliments adquirits en les compres habituals i l'aplicació mostrarà avisos basats en una data de caducitat estimada, segons el moment d'adquisició i aliment, o una data introduïda per l'usuari. A més a més, per a facilitar la introducció d'aquestes dades s'implementarà una funcionalitat que a partir d'una fotografia del tiquet de compra de qualsevol supermercat o negoci s'extregui un llistat dels aliments comprats i l'usuari podrà introduir una data (o utilitzar una data estimada segons l'aliment) en la qual es notificarà a l'usuari si l'usuari no ha marcat com a consumit l'article. Si després d'aquesta data no s'ha marcat l'aliment com a consumit es marcarà aquest aliment com a desaprofitat. L'aplicació també mostrarà estadístiques de progrés de l'usuari. El fet que un aliment de l'inventari d'un usuari sigui marcat com a desaprofitat repercutirà negativament en les estadístiques, aprofitar aliments d'un negoci mitjançant la plataforma, repercutirà positivament en les estadístiques de l'usuari, així com el fet de mantenir l'inventari al dia sense que els aliments introduïts arribin a ser marcats com a desaprofitats. A més a més, la recopilació d'aquestes estadístiques de manera anònima permetrà donar una visió global de l'ús de l'app i del malbaratament d'aliments. També es permetrà que un usuari pugui compartir un inventari amb un o més usuaris i fomentar la cooperació en la gestió de l'inventari, per exemple en un pis compartit.

1.2. Objectius del Treball

L'objectiu principal del treball és proveir als negocis un aplicatiu mòbil que els permeti oferir productes amb una data de caducitat propera a un preu reduït. A més a més, es proveirà als usuaris que utilitzin l'app una funcionalitat per a gestionar l'inventari de la seva llar per a reduir el malbaratament d'aliments a la llar.

Com a objectius específics funcionals l'app ha de:

- Oferir aliments amb una data pròxima de caducitat: Els negocis podran oferir aliments als usuaris.
- Compra d'aliments amb una data pròxima de caducitat a preu reduït: Els usuaris podran comprar els aliments oferits per part dels negocis, mitjançant targeta a través de l'app.
- Descobrir i explorar negocis que ofereixen aliments amb una data pròxima de caducitat, per proximitat i també a través d'un mapa.
- Valorar i opinar sobre el tractament rebut per part d'un establiment registrat en la plataforma en el qual s'ha fet una compra.
- Poder marcar com a favorits els establiments preferits de l'usuari.
- Rebre notificacions de noves ofertes publicades per establiments propers i els establiments favorits.
- Dur un registre dels aliments que es troben al rebost dels usuaris o al magatzem dels negocis: Tant els usuaris com els negocis podran emmagatzemar un registre d'aliments que han comprat en qualsevol establiment o proveïdor i definir una data límit de consum.
- Ser notificats dels aliments a punt de caducar: Els usuaris rebran notificacions i avisos basats en els aliments no consumits que es troben a punt de complir la data límit de consum.
- Millorar l'agilitat en la introducció de les dades: Els usuaris podran fer una fotografia al tiquet de compra per a identificar de manera més àgil els aliments que han comprat.

Tant l'app com el servidor hauran de complir també els següents requisits no funcionals

- Garantir la seguretat i privacitat de les dades confidencials dels usuaris i també garantir l'anonimat dels usuaris en la recopilació de les dades d'ús de l'aplicació.
- Ser accessible des de qualsevol dispositiu mòbil o ordinador, això comprèn optimitzar l'aplicació per a diverses plataformes i resolucions de pantalla, assegurant-se que sigui accessible des de telèfons mòbils, tauletes i ordinadors. A més a més, haurà de complir amb els estàndards mínims d'accessibilitat, així com oferir una navegació i un ús intuïtius.
- Cal assegurar la disponibilitat del servei, utilitzant servidors redundants i pràctiques d'alta disponibilitat per minimitzar el temps d'inactivitat de l'aplicació. Així com gestionar eficientment l'augment de la demanda.

1.3. Enfocament i mètode seguit

L'estratègia per a dur a terme aquest treball serà la d'adaptar un producte existent com poden ser les aplicacions "Too Good To Go" o "Encantado de Comerte", ja que aquestes aplicacions han estat desenvolupades per a reduir el malbaratament d'aliments en supermercats i altres negocis on es venen aliments. Per tant, el que es farà serà brindar les mateixes funcionalitats que brinden aquestes aplicacions en qüestió d'oferir i poder comprar aliments amb data pròxima de caducitat de negocis de proximitat. I a més s'implementaran les funcionalitats necessàries perquè els usuaris puguin dur un registre dels aliments del rebost.

Per a fer-ho s'implementarà una aplicació multiplataforma mitjançant unmarc de treball (*framework*) com és Ionic (utilitzant JavaScript com a llenguatge de programació). Aquesta *framework* permetrà desenvolupar una única aplicació desenvolupant únicament un codi que servirà per a executar l'aplicació en diferents dispositius gràcies al complement (*plugin*) Capacitor. A més, Ionic juntament amb Vue.js i altres llibreries que contenen components UI predefinits, permetran desenvolupar un disseny responsiu on les vistes que creï de l'aplicació seran compatibles en qualsevol dispositiu sense tenir la necessitat de desenvolupar una vista per a cada dispositiu. També gràcies al fet que Ionic és un *framework* de Node.js, permet afegir altres llibreries que faciliten la integració d'elements com mapes interactius, icones, etc.; de manera ràpida i còmoda, i també permetrà integrar llibreries de gestió d'estats (com Pinia.js) o de gestió de peticions HTTP (com Axios).

A més a més, el servidor API que respondrà a les peticions de cada client estarà desenvolupat també amb el mateix llenguatge de programació que el front-end, eliminant la necessitat d'aprendre un altre llenguatge per al desenvolupament. Igual que en el front-end el fet d'utilitzar Node.js, permetrà utilitzar llibreries per a implementar funcionalitats que sense aquestes llibreries desenvolupar aquestes funcionalitats tindrien un cost temporal de desenvolupament més alt. Algunes d'aquestes llibreries seran Express.js per al desenvolupament de l'API i els *endpoints*, Helmet.js per a protegir l'API d'atacs com XSS, injecció de codi... Per una altra banda, el fet d'usar contenidors Docker i Portainer per al desplegament en producció de l'API permetrà desenvolupar un back-end escalable i tolerant a fallades, ja que Portainer permet la implementació d'equilibradors de càrrega i Docker permet la replicació de contenidors en cas de fallada.

L'ús de MongoDB com a gestor de bases de dades ajudarà a poder definir l'estructura de la base de dades amb el mateix llenguatge de programació utilitzat a l'API, d'aquesta manera s'agilitza el procés de creació de la base de dades. A més a més, MongoDB està preparat per a ser emprat en clústers, fet que també facilitarà escalar horitzontalment la base de dades de manera senzilla.

El mètode de desenvolupament de l'aplicació serà en cascada (*waterfall*), és a dir, primer caldrà centrar-se en l'anàlisi i planificació del projecte, perquè posteriorment es tingui una pauta clara de les funcionalitats que haurà de complir l'app. Un cop s'hagi planificat el projecte es procedirà a fer el disseny de la interfície gràfica basat en els usuaris i es durà a terme el disseny de l'arquitectura del sistema i el disseny de la lògica de negoci de l'aplicació. Finalment, quan estigui tot el sistema dissenyat, per a dur a terme la implementació del sistema únicament s'hauran de seguir les pautes de disseny i aprofitar al màxim les funcionalitats que brinden les llibreries de tercers.

Gràcies a la utilització d'aquesta estratègia permetrà desenvolupar i implementar les funcionalitats ja existents en l'aplicació Too Good To Go de manera ràpida per a poder dedicar més temps en dissenyar i implementar les noves funcionalitats necessàries per a complir amb els objectius fixats. Com ja s'ha comentat anteriorment, aquesta estratègia també permetrà complir amb els objectius no funcionals de manera ràpida i senzilla sense haver d'implementar des de 0 aquestes funcionalitats gràcies a la utilització de programari i llibreries de tercers ideades específicament per a complir amb aquests objectius (seguretat, privacitat, accessibilitat, disponibilitat...). A més, el fet que aquestes llibreries hagin estat ideades específicament per aquests objectius, minimitzen el risc que l'aplicació presenti vulnerabilitats o errors. Per exemple, el fet que jo mateix implementi la funcionalitat d'inici de sessió des de 0, pot comportar a un alt cost temporal de desenvolupament o pot comportar problemes de seguretat. En canvi, amb aquestes llibreries es redueix el cost temporal i es minimitza el risc d'errors i vulnerabilitats. A més a més, si s'utilitzen llibreries que estan recolzades per una gran comunitat de desenvolupadors, assegurant així que hi haurà actualitzacions en cas de fallades o vulnerabilitats.

1.4. Planificació del Treball

Per a dur a terme el projecte serà necessari el desenvolupament d'una aplicació multiplataforma que sigui capaç de funcionar en la majoria dels dispositius disponibles en el mercat perquè la màxima quantitat de gent es pugui beneficiar dels avantatges. Per a aconseguir els objectius fixats en l'apartat anterior serà necessari disposar dels recursos següents:

1. Servidor de desenvolupament producció i desenvolupament
    a. SO: Debian 12 Bookworm
    b. Node.js 20.11
    c. Gestor de base de dades: MongoDB V7
    d. Proxy invers/Servidor Web: Nginx
    e. Control de versions: Git i GitHub
2. Dispositius on fer tests
    a. Mòbils i tauletes amb Android (+V.11)
    b. Mòbils amb iOS (+V.14)
    c. Emuladors de dispositius iOS i Android.
    d. Diferents ordinadors amb diferents navegadors com: Chrome, Firefox, Edge, Opera…
3. Eines per al disseny
    a. Figma
    b. PlantUML
4. Perfils d'usuaris amb els quals es duran a terme els tests:
    a. Estudiants o treballadors que viuen en pisos compartits
    b. Persones que viuen en unitats familiars

Les tasques per assolir els objectius fixats i la seva planificació temporal són les següents:

Fase 0 (Gestió del projecte i documentació) 28/02/24 - 21/05/24 → 8h/setmana
- Tasques de gestió del projecte (revisions de requisits, validacions dels requisits) 28/02/24 - 21/05/24 → 2h/ setmana
- Tasques de documentació 28/02/24 - 21/05/24 → 6h/setmana

Fase 1 (Planificació del projecte) 28/02/24 - 11/03/24 → 48h
- Definició projecte (Abast, Objectius, Recursos) 28/02/24 - 01/03/24 → 24h
- Documentació de requisits 04/03/24 - 05/03/24 → 16h
- Investigació del mercat actual 06/03/24 - 06/03/24 → 8h
- Investigació de *frameworks* i llibreries més apropiades per a un desenvolupament àgil d'una aplicació multiplataforma 07/03/24 - 08/03/24 → 16h
- Planificació temporal 11/03/24 - 11/03/24 → 8h

Fase 2 (Disseny) 12/03/24 - 03/04/24 → 80h
- Disseny de l'arquitectura global del sistema 12/03/24 - 12/03/24 →8h

- Disseny de l'estructura de la base de dades 13/03/24 - 14/03/24 → 16h
- Modelatge visual del servidor (Diagrama de flux) 15/03/24 - 18/03/24 → 16h
- Modelatge visual de l'aplicació (Diagrama de flux) 19/03/24 - 20/03/24 → 16h
- Disseny UX 21/03/24 - 25/03/24 → 24h
- Disseny UI 26/03/24 - 28/03/24 → 24h
- Revisions i proves de disseny 29/03/24 - 03/04/24 → 24h

Fase 3 (Implementació) 04/04/24 - 20/05/24 → 264h
- Implementació del disseny 04/04/24 - 11/04/24 → 48h
- Implementació de la lògica de negoci (Servidor) 12/04/24 - 19/04/24 → 48h
- Implementació de la lògica de negoci (Client) 22/04/24 - 25/04/24 → 32h
- Implementació funcionalitat reconèixer tiquet 26/04/24 - 01/05/24 → 32h
- Proves unitàries i d'integració 2/05/24 - 06/05/24 → 24h
- Proves de funcionalitats 07/05/24 - 07/05/24 8h
- Proves de compatibilitat 08/05/24 - 08/05/24 → 8h
- Proves amb usuaris 09/05/24 - 13/05/24 → 24h
- Empaquetament de l'aplicació 14/05/24 - 15/05/24 → 16h
- Desplegament i posada en marxa 16/05/24 - 20/05/24 → 24h

Dedicació:
- Dies laborables: 8h
- Dies festius: 4h

Dates clau:

- 12/03: La planificació del treball haurà d'estar definida.
- 4/04: El disseny de l'aplicació haurà d'estar definit.
- 21/05: La implementació de l'app i del servidor hauran d'estar acabats.
- 11/06: El servidor haurà d'estar desplegat en un servidor de producció i tota la documentació haurà d'estar finalitzada.

A continuació es poden observar els diagrames de Gantt de cada fase:

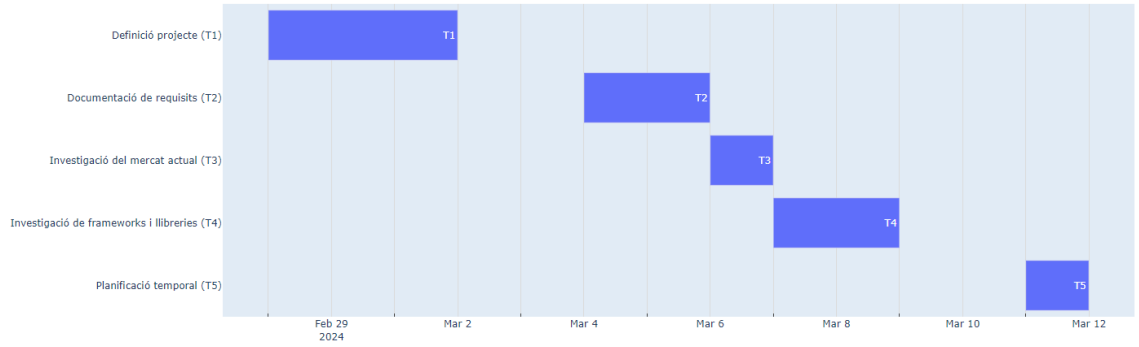


Fig. 1 Diagrama de Gantt de la Fase 1

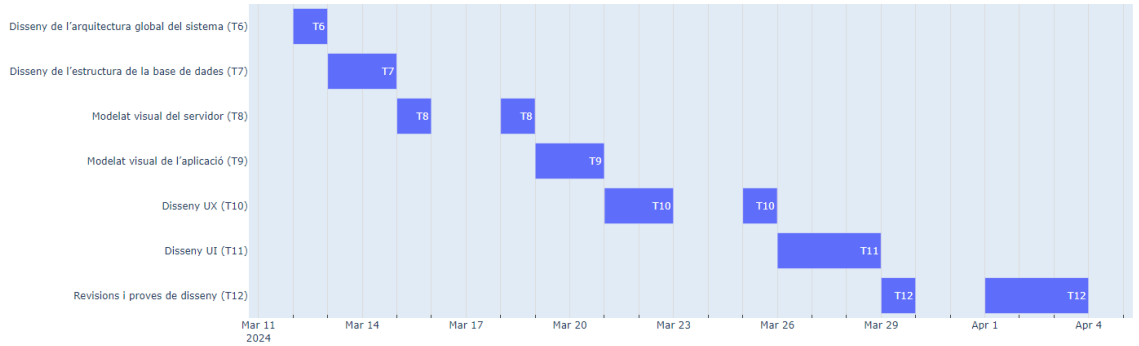


Fig. 2 Diagrama de Gantt de la Fase 2

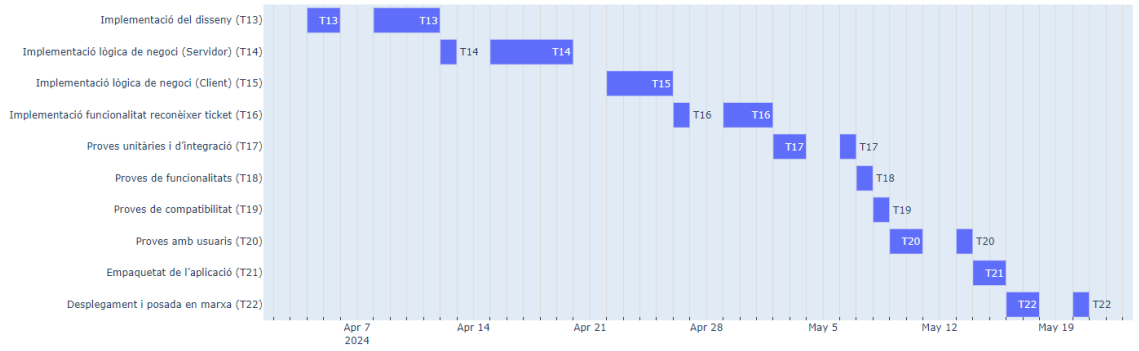


Fig. 3 Diagrama de Gantt de la Fase 3

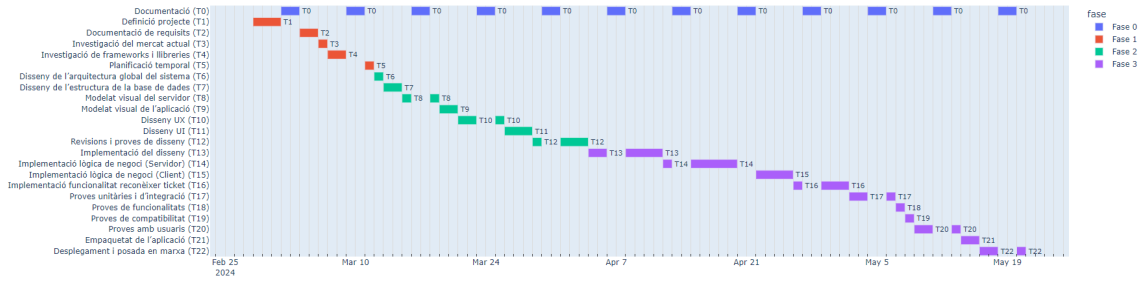


Fig. 4 Diagrama de Gantt de tot el projecte

1.5. Breu sumari de productes esperats

Els productes que s'obtindran un cop hagi finalitzat el projecte seran:

- Aplicació compatible en diferents plataformes, Android, iOS i en navegadors web.
- Petit manual de com funciona l'aplicació.
- Un contenidor de Docker que permeti el desplegament automàtic del servidor
- Manual de com fer el desplegament automàtic del servidor
- Memòria.
- Presentació de l'aplicació

1.6. Breu descripció dels altres capítols de la memòria

Els altres capítols de la memòria seran:

- Disseny i arquitectura: En aquest apartat s'analitzaran els perfils dels usuaris que utilitzaran l'app, els casos d'ús possibles de l'aplicació, el disseny de l'arquitectura general del sistema, el disseny de la lògica de negoci i el prototipat de la interfície gràfica de l'aplicació.
- Implementació: En aquest apartat es detallarà el procés de desenvolupament de l'aplicació. Es detallarà de manera exhaustiva totes les eines fetes servir i de quina manera s'han integrat a l'aplicació per aconseguir implementar totes les funcionalitats. A més a més, també s'explicaran les possibles complicacions trobades durant la implementació i les solucions aplicades. També s'inclouran els fragments de codi més rellevants i exemples per a detallar encara més el procés seguit durant el desenvolupament.
- Conclusions: En aquest apartat es presentaran les conclusions del treball, és a dir, un resum dels resultats assolits durant el treball i es discutirà sobre si s'han assolit els objectius plantejats a l'inici del treball i les lliçons apreses durant el procés.
- Glossari: En aquest apartat es proporcionarà un glossari de conceptes tècnics usats durant el treball per a facilitar la comprensió de tot el projecte.
- Bibliografia: En aquest apartat s'enumeraran totes les fonts d'informació fetes servir durant el projecte.
- Annexos: En aquest apartat s'inclouran els elements que complementaran el treball principal.

2. Disseny

Els mètodes de disseny centrat en l'usuari (DCU) són àmpliament acceptats en el camp del disseny d'interfícies i la interacció persona-ordinador, per la seva eficàcia demostrada en la creació de productes que responen millor a les necessitats dels usuaris. La recerca ha demostrat que els productes dissenyats amb aquest enfocament són més eficients, efectius i satisfactoris per als usuaris [4 *Ras, Aina. 2023*]. A més, aquestes tècniques permeten identificar problemes d'usabilitat en les primeres fases del disseny, reduint així el cost i el temps de desenvolupament. Aquestes pràctiques són considerades bones pràctiques perquè han demostrat millorar la qualitat del producte i la satisfacció de l'usuari.

2.1. Anàlisi

Els mètodes que se seguiran com a mètodes per a l'anàlisi del comportament dels usuaris són principalment:

- Benchmarking: La utilització d'aquesta tècnica permetrà comparar els productes que són competidors actuals de l'aplicació a dissenyar des del punt de vista de l'usuari. Aquesta tècnica és reconeguda per la seva capacitat de proporcionar un context comparatiu i establir estàndards de referència basats en les millors pràctiques de la indústria [5 *Estrategas Digitales. 2023*]. Permet als dissenyadors identificar oportunitats de diferenciació i innovació a partir de l'anàlisi crítica dels productes existents, sempre des del punt de vista de l'usuari.
- Avaluació heurística: La utilització d'aquesta tècnica facilita avaluar el disseny d'un producte com si ho fes un expert, d'acord amb un conjunt de regles i principis de disseny establerts prèviament. Aquesta tècnica és valorada [6 *Casabona, Eugenia. 2023*] per la seva eficiència en la identificació de problemes d'usabilitat amb una inversió relativament baixa de temps i recursos.
- Entrevistes a usuaris: La utilització d'aquesta tècnica possibilitarà un disseny centrat en usuaris que utilitzaran l'aplicació i d'aquesta manera s'aconseguirà un disseny més apropiat per a l'aplicació.

2.1.1. Benchmarking

2.1.1.1. Plantejament

Abans de començar amb el Benchmarking el primer el que caldrà fer serà triar i justificar quins seran els productes que s'analitzaran, aquests productes seran potencials competidors de l'aplicació que s'està desenvolupant. En aquest Benchmarking s'analitzaran els productes següents:

- Too Good To Go [7 *TooGoodToGo. 2023*]. Aplicació de referència en la lluita contra el malbaratament d'aliments. Present a diversos països de la UE amb origen a Dinamarca des de l'any 2016.
- Encantado de comerte [8 *Encantadodecomerte. 2024*]. Aplicació d'origen espanyol en la lluita contra el malbaratament d'aliments.
- Phenix [9 *Phenix. 2023*]. Aplicació d'origen francès involucrada també en la lluita contra el malbaratament d'aliments.
- Foody Bag [10 *FoodyBag. 2024*]. Aplicació d'origen australià semblant a les anteriors, però segons la seva descripció a Google Play implementa funcionalitats extra, com l'intercanvi d'aliments; tot i això, aquesta funcionalitat no s'ha pogut provar, ja que m'ha estat impossible registrar-me a l'aplicació, tant amb el meu número de telèfon, com en un número de telèfon australià.

Un cop s'han triat els productes els quals s'han d'analitzar a continuació es defineixen els criteris d'anàlisi. S'analitzaran els següents criteris sobre l'app. Tots aquests criteris s'han extret observant cada una de les aplicacions mencionades:

1. **Registre obligatori per a utilitzar l'app**. En aquest criteri s'avaluarà si l'aplicació obliga a iniciar sessió per a fer ús de les seves funcionalitats.
2. **Guia de com funciona l'App**. Aquest criteri avalua si existeix algun tipus d'explicació sobre les funcionalitats i objectius de l'aplicació.
3. **Maquetació** (*Layout*) **de l'App**. En aquest criteri s'avaluarà l'esquema de disseny de l'aplicació. És a dir si s'ha usat una *layout* amb barra d'eines, amb menú, si s'han utilitzat pestanyes... A més a més, en aquest criteri s'avaluarà l'accessibilitat a les diferents funcionalitats. És a dir, la dificultat que pot tenir un usuari per a trobar i accedir a una funcionalitat
4. **Mapa d'ofertes i establiments**. En aquest criteri s'avaluarà si existeix alguna funcionalitat per a explorar de manera interactiva els establiments
5. **Llista d'ofertes i establiments**. En aquest criteri s'avaluarà la funcionalitat per a explorar mitjançant una llista els establiments i/o ofertes disponibles
6. **Procés de pagament**. En aquest criteri s'avaluarà la forma de pagament feta servir en cada plataforma.
7. **Avaluar l'establiment**. En aquest criteri s'avaluarà la funcionalitat que permet als usuaris avaluar als establiments.
8. **Estadístiques**. Aquest criteri avalua si es mostren estadístiques als usuaris i als establiments per a fomentar l'activitat en l'aplicació.
9. **Accessibilitat**. Aquest criteri avalua l'accessibilitat de l'aplicació. Per a avaluar aquest criteri s'utilitzarà Talkback de Google i s'avaluarà si existeix la possibilitat d'adaptar els colors de l'aplicació.
10. **Verbositat**. Aquest criteri avalua el nivell de reciprocitat de l'aplicació amb l'usuari. És a dir, la freqüència en la qual l'aplicació informa l'usuari sobre el resultat de les seves accions.

Per a poder obtenir més detalls sobre l'anàlisi es pot trobar el desenvolupament de l'anàlisi del benchmarking a l'annex, a l'apartat 7.1.

2.1.1.2. Resultats i Conclusions

- **3 de 4** de les aplicacions **limiten funcions** fins que no **s'inicia sessió**, per tant, es farà el mateix en aquesta aplicació.

- **Totes** les aplicacions agrupen les funcionalitats per **pestanyes**. Per tant, es farà el mateix per fer la navegació intuïtiva a l'aplicació.

- **Totes** les aplicacions fan servir la component **targeta** per a mostrar el llistat d'ofertes, per tant, es decideix que s'emprarà un component del tipus targeta per a mostrar els resultats de les ofertes i establiments.

- **3 de 4** de les aplicacions contenen una **petita guia** o explicació dels objectius i funcionalitats principals de l'aplicació. Com que pot ser útil per a l'usuari saber de primera mà on trobarà totes cada una de les funcionalitats i saber cada funcionalitat que té l'aplicació, s'inclourà en l'aplicació un *onboarding* on s'explicaran les funcionalitats i els objectius principals de l'app.

- **2 de 4** aplicacions **notifiquen** a l'usuari constantment sobre les accions que està duent a terme i els possibles errors que es produeixen. És un fet a tenir en compte, ja que també es menciona durant l'avaluació heurística.

- **2 de 4** apps funcionen d'una manera correcta amb un **lector de pantalla** (*screenreader*) com pot ser Google Talkback. Com que és beneficiós per a usuaris que poden tenir algun tipus de dificultat de visió es desenvoluparà l'aplicació aplicant tècniques d'accessibilitat.

- **2 de 4** apps permeten **avaluar** numèricament als establiments els quals han realitzat com a comanda. Com que pot ser beneficiós pels usuaris saber l'opinió d'altres usuaris sobre l'establiment, s'implementarà la possibilitat d'avaluar numèricament l'establiment i a més a més es permetrà a l'usuari avaluar l'establiment mitjançant un comentari.

- 2 de 4 aplicacions permeten fer un petit seguiment de la quantitat de **lots salvats** per part de l'establiment, per tant, estaria bé afegir un paràmetre als establiments a la base de dades per a fer un seguiment de lots salvats.

- **Cap** aplicació ha estat preparada per a poder adaptar l'**esquema de colors** de l'aplicació i que l'usuari pugui triar entre diferents temes per a poder millorar la seva experiència en la utilització de l'app. Però per aconseguir fer una aplicació accessible al menú de configuració de l'aplicació es permetrà a l'usuari elegir entre diferents temes en el qual un estarà pensat per a millorar l'accessibilitat a usuaris amb problemes de visió.

2.1.2. Avaluació heurística

En aquesta fase d'anàlisi en l'avaluació heurística, es presenta el recurs amb la llista de verificació (*checklist*) que haurà de complir el prototipat del disseny. S'ha utilitzat com a base la referència següent [11 *Legaspi, Aaron, i Amit Jakhu. 2023*]

1. **Affordance**. És a dir la capacitat del producte perquè l'usuari s'anticipi a com utilitzar-lo.
   a. Els controls de l'aplicació donen com a resultat el que espera l'usuari.
   b. Les funcionalitats dels controls es determinen fàcilment en una primera vista de l'aplicació.
   c. Les icones simbolitzen clarament la funció o la informació a transmetre als usuaris.
2. **Feedback**. L'aplicació haurà de ser clara i interactiva amb l'usuari per informar-lo en cada moment que ha passat i que és el que està passant.
   a. Tots els estats dels elements i de les vistes es veuen clarament.
   b. Qualsevol estat el qual pot arribar l'aplicació és comprensible per l'usuari.
   c. Es proveeix d'una retroalimentació (*feedback)* quan una tasca és completada.
   d. Es proveeix un *feedback* quan una tasca ha donat error.
3. **Simplicitat i Organització**. L'aplicació haurà de ser el més simple possible i enfocada en la tasca per la qual ha estat programada.
   a. Cada passa d'un flux és evident per a l'usuari.
   b. Optimitzar la quantitat de símbols perquè siguin més fàcils de reconèixer.
   c. No representar múltiples accions amb símbols o icones similars.
   d. L'arquitectura i l'estructura de l'aplicació està organitzada clarament.
4. **Consistència**. L'aplicació haurà de ser consistent en totes les seves vistes per a proporcionar una navegació intuïtiva a l'usuari
   a. Les animacions i el moviment a través de les vistes són consistents en tot el sistema.
   b. Aprofitar elements existents en el disseny per a minimitzar la inconsistència.
   c. La localització dels elements són consistents en tot el sistema.
   d. El llenguatge usat és clar i consistent en tot el sistema.
   e. El text lligat a un símbol clarament representa el mateix que el símbol.
5. **Tolerància a fallades**. L'aplicació haurà d'estar capacitada per a prevenir i recuperar-se de possibles errors del sistema i de l'usuari.
   a. Els usuaris hauran de poder reconèixer quan han tingut un error, quin error han tingut i hauran de poder reconèixer com recuperar-se.
   b. Tota ajuda i suport haurà de ser fàcilment llegible per part de l'usuari.
6. **Accessibilitat**.
   a. Els símbols i els controls de l'aplicació hauran de complir una mida, espaiat i contrast mínims.
   b. La tipografia és òptima per a ser llegida en qualsevol entorn o context.
   c. Assegurar-se que existeixen alternatives en termes d'accessibilitat, descripció dels controls, dels fitxers multimèdia...

2.1.3 Entrevistes

2.1.3.1 Plantejament

Després de dur a terme el benchmarking per a estudiar la competència i després de l'avaluació heurística, es faran entrevistes als usuaris per a obtenir una comprensió més profunda sobre les seves necessitats i preferències per tal de millorar la seva experiència en l'aplicació. Es plantegen dues entrevistes. Una orientada als usuaris que utilitzaran l'aplicació com a clients i una altra orientada als usuaris que utilitzaran l'aplicació com a establiments.

Les entrevistes es poden trobar a l'apartat 7.2.

2.1.3.2. Conclusions clients

Tots els entrevistats compren aliments en el supermercat que tenen una data pròxima de caducitat sempre que poden sigui del tipus d'aliment que sigui. De tots els usuaris 2 han utilitzat una aplicació com la proposada i és Too Good To Go, ja que és la que té més varietat d'ofertes.

Un dels usuaris ha expressat que no té cap mena de problema o dificultat en lluitar contra el malbaratament d'aliments en la seva llar. Tot i això, alguns dels usuaris expressen que a causa de l'estrès diari, la falta de comunicació amb els companys o la gran quantitat d'activitats que realitzen, a vegades no poden portar mentalment un control dels aliments que tenen i compren més aliments del compte i a vegades això porta al residu d'aliments en les seves llars i trobarien útil una solució per a reduir aquest malbaratament.

Els aliments que més han anomenat els usuaris que estarien interessats a comprar a través de l'app són forn i menjars preparats. 2 dels 3 usuaris han expressat que no tindrien cap mena de problema en comprar qualsevol classe d'aliment a través de l'app, en canvi, el darrer usuari no comprarien aliments com carn o peix. Ja que la majoria han expressat que comprarien qualsevol classe d'aliment, no es pot descartar cap mena d'aliment, per la qual cosa es permetrà als usuaris que ja hagin comprat en un establiment, que el valorin per la qualitat dels aliments dels lots i incentivar així a tots els usuaris que comprin aliments de tota mena en els millors establiments.

Tots els usuaris han expressat que una de les característiques més importants que trobarien necessàries seria la de rebre notificacions sobre les últimes ofertes i establiments propers a la seva ubicació, així com poder avaluar els establiments segons certs criteris (com qualitat, quantitat...) i algun comentari. Dels usuaris que havien utilitzat l'aplicació de Too Good To Go expressen que també seria necessari poder explorar els establiments propers mitjançant el mapa i aplicar-hi filtres per a facilitar la cerca.

En aquest cas, 2 dels 3 usuaris han expressat que l'incentiu més apropiat podrien ser cupons de descompte i l'usuari restant ha expressat que l'incentiu

més apropiat és que per cada 5 o 10 comandes a l'aplicació s'obtingui un descompte. Per facilitat en la programació, s'aplicarà un descompte del 50% de la comanda cada 10 comandes d'un mateix usuari automàticament. Per a saber quan li toca rebre un descompte a cada usuari s'emmagatzemarà un comptador inicialitzat en 10 i que anirà decreixent a mesura que es facin comandes. Quan aquest comptador arribi a 0, es reiniciarà i s'aplicarà el descompte.

L'usuari que anteriorment ha expressat que no tenia cap mena de dificultat amb el malbaratament d'aliments en la seva llar, no troba necessari l'ús d'un inventari per portar un control sobre els aliments de la seva llar. Tot i això, els usuaris que si han expressat alguna dificultat, trobarien beneficiós el fet de poder portar un control del seu rebost.

2 dels 3 usuaris diuen que recomanarien l'aplicació si realment poguessin utilitzar l'aplicació diàriament beneficiant-se de descomptes, és a dir la recomanarien si la trobessin una aplicació útil. L'altra meitat dels usuaris expressen que si poguessin beneficiar-se d'avantatges o descomptes per cada usuari que els menciona en el registre.

2.1.3.3. Conclusions establiments

Segons els establiments entrevistats, els aliments que més es malbaraten en aquests establiments són les fruites, verdures, i també menjars preparats i dolços. Segons les conclusions extretes dels usuaris de tipus client, no tindrien cap mena de problema en comprar aquests aliments a través de l'aplicació. Tot i això, com s'ha decidit anteriorment no es descarta cap classe d'aliment.

Dels establiments entrevistats cap ha utilitzat anteriorment una aplicació per a reduir el malbaratament d'aliments. Indiquen que és a causa del desconeixement. Però tots dos coincideixen en el fet que és una opció que a partir d'ara tindran en compte. Per tant, són raons que fan pensar que pot arribar a ser una aplicació popular i usada en els establiments que no fan servir alguna aplicació d'aquest tipus, que com han dit els usuaris de tipus client, en zones en nuclis de població dispersos, l'oferta en aquestes aplicacions és molt baixa.

Els entrevistats indiquen que l'ús de l'aplicació seria beneficiós, ja que tots els establiments han de llençar algun aliment almenys 1 cop per setmana i normalment, intenten aprofitar ells mateixos aquests aliments per a no malbaratar-los. Afirmen que si poguessin obtenir algun tipus de benefici a partir d'aquests aliments, augmentaria substancialment la seva eficiència i estarien interessats a fer servir l'aplicació.

Un dels entrevistats fa servir un programa TPV en el qual té l'inventari, però únicament un disposa del nom de l'article relacionat amb el codi de barres i el preu. No tenen cap mena de control informatitzat sobre l'inventari i quantitats restants. Tots dos coincideixen en el fet que dissenyar un sistema que pugui

escanejar un albarà a través d'una fotografia amb el mòbil i portar un control sobre les dates de caducitat de cada element d'un inventari ho trobarien beneficiós per a la gestió del seu negoci.

La informació que majoritàriament els agradaria saber als establiments entrevistats seria la qualitat i la quantitat de les comandes que ofereixen. Per tant, serà important poder avaluar numèricament aquests dos aspectes i que puguin mostrar un resultat orientatiu per als dos tipus d'usuari.

Tots els entrevistats indiquen que no tindrien cap mena de preocupació si els pagaments s'efectuen a través de l'aplicació de forma segura en el moment en el qual es produeix el pagament i no són transaccions que es produeixen mensualment.

2.2. Disseny centrat en l'usuari

2.2.1. Perfils d'usuari

Si s'observa qualsevol de les aplicacions que s'han analitzat durant l'anàlisi de Benchmarking es pot arribar ràpidament a la conclusió que és necessari diferenciar entre dos tipus d'usuaris que són:

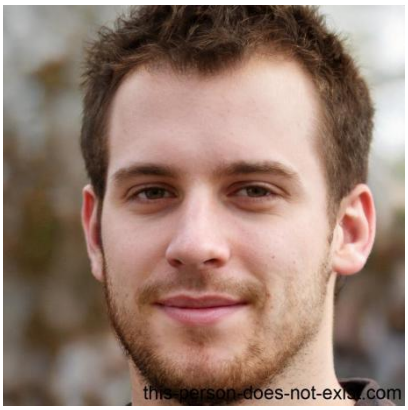
- Usuari de tipus **client**: Serà l'usuari el qual podrà explorar establiments i comprarà aliments oferits i posteriorment podrà avaluar-los.
- Usuari de tipus **establiment**: Serà l'usuari el qual podrà publicar ofertes i rebre comandes i comentaris dels usuaris de tipus client.

Perfil d'usuari de tipus client:

Josep 	- Sovint aprofita per comprar després de treballar o aprofita entre les hores de les activitats extraescolars dels seus fills. - Intenta establir una dieta equilibrada per a tota la família
- 30-55 anys - Viu a Mataró - Treballa a l'Hospitalet de Llobregat - Els seus fills estudien a Barcelona	- Accés a opcions de menjar saludable i assequible per a la família. - Valora la possibilitat d'explorar nous establiments segons la seva localització i d'una manera interactiva. - Necessitat de gestionar eficientment el rebost per a planificar menjades i evitar el malbaratament d'aliments - Interès a rebre notificacions sobre les últimes ofertes per a aprofitar les millors ofertes abans que s'esgotin. - Valoració en la comoditat i la facilitat d'ús de l'aplicació en qualsevol escenari. - Valoració en la rapidesa de gestió de l'inventari

	- Interès a rebre notificacions quan un aliment està a punt de caducar al rebost. - Valoració en la capacitat de valorar establiments i veure els millors establiments a prop meu.

Usuari de tipus establiment:

Carles  	- Alguns caps de setmana es veu obligat a fer hores extres per gestionar l'inventari del seu negoci. - Sovint canvia la seva dieta per aprofitar aliments a punt de caducar del seu negoci.
- 25 - 55 anys - Gerent en un petit negoci de queviures - Viu en un nucli de població petit	- Reduir el malbaratament d'aliments al final del dia - Augmentar les vendes podent arribar a nous clients interessats en ofertes d'aliments. - Valoració d'una plataforma fàcil d'utilitzar per a llistar i vendre excedents d'aliments. - Interès a rebre retroalimentació i comentaris dels clients per a millorar l'oferta d'aliments. - Millorar la gestió de l'inventari per evitar excedents innecessaris - Valora rebre estadístiques sobre l'ús de l'app en la gestió de l'inventari.

2.2.2. Points Of View

Les persones usuàries tipus client necessiten una forma d'explorar els establiments i ofertes disponibles a la seva població, per a poder accedir a ofertes mentre contribueixen a la lluita contra el malbaratament d'aliments en els negocis de proximitat a la vegada que es beneficien de rebaixes en els preus dels productes.

Les persones usuàries de tipus client, necessiten una forma d'explorar els establiments i ofertes properes de forma interactiva o segons la seva localització GPS per a explorar de forma còmoda les diferents ofertes.

Les persones usuàries de tipus client estan interessades a rebre notificacions de les últimes ofertes dels establiments propers per mantenir-se informat i aprofitar les millors ofertes.

Les persones usuàries de tipus client necessiten una forma d'avaluar els establiments per a poder informar a altres usuaris sobre el seu servei i ajudar a millorar l'establiment.

Les persones usuàries de tipus client necessiten portar l'inventari del seu rebost per a portar un control dels aliments de la llar i saber quan es va comprar cada aliment, a més a més, de rebre notificacions dels aliments a punt de caducar del seu rebost per a reduir el malbaratament d'aliments a la seva llar.

Les persones usuàries del tipus client necessiten afegir tots els aliments a l'inventari del seu rebost de manera automatitzada per a reduir el temps de gestió del rebost i sigui una eina útil en comptes de feixuga.

Les persones usuàries de tipus establiment necessiten una forma d'afegir, modificar i eliminar ofertes, sense límits, sobre els excedents d'aliments del seu negoci per a reduir el malbaratament d'aliments i augmentar les vendes.

Les persones usuàries de tipus establiment, necessiten veure de manera simple en un cop d'ull les últimes comandes per a gestionar de forma còmoda totes les comandes dels clients.

Les persones usuàries de tipus establiment necessiten una forma de veure les avaluacions i comentaris dels usuaris i ser notificats quan es produeixi una nova avaluació per a millorar el seu servei.

Les persones usuàries de tipus establiment necessiten poder portar un inventari de les existències del seu rebost per a millorar la gestió del seu negoci.

2.2.3. Fluxos d'interacció

2.2.3.1. Inici de sessió i registre

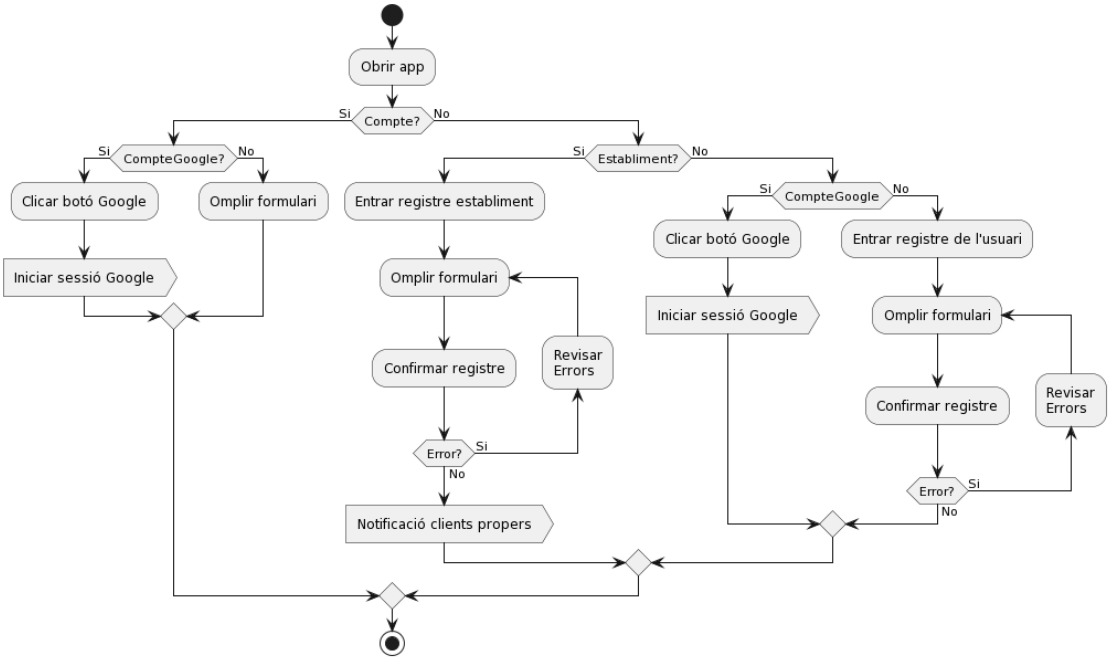


Fig. 5 Flux d'interacció d'inici de sessió

En l'anterior imatge es pot veure tot el flux d'interacció que seguirà un usuari o establiment en el procés d'inici de sessió o registre. En el cas que l'usuari sigui el primer cop que inicia l'aplicació no tindrà compte per a iniciar sessió i, per tant, tindrà tres possibilitats per registrar-se, una en el cas que sigui un establiment i dues en el cas que sigui un usuari. En conseqüència, si l'usuari és un establiment, veurà un enllaç que el conduirà a la vista de registre de l'establiment que li permetrà crear un compte per a iniciar sessió. En el cas que sigui un usuari de tipus client, aquest tindrà la possibilitat d'iniciar sessió o registrar-se automàticament amb el compte de Google. En el cas que l'usuari de tipus client no vulgui utilitzar el compte de Google o no en disposi en aquesta vista també li apareixerà un enllaç que li mostrarà el formulari de registre que li permetrà crear un compte a l'aplicació.

Si l'usuari no és el primer cop que fa servir l'aplicació i ja disposa d'un compte el qual no és de Google, únicament haurà d'omplir el formulari d'inici de sessió amb el correu i la contrasenya per a poder accedir a l'aplicació.

2.2.3.2. Cercar establiments propers i comprar oferta

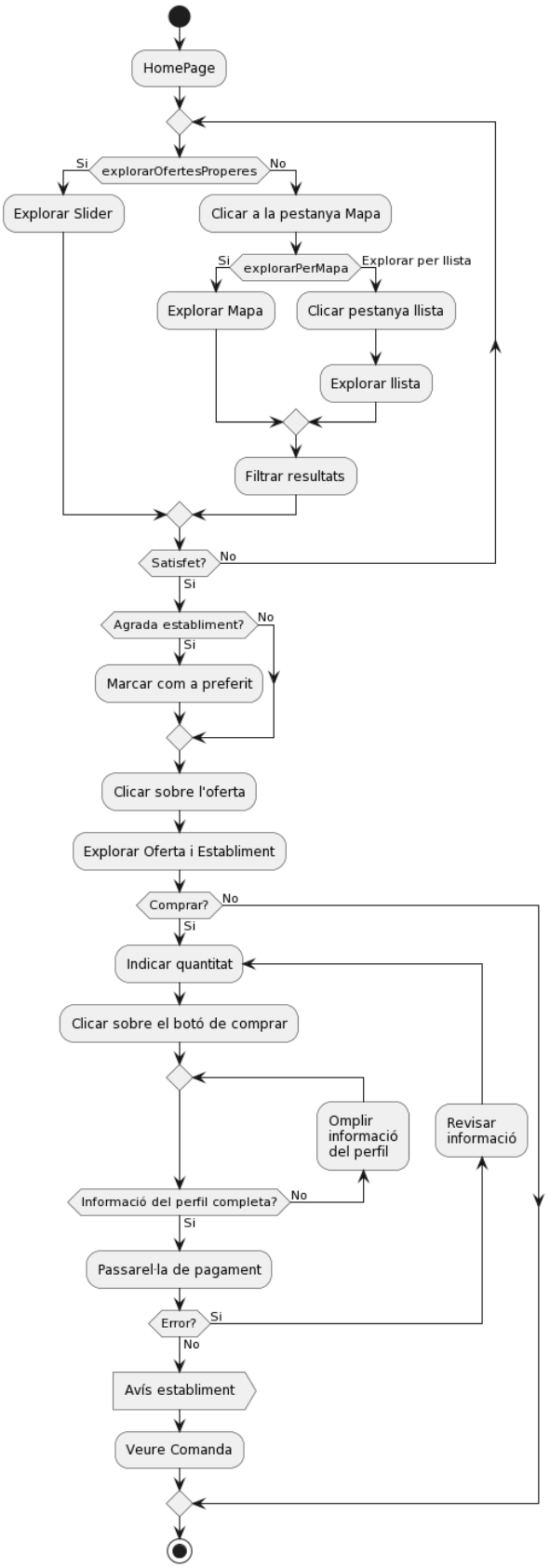


Fig. 6 Flux d'interacció d'exploració d'ofertes i establiments

L'usuari de tipus client tindrà diverses opcions per a cercar establiments propers. Posant de partida que l'usuari es trobarà a la pàgina principal, podrà explorar les ofertes properes des d'aquesta pàgina on apareixerà un carrusel (*slider*) amb les últimes ofertes més properes a la seva localització de registre. En el cas que no es trobin ofertes properes, l'usuari no té marcat cap establiment com a preferit o l'usuari prefereix explorar més ofertes; tindrà l'oportunitat d'anar a la pestanya anomenada Mapa. On tindrà dues opcions disponibles per a cercar més ofertes. En aquesta vista primerament li apareixerà un mapa i en la part superior un botó que permetrà canviar des de la vista del mapa al format de llista i, per tant, podrà explorar altres establiments mitjançant un mapa o una llista. En les dues vistes es permetrà a l'usuari filtrar els resultats de la seva cerca.

Si l'usuari no està satisfet amb els resultats trobats, sempre podrà desfer la seva cerca i tornar a triar un mètode de cerca per a trobar ofertes que satisfacin l'usuari. En el cas que l'usuari estigui satisfet sobre els resultats, aquest haurà de clicar sobre l'oferta corresponent i podrà explorar els detalls de l'establiment i de l'oferta des de la vista que li apareixerà. Quan finalment, l'usuari es decideixi per qualsevol oferta, haurà d'indicar la quantitat de lots que vol comprar i a continuació, haurà de clicar sobre el botó de comprar. En aquest punt, si l'usuari no té tota la informació necessària per a poder fer una comanda, aquest usuari es redirigirà a la vista de configuració de perfil perquè completi la informació necessària. Quan hagi completat aquesta tasca, ja podrà completar la compra. En el cas que durant el procés es produeixi algun error, es demanarà a l'usuari revisar la informació del seu mètode de pagament i la informació de la comanda per a intentar pal·liar l'error i es reenviarà a l'usuari de nou a la vista de l'establiment abans d'indicar la quantitat.

Un cop completada la tasca de la passarel·la de pagament, si no ha tingut lloc cap error, s'avisarà a l'establiment sobre la comanda, l'usuari podrà veure l'extracte de la comanda realitzada.

2.2.3.3. Gestionar ofertes

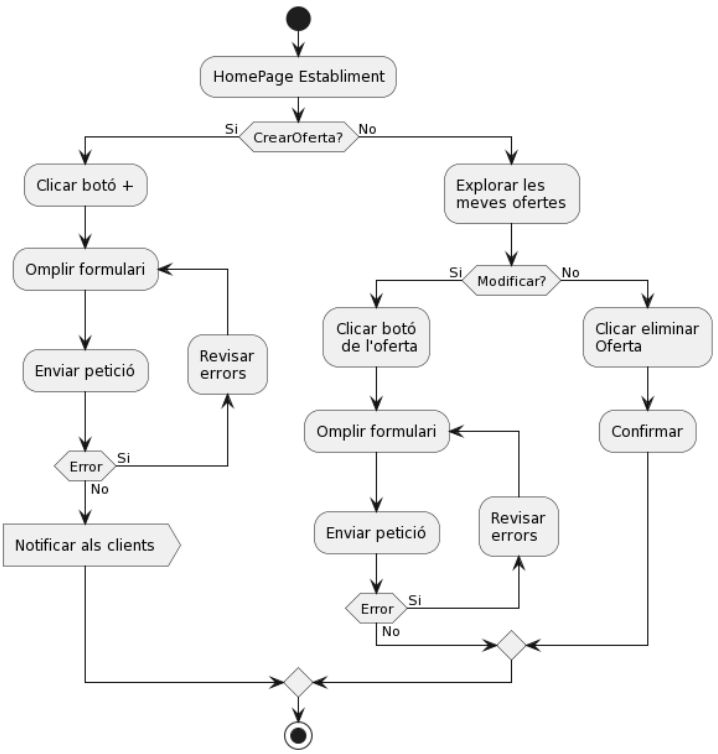


Fig. 7 Flux d'interacció de gestió de les ofertes per part d'un establiment

Quan un establiment vulgui gestionar les ofertes que té disponibles per a la venda o crear-ne una de nova, des de la pàgina principal de l'establiment podrà dur a terme totes aquestes accions. Des d'allà, l'establiment haurà de triar si crea una oferta o gestiona les ofertes existents. En el cas que l'establiment necessiti crear una nova oferta, caldrà clicar sobre un botó flotant, que mostrarà un formulari per a afegir ofertes. Quan es validi aquest formulari s'enviarà la petició al servidor, si aquest retorna error s'informarà l'usuari i es retornarà a la vista del formulari, en cas contrari, s'haurà completat el flux i l'usuari es trobarà de nou a la vista *HomePage* de l'establiment, on apareixerà l'oferta creada.

En el cas que l'usuari, necessiti modificar o eliminar una oferta, haurà de cercar-la a la llista que apareixerà a la mateixa vista de la *Home* de l'establiment. Un cop l'establiment trobi l'oferta desitjada, haurà de decidir entre modificar-la o eliminar-la. Prémer el botó corresponent i confirmar els canvis.

2.2.3.4. Gestionar rebost

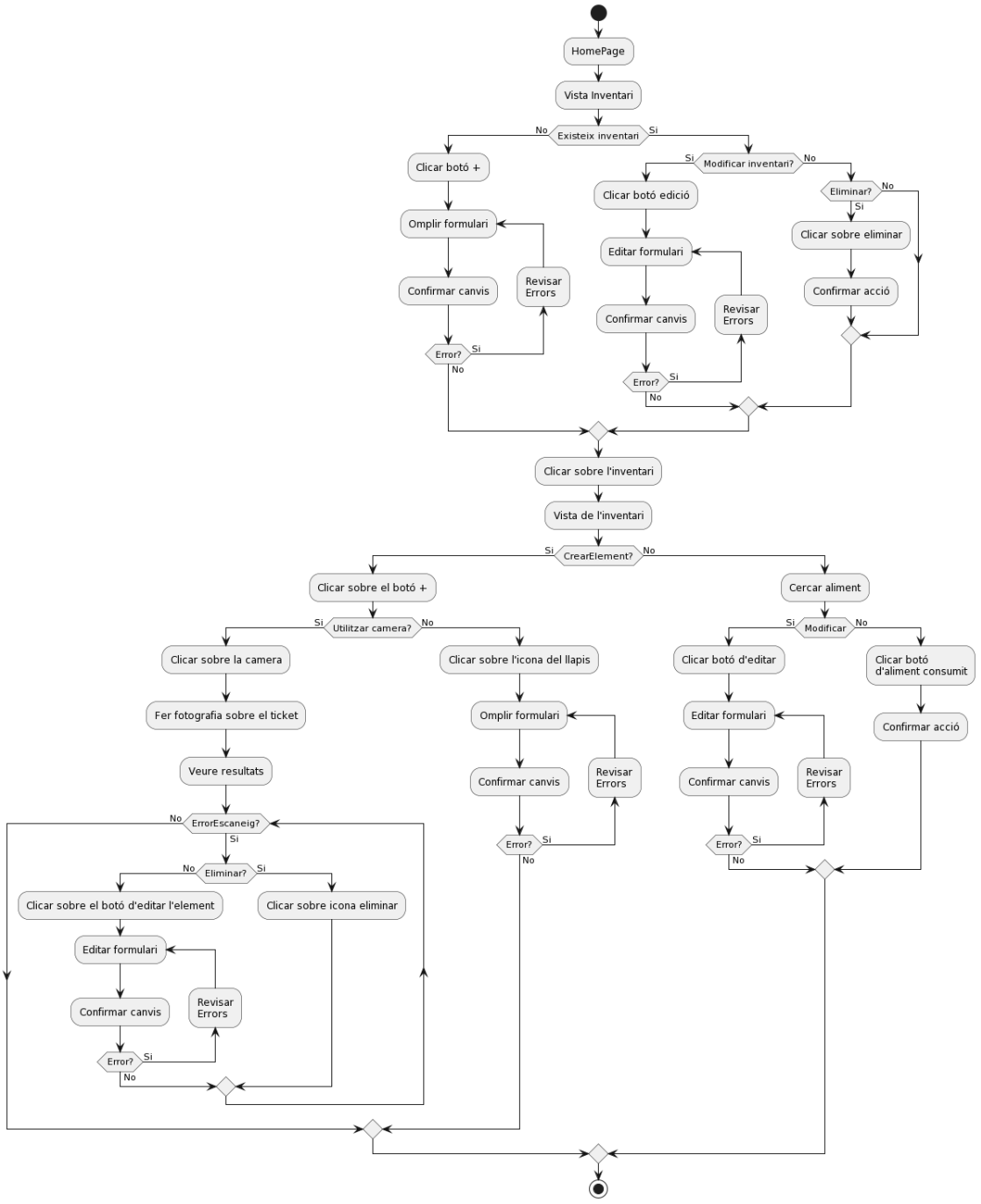


Fig. 8 Flux d'interacció de gestió d'un rebost o inventari

Un usuari o establiment podrà utilitzar aquesta funcionalitat, així doncs, per a poder utilitzar-la s'haurà d'ubicar a la pestanya Rebost. En aquesta pestanya si no existeix cap rebost, se'n podrà crear un de nou. Quan ja existeixin inventaris disponibles per a gestionar, apareixeran en forma de llista a la vista dels rebosts. Per a gestionar-ne un, s'haurà de clicar sobre el component que

representarà el rebost. En aquesta vista l'usuari tindrà disponibles totes les accions següents:

- Afegir un nou element al rebost manualment: Per a fer-ho, haurà de clicar sobre el botó flotant amb el símbol "+" a continuació omplir el formulari.

- Afegir una llista d'elements mitjançant la fotografia d'un tiquet de la compra: Per a fer-ho caldrà clicar a sobre del mateix botó del símbol "+" i a continuació a la icona corresponent a la càmera. A continuació, l'usuari podrà fer una fotografia mitjançant la càmera del dispositiu que estigui usant. Després de fer la fotografia, li apareixeran els resultats de l'escaneig del tiquet. En aquesta vista l'usuari, podrà modificar qualsevol dels elements escanejats prement el botó d'edició de l'element a modificar. L'usuari també podrà eliminar qualsevol dels resultats erronis. Quan l'usuari hagi corregit tots els errors dels elements escanejats, podrà afegir-los prement el botó de continuar.

- Modificar un element del rebost: Per a fer-ho, l'usuari haurà de buscar l'element que vol modificar i prémer el botó d'editar corresponent a l'element. Quan l'usuari cliqui sobre el botó li apareixerà un formulari amb les dades actuals per a modificar-lo.

- Marcar un element del rebost com a consumit (també es podria expressar com que s'elimina un element del rebost).

2.2.3.5. Veure comandes

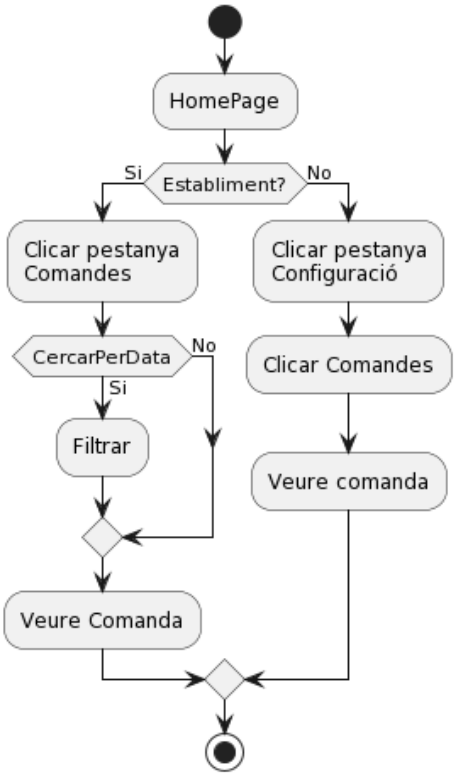


Fig. 9 Flux d'interacció per a veure comandes

Aquest és el flux d'interacció on un usuari podrà veure les comandes que ha comprat en el cas que sigui un usuari del tipus client o podrà veure totes les comandes realitzades en la seva oferta en el cas que sigui un usuari de tipus establiment.

2.2.3.6. Marcar establiment com a preferit

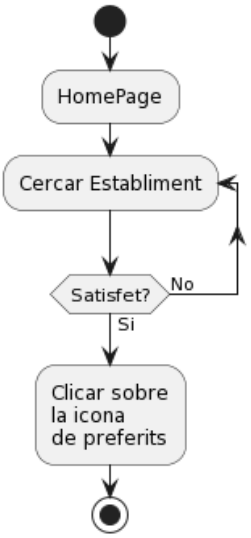


Fig. 10 Flux d'interacció de marcar/desmarcar un establiment com a preferit

L'usuari de tipus client serà l'únic que podrà marcar un establiment com a preferit. Per a fer-ho l'usuari haurà de trobar-se en la cerca d'un establiment. És a dir, l'usuari s'haurà de trobar en la llista de resultats de la cerca en mode llista o trobar-se en la vista de la informació detallada de l'establiment per a poder fer-ho. Quan l'usuari estigui satisfet amb un establiment únicament haurà de clicar sobre la icona del cor per a afegir-lo a la llista de preferits de l'usuari. Per a desmarcar un establiment de la llista de preferits l'usuari haurà de seguir el mateix flux i en un establiment que està marcat com a preferit únicament haurà de prémer de nou sobre la icona del cor.

2.2.3.7. Avaluar establiment

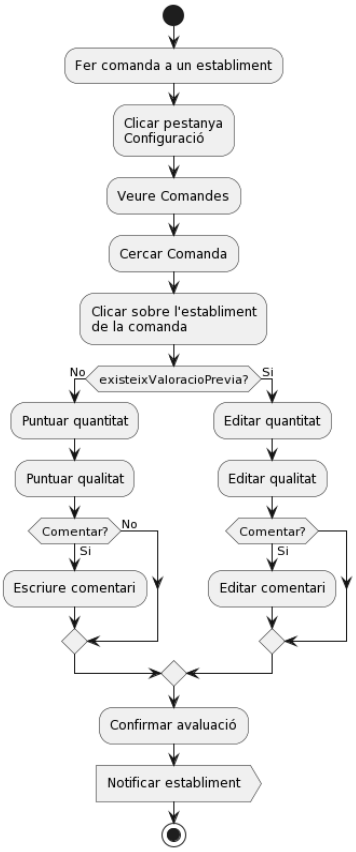


Fig. 11 Flux d'interacció per a avaluar un establiment

Per a poder avaluar a un establiment un usuari primer haurà d'haver fet una comanda a l'establiment. Per a poder accedir ràpidament a l'establiment que es vol avaluar es proposa el següent flux d'interacció com a forma més ràpida per a avaluar un establiment, però si l'usuari torna a cercar a l'establiment i entra dins de la pàgina de l'establiment, li apareixerà l'opció d'avaluar l'establiment. El flux proposat comença des de la pàgina d'inici i a continuació quan s'accedeix a la pestanya de configuració, l'usuari podrà veure totes les comandes realitzades si clica sobre l'enllaç de veure comandes del menú. Un cop en la vista de les comandes, l'usuari haurà de cercar la comanda feta a l'establiment i clicar sobre el nom de l'establiment. Un cop a la pàgina, l'usuari podrà d'avaluar mitjançant les icones d'estrelles el nivell de satisfacció amb la quantitat i la qualitat d'aliments, tal com s'ha especificat durant l'entrevista als usuaris. A continuació, si ho desitja, podrà deixar un comentari i haurà de confirmar els canvis per a guardar l'avaluació. Si l'usuari ja havia avaluat anteriorment l'establiment, podrà editar la seva avaluació mitjançant el mateix formulari, editant els camps que havia introduït anteriorment. Quan l'usuari confirmi els canvis en qualsevol de les dues possibilitats s'enviarà una *notificació push* a l'establiment per a informar que té una nova avaluació d'un usuari de tipus client.

2.2.3.8. Configuració

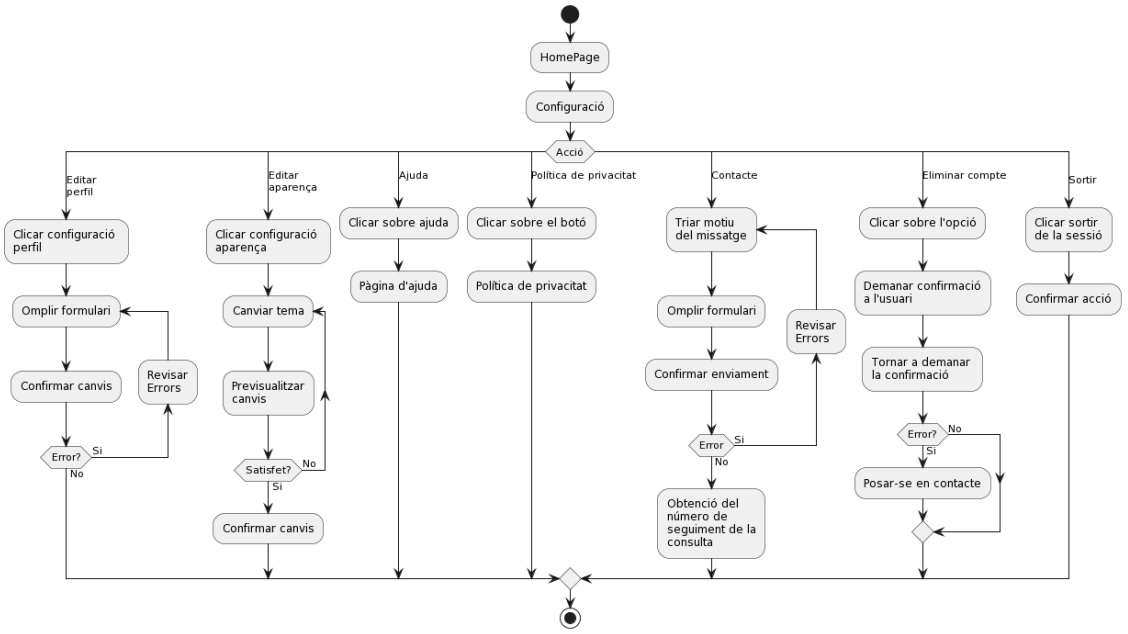


Fig. 12 Flux d'interacció de gestió de les ofertes per part d'un establiment

Des de la pestanya de la pàgina principal, tant els usuaris com els establiments tindran el mateix accés al menú de configuració de l'aplicació. En aquest menú apareixerà una llista de les diferents accions que es poden dur a terme:

- Editar el perfil: Li apareixerà un formulari on editar informació del perfil.
- Editar aparença: Aquí podrà canviar l'aparença de l'aplicació entre clar, fosc i alt contrast.
- Ajuda: Quan es cliqui sobre l'opció d'ajuda es mostrarà a l'usuari, *l'onboarding* de l'aplicació.
- Política de privacitat.: Quan es cliqui aquí es redirigirà a l'usuari a una pàgina amb la política de privacitat de l'aplicació.
- Eliminar compte: Si l'usuari desitja eliminar el seu compte, podrà fer-ho clicant a sobre d'aquesta opció.
- Sortir: Per a sortir sessió l'usuari haurà de clicar a sobre d'aquesta opció.

2.2.4. Sketchs

Abans de començar a fer proves directament a Figma es realitza un procés de *Sketching* on es generen esbossos de les principals vistes de l'aplicació segons el tipus d'usuari que servirà com a base per al disseny. En primer lloc, en la següent figura es mostren les 3 vistes principals de les pestanyes de l'usuari client, que són el *Home*, el mapa amb la llista d'establiments i la vista de configuració. També es pot veure l'aspecte que tindran els elements de tipus targeta. I que quan es cliquin aniran a la vista on es permet comprar l'oferta. També es veuen algunes de les funcionalitats que es trobaran a la pantalla de configuració:

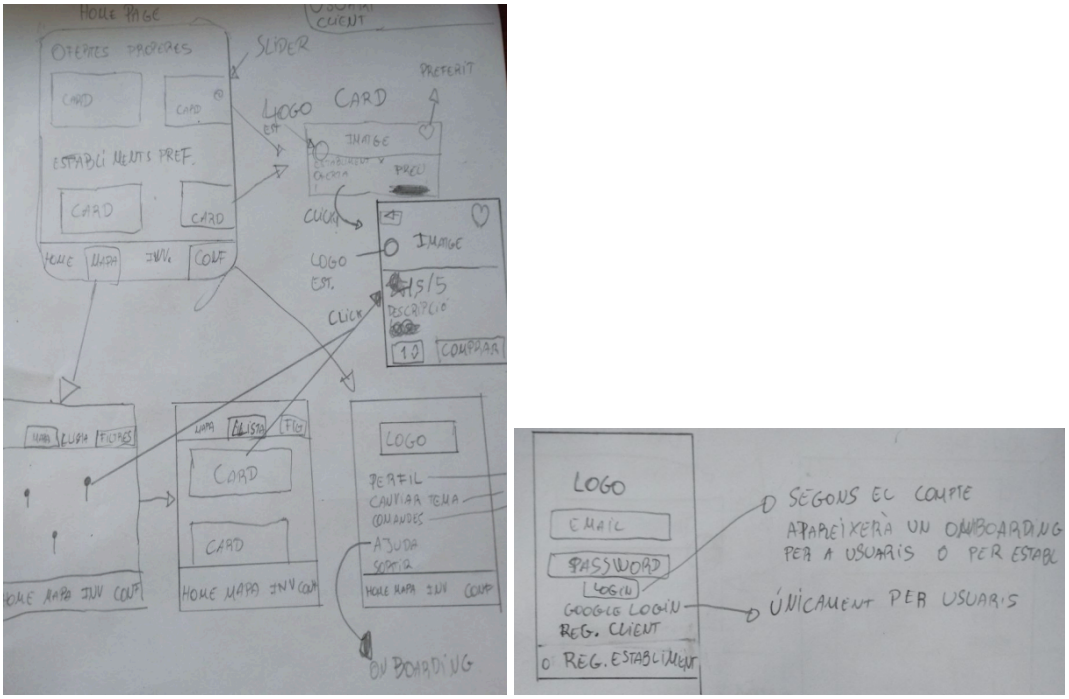


Fig. 13 Vistes principals de l'usuari i d'inici de sessió

Tal com s'havia decidit durant el benchmarking, no es podrà utilitzar l'aplicació si no s'inicia sessió. Tal com es pot observar, l'aplicació distingirà els comptes que comencin sessió a través d'un únic formulari i mostrarà *l'onboarding* corresponent si són de tipus client o de tipus establiment. Per als usuaris de tipus client se'ls permetrà obrir sessió amb el compte de Google. I a continuació, hi ha els dos enllaços que permeten registrar cada un dels tipus de compte possible.

En la següent captura, es pot veure el funcionament de la pestanya de l'inventari, on es permetrà a l'usuari crear rebosts, editar-los i eliminar-los així com afegir elements mitjançant un *Modal* amb un formulari o una fotografia. Aquestes vistes estaran disponibles tant pel client com per l'establiment.

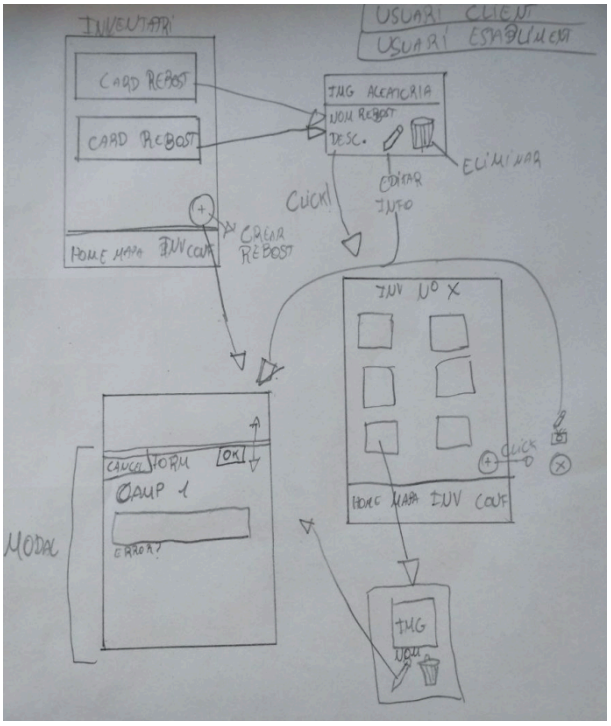


Fig. 14 Vistes principals de la gestió d'un inventari/rebost

Com que l'usuari i l'establiment no tenen els mateixos fluxos d'interacció, les vistes que veuran els usuaris de tipus establiment seran:

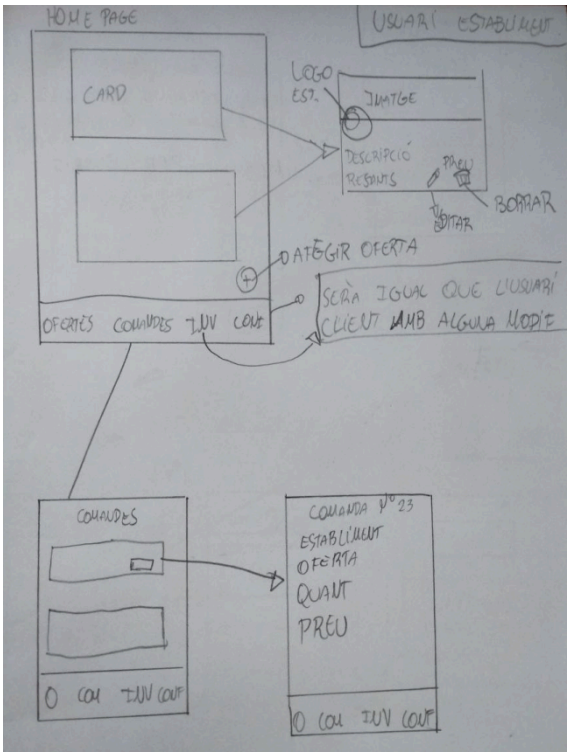


Fig. 15 Vistes principals d'un establiment

2.2.5. Prototipatge

Per al disseny del prototip de l'aplicació s'ha utilitzat Figma, ja que permet fer prototips interactius de manera ràpida i senzilla. A més a més, permet compartir el prototip mitjançant un enllaç. Cal remarcar que a causa de no disposar d'un compte de pagament hi ha certes funcionalitats de Figma que no he pogut usar com per exemple la utilització de variables que poden canviar de valor amb un clic, accés a components UI interactius... És per aquest motiu que en el prototip no s'inclouen diàlegs de confirmació d'accions com poden ser alertes, ni es pot mostrar com es validarien els formularis... Però encara que no s'incloguin en el prototipat s'hauran de tenir en compte tots els diàlegs on es demana la confirmació de l'usuari en els fluxos d'interacció anteriorment definits. Per aquest mateix motiu, tampoc es pot fer la simulació d'afegir ni eliminar objectes tot i que s'ha intentat representar el flux d'interacció de la forma més fidel possible. Tampoc s'ha aconseguit un prototip responsiu en les diferents vistes en què s'espera veure el dispositiu, s'anirà adaptant la responsivitat a mesura que s'implementi el disseny mitjançant estils CSS.

Encara que el prototip tingui les seves limitacions, s'ha intentat representar l'estructura de les diferents vistes i components de l'app seguint totes les decisions preses durant el benchmarking i les entrevistes i també s'ha dissenyat el prototip seguint les especificacions de l'avaluació heurística com poden són les animacions, les icones, el text... També s'ha seguit amb l'esquema general durant *l'sketching*. Durant aquesta fase de prototipatge també s'ha dissenyat el logotip el qual es pot veure en algunes vistes.

Tal com s'ha decidit durant el Benchmarking i s'ha esbossat durant *l'sketching* en aquest prototip s'ha implementat una *layout* en pestanyes. Però si l'usuari no ha utilitzat l'aplicació amb anterioritat, primerament es trobarà amb la pantalla d'inici de sessió, tot usuari que vulgui utilitzar l'aplicació haurà d'estar registrat i haurà d'haver iniciat sessió. En aquesta vista apareixerà un formulari amb dos camps, el correu i la contrasenya. Per a complir amb els requisits de l'avaluació heurística, es validarà de forma automàtica (mentre l'usuari estigui introduint les dades) els camps del formulari i s'informarà l'usuari dels errors mitjançant *labels* davall dels *inputs*. En el cas que es produeixi un error durant la petició d'inici de sessió al servidor, es mostrarà una alerta informant de l'error. A continuació del formulari, apareixerà un botó per a permetre a l'usuari iniciar sessió al sistema amb el seu compte de Google. Seguidament, davall d'aquest botó, apareixerà un enllaç informant l'usuari que si ho prefereix, podrà registrar-se mitjançant correu i contrasenya en una altra vista.

Si l'usuari prem sobre l'enllaç de registre, li apareixerà un *Modal* amb un formulari que li permetrà formalitzar el registre a la plataforma. Aquest formulari, també comptarà amb una validació automàtica que informarà l'usuari sobre els possibles camps erronis en el formulari. A més a més, també s'informarà l'usuari de qualsevol error durant la petició al servidor mitjançant una alerta.

En la vista d'inici de sessió, davall de l'enllaç de registre, apareixerà un altre enllaç que permetrà registrar comptes del tipus establiment. Quan es premi aquest enllaç es mostrarà un *Modal* amb un formulari de registre amb els camps necessaris per al registre d'un establiment. Com en tots els formularis els camps seran validats de forma automàtica i informaran l'usuari sobre els possibles camps invàlids del formulari. Per una altra banda, qualsevol error que sorgeixi durant la petició de registre al servidor es mostrarà una alerta informant del problema. Les vistes d'inici de sessió i de registre, tant de l'usuari com de l'establiment seran com les mostrades a continuació:

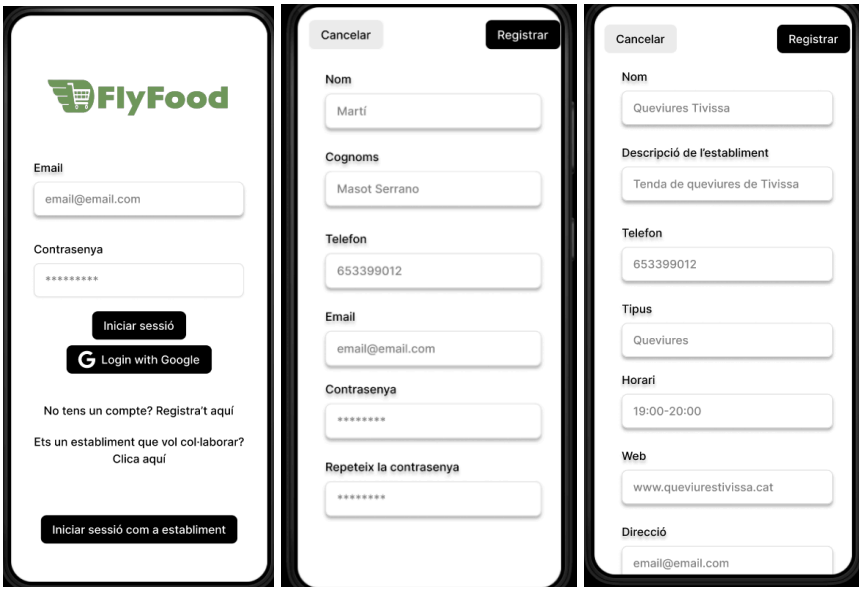


Fig. 16 Vistes del prototip d'inici de sessió i de registre del client i de l'establiment

El botó d'inici de sessió com a establiment no serà necessari en el disseny final, ja que tant els usuaris com els establiments iniciaran sessió en el mateix formulari d'inici de sessió.

Un cop l'usuari obre sessió visualitzarà un *onboarding*, pel fet que durant el Benchmarking s'ha decidit que era necessari perquè totes les aplicacions analitzades en tenien. En aquesta aplicació es mostraran dos tipus, segons si l'usuari que obre sessió és del tipus client o si és del tipus establiment. En el cas que sigui un usuari del tipus client li apareixerà el següent *onboarding*:

Fig. 17 Vista de *l'onboarding* de l'usuari

En el cas que inicií sessió un establiment, les dues primeres pantalles de *l'onboarding* seran diferents, ja que un establiment no podrà explorar ni comprar ofertes d'altres establiments.

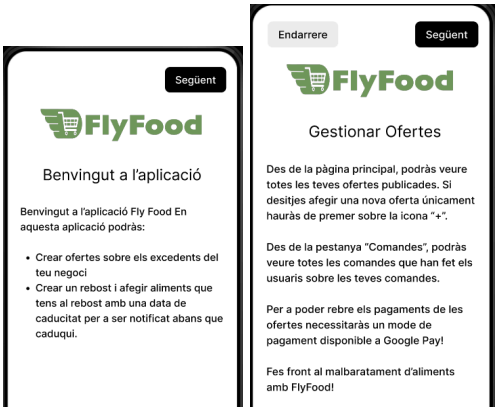


Fig. 18 Vista de *l'onboarding* de l'establiment

Després de *l'onboarding* corresponent a cada tipus d'usuari es mostrarà la pantalla d'inici de l'aplicació. Aquesta vista serà la que primer es visualitzarà cada vegada que l'usuari torni a entrar a l'aplicació i tingui la sessió iniciada. En aquesta vista, es podrà veure ja la *layout* de pestanyes que permetrà a l'usuari navegar entre les diferents funcionalitats de l'aplicació.

Si l'usuari que inicia sessió no és un perfil d'establiment apareixeran les següents pestanyes:

- **Pàgina principal**. A la barra de pestanyes s'utilitzarà una icona d'una casa amb el text "Inici". S'ha escollit aquesta icona perquè normalment s'usa en la majoria d'aplicacions per a simbolitzar la pàgina d'inici. En aquesta pestanya li apareixeran estadístiques rellevants a l'usuari sobre l'ús de l'aplicació i recomanacions d'ofertes d'establiments propers i dels favorits.
- **Mapa o explorar**. A la barra de pestanyes s'usarà una icona d'un mapa amb el text corresponent que serà "Mapa". S'ha fet servir aquesta icona

perquè simbolitza clarament que en aquesta vista es trobarà un mapa interactiu en el qual l'usuari podrà relacionar ràpidament que en aquesta vista es trobarà un mapa interactiu on explorar establiments. A més a més, dins d'aquesta vista es podrà canviar el format de visualització dels establiments en tipus llista.

- **Rebost**. A la barra de pestanyes s'emprarà una icona d'una caixa amb el text corresponent que serà "Rebost". S'ha usat la icona d'una caixa, ja que simbolitza que en aquesta vista l'usuari podrà gestionar un inventari.
- **Configuració**: A la barra de pestanyes es farà servir una icona d'un engranatge amb el text corresponent que serà "Config". S'ha usat la icona d'un engranatge, pel fet que en la majoria d'aplicacions i pàgines web s'utilitza aquesta icona per a simbolitzar aquest apartat. En aquesta pestanya es podran accedir a funcionalitats secundàries que es detallaran a l'explicació corresponent a la vista.

Si l'usuari que inicia sessió és un establiment tindrà les pestanyes següents:

- **Ofertes**: A la barra de pestanyes s'utilitzarà una icona d'una bossa de la compra amb el text corresponent que serà "Ofertes". En aquesta pestanya l'establiment podrà gestionar les ofertes disponibles per als usuaris. S'ha fet servir aquesta icona, ja que simbolitza el fet de comprar o vendre i transmetrà correctament l'acció a l'usuari.
- **Comandes**: A la barra de pestanyes s'usarà una icona d'un paper escrit amb el text corresponent que serà "Comandes". En aquesta vista l'establiment podrà visualitzar totes les comandes fetes per part dels usuaris. Per aquesta pestanya s'ha emprat aquesta icona perquè la llista simbolitza el tiquet de la compra que a la vegada simbolitza veure les comandes fetes mitjançant l'App.
- **Rebost**: Aquesta pestanya tindrà la mateixa icona, text i funcionalitats internes que per als usuaris.
- **Configuració**: Aquesta pestanya tindrà la mateixa icona i text per als usuaris tot i que les funcionalitats disponibles canviaran dels usuaris als establiments.

Fig. 19 Vista de la barra de pestanyes de l'usuari (esquerra) i de l'establiment (dreta)

A continuació, s'explicaran cada una de les vistes mencionades anteriorment i totes les vistes que depenen d'aquestes pestanyes.

Pàgina principal

En aquesta vista l'usuari podrà veure en primera instància les estadístiques rellevants de l'usuari segons la seva activitat en l'aplicació, com per exemple el nombre de kg salvats, quantitat de lots salvats... A més a més, li apareixerà un carrusel amb les últimes ofertes publicades en la seva localitat segons la població del registre i les últimes ofertes dels establiments marcats com a favorits, tal com es pot observar en la següent figura:

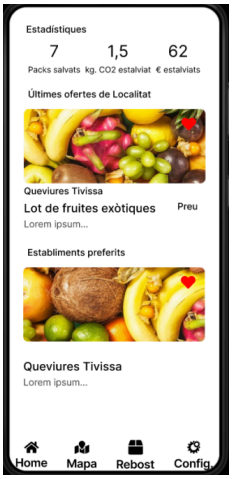


Fig. 20 Vista de la pàgina principal de l'usuari

Per les limitacions comentades anteriorment no s'ha pogut representar el carrusel en aquesta vista tal com s'havia esquematitzat en l'esbós, ja que representar un carrusel amb Figma es necessiten components prèmium als quals no es tenen accés. Tant per a representar els establiments com les ofertes s'utilitza un component de tipus targeta. Aquest component permet agrupar tota la informació i les accions disponibles en cada element.

Mapa

En aquesta vista l'usuari podrà interactuar amb un mapa interactiu i veure els establiments que tenen ofertes disponibles mitjançant la plataforma. Des dels marcadors disponibles en el mapa, l'usuari podrà accedir directament a la vista de l'establiment, explorar les ofertes disponibles i comentar sobre el servei rebut. A més a més, la vista del mapa, contindrà un botó, que permetrà canviar la vista dels establiments d'un mapa a una llista. Des d'aquesta llista, l'usuari veurà els mateixos establiments que en el mapa, però en format de llista. Clicant sobre qualsevol establiment disponible podrà visualitzar la informació de l'establiment, les ofertes disponibles i donar una opinió sobre l'establiment. L'usuari, ja estigui en la vista del mapa com en la vista de la llista, podrà filtrar els resultats a partir de les seves preferències que podrà editar prement sobre el botó filtrar que es troba a la part superior dreta de la vista.

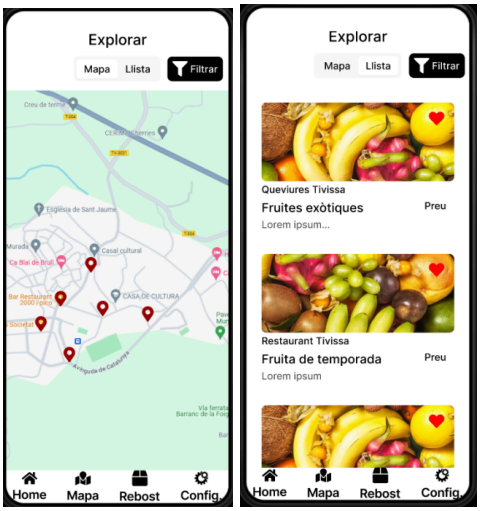


Fig. 21 Vista de la pestanya Mapa/Explorar

Filtrar

Tal com s'ha mencionat en la vista del mapa i de la llista d'establiments hi ha un botó el qual permet filtrar els resultats de la cerca. Aquest botó mostrarà un *Modal* el qual tindrà la següent aparença:

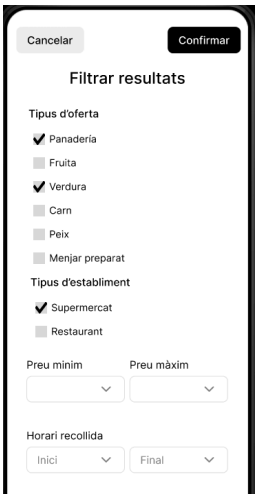


Fig. 22 Vista del formulari de filtrar resultats

Gràcies a aquest formulari es podran triar els tipus d'aliments i els tipus d'establiments que es volen filtrar durant la cerca. Per a poder triar els diferents tipus d'ofertes i els diferents tipus d'establiments s'han utilitzat grups de caselles d'opcions (*checkbox*), ja que permeten triar diferents opcions de manera simultània i en el cas que l'usuari no marqui cap *checkbox* indicarà que no s'aplicarà cap filtre d'aquest tipus. També es podrà filtrar pel preu i per l'horari de recollida. Un cop l'usuari hagi triat els criteris corresponents haurà de clicar sobre confirmar per a filtrar els resultats en la cerca al mapa o a la llista.

Rebost

En aquesta vista, apareixerà un llistat de tots els rebosts que ha creat l'usuari i un botó flotant amb la icona "+" que en ser clicat mostrarà un *Modal* amb un

formulari que permetrà crear un rebost. Els rebosts creats apareixeran com una llista de components tipus targeta, que seguiran el següent disseny:

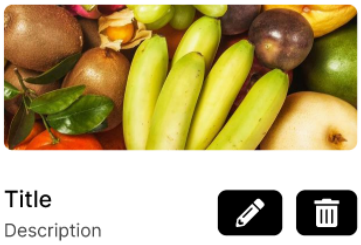


Fig. 23 Vista del component tipus targeta per a rebosts

Aquesta component contindrà com a títol el nom del rebost, una petita descripció i dos botons, un que permetrà eliminar el rebost i l'altre que permetrà editar-lo. Per aconseguir un desenvolupament més ràpid i àgil, quan un usuari vulgui editar el nom de l'inventari li apareixerà el mateix *Modal* que s'utilitza per a crear un nou inventari, però el *Modal* realitzarà la petició corresponent amb les dades del formulari segons si s'edita (**PUT**) o es crea (**POST**) un nou element de forma automàtica. Cal recordar que aquest formulari també es validarà de forma automàtica i informarà l'usuari sobre els possibles errors de validació de cada camp. Finalment, si es prem sobre qualsevol lloc de la targeta d'un rebost, es mostrarà la vista de gestió del rebost:

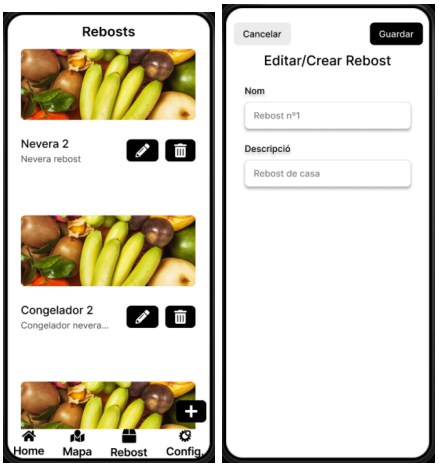


Fig. 24 Vista de la llista de rebosts d'un usuari i el *Modal* d'edició/creació

Gestió del rebost

En aquesta vista apareixerà una llista de tots els elements que es troben al rebost. Aquí serà on es durà a terme tota la gestió del rebost. A la part inferior dreta es pot veure un botó flotant amb la icona "+". Aquest botó permetrà a l'usuari afegir nous elements al rebost. Aquests elements es podran afegir mitjançant un formulari o mitjançant una fotografia d'un tiquet. És per això que quan es premi aquest botó apareixerà un desplegable amb dos botons més.

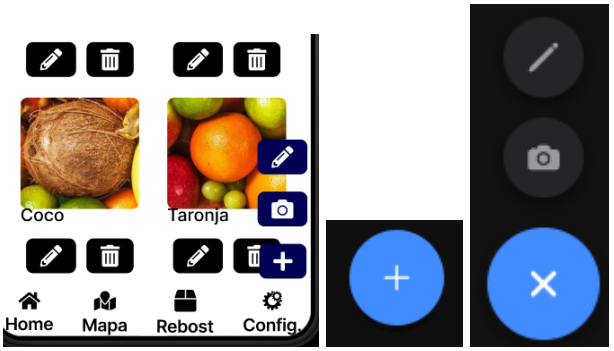


Fig. 25 Gestió d'un inventari i comparació del desplegable a Figma i Ionic

D'esquerra a dreta de la figura anterior es pot veure en primera instància el disseny de la vista de gestió d'inventari al prototip amb els 3 botons que simbolitzen l'animació final que quedarà amb *Ionic* una vegada implementat el disseny a l'aplicació. El botó amb la icona del llapis farà aparèixer el formulari que permetrà crear un nou element dins de l'inventari de forma manual. També cal recordar que aquest formulari serà validat de forma automàtica i s'informarà l'usuari dels possibles errors de validació dels camps del formulari i també s'informarà mitjançant alertes, els possibles errors durant la petició POST al servidor.

Fig. 26 Vista del formulari per a crear o editar un element del rebost

Per l'altra banda el botó amb la icona de la càmera reencaminarà a l'usuari a la càmera del sistema que estigui utilitzant. Si està usant Android usarà l'aplicació nativa de la càmera del dispositiu, si usa iOS també farà servir l'aplicació nativa de la càmera del dispositiu i si fa ús d'un navegador web, se li demanarà a l'usuari que indiqui quina càmera web vol emprar. Un cop a la càmera, l'usuari farà una foto i posteriorment li apareixerà un diàleg si vol fer ús de la foto que es mostra com a resultat o si vol repetir el procés.

Quan l'usuari verifiqui la fotografia, l'aplicació executarà un *OCR*, en aquest cas serà una mena d'embolcall (*wrapper*) per a JavaScript del popular Tesseract-OCR. L'*OCR* retornarà el text detectat a la imatge i a continuació, una funció cercarà paraules clau en tot el text detectat. Per exemple si es

detecta la paraula clau "logurt", afegirà als resultats un possible element del tipus logurt amb una data aproximada de caducitat.

Un cop l'*OCR* hagi escanejat la fotografia i hagi escanejat el text que ha extret de l'*OCR* i detecti els noms de la taula anterior. L'aplicació mostrarà una vista on apareixeran els ítems detectats en l'escaneig. Es mostraran en format de targeta i amb el nom i la data de caducitat aproximada. En aquest punt si existeix algun error en l'escaneig, l'usuari podrà eliminar o editar els resultats mitjançant dos botons que es trobaran en cada targeta de cada ítem. El d'eliminar l'ítem i el botó per a actualitzar-lo, que farà aparèixer un formulari per a editar-lo. Finalment, si l'usuari està conforme amb l'escaneig, podrà confirmar que els ítems s'afegeixin al rebost.

Fig. 27 Vista de correcció dels elements després de l'escaneig del tiquet

Com s'ha dit anteriorment, quan hi hagi ítems existents al rebost, a la pàgina de gestió del rebost apareixeran una llista de targetes que amb la informació de cada aliment del rebost. En aquesta vista es podran editar els ítems del rebost, mitjançant el botó d'edició. Per altra banda, cada targeta tindrà un botó que permetrà marcar qualsevol aliment com a consumit. En el cas que un dels ítems del rebost arribi a la data de caducitat i no s'hagi consumit, l'aplicació notificarà a l'usuari mitjançant una *notificació push* avisant-lo que és l'últim dia per a consumir l'aliment.

Fig. 28 Vista de la gestió d'un rebost

Configuració

En aquesta pestanya apareixerà una vista d'una llista amb diferents funcionalitats secundàries. En el cas de l'usuari de tipus client veurà les següents opcions:

- **Configuració del perfil**: Quan es premi sobre aquest botó apareixerà un *Modal* amb un formulari que li permetrà canviar informació referent al perfil.
- **Canviar contrasenya**: Quan l'usuari premi sobre aquest botó li apareixerà un *Modal* amb un formulari per a canviar la contrasenya
- **Canviar aparença**: Aquest menú permetrà a l'usuari canviar l'esquema de colors de l'aplicació.
- **Comandes**: En aquesta vista l'usuari podrà veure totes les comandes fetes i accedir a una vista més detallada de cada una.
- **Política de privacitat**: Serà un enllaç que reencaminarà a l'usuari a la vista on s'exposa la política de privacitat de l'aplicació.
- **Ajuda**: Quan l'usuari premi sobre aquest botó es reencaminarà a *l'onboarding* de l'aplicació.
- **Eliminar compte**: Permetrà a l'usuari eliminar el compte i totes les seves dades.
- **Sortir de la sessió**. Permetrà a l'usuari sortir de la sessió.

Fig. 29 Vista de la configuració de l'usuari (esquerra) i de l'establiment (dreta)

Per una altra banda, un establiment tindrà les mateixes opcions que l'usuari excepte el menú de les comandes. Ja que l'establiment ja tindrà una pestanya per a les comandes i no serà necessari accedir-hi des d'aquesta vista. En comptes d'aquesta opció tindrà l'opció d'"Avaluacions i comentaris" on podrà veure una llista de tots els comentaris i la mitja de les avaluacions.

Gestionar Ofertes

Aquesta vista serà la que els establiments veuran a la primera pestanya de la seva barra del menú inferior. En aquesta vista els establiments, podran gestionar les ofertes que han creat o afegir noves ofertes al sistema. Les ofertes ja creades apareixeran en format de llista com a components del tipus targeta. En cada targeta apareixeran dos botons, un que permetrà l'eliminació de l'oferta i un altre botó que farà aparèixer un *Modal* on es mostrarà un formulari que permetrà a l'establiment modificar l'oferta. Aquest formulari també es validarà de forma automàtica i notificarà a l'usuari sobre possibles errors de validació.

Per l'altra banda, a la part inferior dreta de la vista on es veurà la llista d'ofertes publicades, apareixerà un botó flotant que permetrà a l'establiment afegir noves ofertes al sistema. Quan es premi aquest botó apareixerà el mateix *Modal* que s'utilitza per a modificar una oferta, per estalviar temps en desenvolupar més vistes de les necessàries.

Fig. 30 Vista de la gestió d'ofertes per part d'un establiment

Comandes

Aquesta vista estarà disponible tant per als usuaris com per als establiments i els permetrà visualitzar les últimes comandes, cada element de la llista serà un component del tipus targeta en el qual apareixerà l'identificador de l'usuari i els elements de la comanda que ha realitzat.

Fig. 31 Vista de veure les comandes d'un establiment (esquerra) i veure les comandes fetes per un usuari (dreta)

Quan l'usuari cliqui sobre la comanda que desitja veure'n els detalls li apareixerà una vista amb tota a informació relacionada amb la comanda en forma de llista simple.

Veure Establiment

Quan un usuari està explorant establiments o ofertes a través de la pàgina principal o des del mapa, quan premin damunt d'algun establiment o oferta seran reencaminats a aquesta vista on veuran la informació relacionada amb l'establiment i les ofertes que ofereix l'establiment. En aquesta vista l'usuari podrà veure el logotip de l'establiment, el nom, una petita descripció de l'establiment, distintius i les valoracions de l'establiment. Així com un llistat de totes les ofertes que ofereix en aquell moment l'establiment.

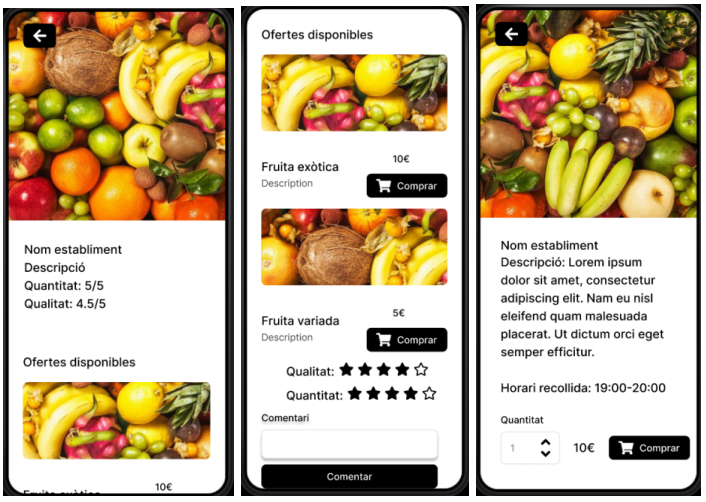


Fig. 32 Vista d'un establiment amb les ofertes disponibles, el formulari d'avaluació i la vista detallada d'una oferta

Si l'usuari està interessat a comprar alguna de les ofertes disponibles per l'establiment, haurà de clicar sobre l'oferta per accedir una vista més detallada de l'oferta. En aquesta vista més detallada s'especificaran els possibles articles que podrà trobar dins del lot, l'horari a partir del qual podrà a passar a recollir la

comanda i el preu total. A més a més, apareixerà un botó que permetrà fer la transacció per a comprar finalment l'oferta.

Configuració del perfil

En aquesta vista a l'usuari li apareixerà un *Modal* amb un formulari i la informació existent a la base de dades sobre el seu perfil. En aquesta vista, l'usuari podrà editar qualsevol dada del formulari. Sempre que les dades introduïdes siguin correctes, quan l'usuari premi sobre confirmar es guardarà la informació a la base de dades mitjançant una petició a l'API. Si alguna de les dades no és correcta, com que el formulari estarà validat automàticament, s'informarà l'usuari sobre els camps que no són correctes i quines regles necessiten complir abans de guardar els canvis. Si durant la petició al servidor hi ha algun problema, s'informarà l'usuari mitjançant una alerta.

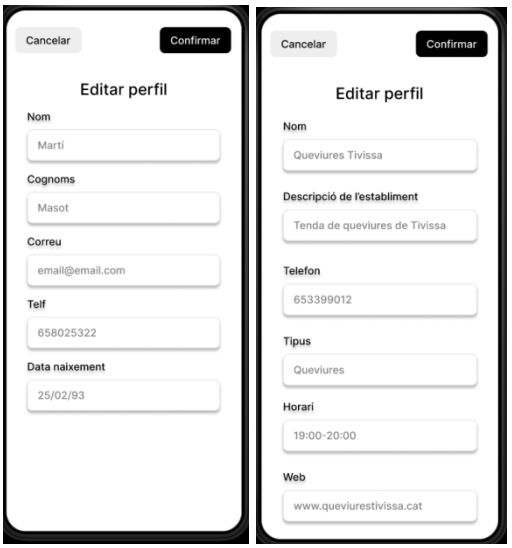


Fig. 33 Vista de la configuració del perfil de l'usuari client (esquerra) i de l'establiment (dreta)

Canviar contrasenya

Aquest formulari s'ha separat de la vista d'editar el perfil de l'usuari, ja que és possible que un usuari vulgui editar el seu perfil sense haver de canviar la contrasenya. Per a facilitar la implementació de les regles de verificació dels camps del formulari de les dues accions s'ha decidit què serà millor separar les accions en dos formularis per separat.

Fig. 34 Vista del formulari per a canviar de contrasenya

Configuració de la vista

En aquesta vista a l'usuari li apareixerà un *Modal* amb un formulari amb un grup de *Radio Buttons* on l'usuari podrà escollir entre 3 esquemes de colors disponibles i veure una petita previsualització dels canvis abans de guardar els canvis.

Fig. 35 Vista per a canviar d'aparença

Veure comentaris

Un usuari de tipus establiment serà necessari que pugui visualitzar tal com s'ha explicat anteriorment. A la part superior de la vista apareixeran els dos paràmetres especificats com a importants en l'entrevista als usuaris de tipus client que són la valoració de la qualitat i la quantitat. A més es mostrarà una llista de comentaris ordenats de més recents a més antics.

Fig. 36 Vista de veure comentaris

Verificació dels formularis

Per a informar els usuaris sobre si un camp d'un formulari no és correcte o que és obligatori introduir alguna dada s'utilitzarà un text de color vermell que apareixerà davall del camp en qüestió de la següent manera:

Fig. 37 Missatge de verificació d'un camp d'un formulari

Gràcies a la utilització d'una llibreria específica per a la verificació de formularis aquest missatge apareixerà automàticament a mesura que l'usuari va omplint cada camp del formulari i en el moment en el qual la informació del camp sigui correcta aquest missatge desapareixerà automàticament. D'aquesta manera, l'usuari no haurà d'esperar a confirmar el formulari per a saber que un camp no compleix alguna de les regles de verificació.

2.2.6. Tests amb usuaris

2.2.6.1. Plantejament

Per a poder avaluar correctament cada una de les funcionalitats s'estableix un test d'usuari per a cada perfil d'usuari:

Tasques per als usuaris de tipus client:
- Registrar-se amb correu i contrasenya
- Iniciar sessió
- Cerca un establiment amb el mapa
- Cerca un establiment amb la llista
- Filtra els resultats
- Marca un establiment com a preferit
- Compra un article
- Veure últimes comandes
- Avalua l'establiment
- Crea un rebost
- Afegeix un article al rebost manualment
- Afegeix un article al rebost amb una fotografia
- Edita un article del rebost
- Canviar informació de perfil
- Canviar aparença
- Sortir de la sessió

Tasques per als usuaris de tipus establiment:
- Registrar-se
- Iniciar sessió
- Crea una oferta
- Edita una oferta
- Veure les comandes
- Crea un rebost
- Afegeix un article al rebost manualment
- Afegeix un article al rebost amb una fotografia
- Edita un article del rebost
- Canviar informació de perfil
- Canviar aparença
- Veure comentaris
- Sortir de la sessió

Un cop acabat el test es faran les següents preguntes sigui un usuari de tipus client o tipus establiment
- Trobes que hi ha suficient informació a *l'onboarding*?
- Les icones utilitzades en el prototip trobes que transmeten correctament la informació suficient sobre l'acció que volen transmetre?
- Trobes complexa la navegació entre les diferents funcionalitats?

Finalment, es preguntarà a l'usuari sobre suggeriments per a millorar el disseny de cara a la implementació.

2.2.6.2. Conclusions clients

L'usuari no troba cap mena de dificultat per a registrar-se ni per a iniciar sessió, ni per a cercar un establiment a partir del mapa. Tot i això, dona algunes indicacions sobre la informació que veu necessària en els components de tipus targeta dels establiments. Tampoc té cap problema per a filtrar els resultats, però presenta dificultats per a trobar el botó dels preferits. Els reptes més significatius que acaba trobant l'usuari és en el moment de trobar les comandes fetes i com avaluar un establiment. Ja que espera trobar aquesta opció en l'apartat de comandes. L'usuari acaba recomanant canviar el fet d'avaluar establiments per avaluar les comandes dels establiments. Finalment, per la resta de tasques l'usuari no troba cap dificultat important.

Preguntes

- Trobes que hi ha suficient informació a *l'onboarding*?

L'usuari transmet que la informació escrita en *l'onboarding* és suficient tot i que les combinaria amb captures de l'aplicació, per a aconseguir l'atenció de l'usuari. Si l'usuari es troba massa text molt possiblement no llegirà totes les instruccions.

- Les icones utilitzades en el prototip trobes que transmeten correctament la informació suficient sobre l'acció que volen transmetre?

Sí, les icones les troba molt apropiades i defineixen correctament la funció que duen a terme.

- Trobes complexa la navegació entre les diferents funcionalitats?

L'usuari també comunica que mitjançant el prototip és complicat trobar l'apartat per veure les últimes comandes. Suggereix de canviar el nom de l'última pestanya de "Config" amb la icona dels engranatges per "Compte" amb la icona de la silueta d'una persona.

Suggeriments

1. Ja que un mateix usuari pot fer dues comandes en un mateix establiment i les dues comandes poden tenir una valoració diferent, l'usuari suggereix que l'avaluació es faci sobre la comanda i no directament sobre l'establiment.
2. L'usuari suggereix canviar de lloc la icona de preferits damunt del preu per a millorar la seva visibilitat.

2.2.6.3. Conclusions establiments

L'usuari diferencia correctament el registre dels clients amb els dels establiments i no mostra cap dificultat per a iniciar sessió. Un cop oberta la sessió té alguna dificultat per a crear una oferta i editar-la mitjançant la icona del llapis. Quan l'usuari arriba la tasca de crear un rebost pregunta la finalitat de la funció i un cop entesa, no troba cap mena de dificultat per a completar-la. A més a més, expressa que seria necessari una mica més d'ajuda en aquest apartat que expliqués la funcionalitat de manera més clara i directa. Com a consells finals, l'usuari puntualitza que faltaria alguna manera d'expressar la quantitat restant d'un element del rebost. Per a la resta de tasques proposades, l'usuari no mostra cap mena de dificultat per a completar-les.

Preguntes

- Trobes que hi ha suficient informació a *l'onboarding*?

L'usuari transmet que hi ha massa informació escrita i que seria millor afegir alguna captura o imatge.

- Les icones utilitzades en el prototip trobes que transmeten correctament la informació suficient sobre l'acció que volen transmetre?

Sí, explica que les troba apropiades un cop prem el botó i veu l'acció del botó. És per això que recomana afegir text en alguns botons per a clarificar l'acció.

- Trobes complexa la navegació entre les diferents funcionalitats?

L'usuari explica que la navegació generalment no és complexa, tot i que troba una mica confosa la navegació per la pestanya de rebosts.

Suggeriments

1. L'usuari indica que per a aportar més informació afegiria icones informatives, per exemple al costat del títol de rebost que quan es cliqués a sobre donés informació sobre la funcionalitat.
2. L'usuari indica que troba necessari alguna manera d'indicar la quantitat restant i que sigui coherent amb el tipus d'aliment que representa, com per exemple, quilograms, grams, litres i unitats.
3. Per possibles actualitzacions futures l'usuari suggereix desenvolupar algun tipus d'integració de l'API amb els TPV dels comerços per anar actualitzant la quantitat disponible dels elements del rebost d'un establiment a mesura que es cobren els clients a les caixes. Tot i que és un molt bon suggeriment es descarta per la gran càrrega de treball que suposa.

Les taules dels inputs positius i els inputs negatius de cada test d'usuari es pot trobar a l'annex 7.3.

2.2.6.4. Millores a dur a terme

1. Afegir consells (*tooltips*) per a transmetre encara més informació polsant sobre icones d'informació.
2. Afegir text en els botons de les icones per a clarificar la seva acció.
3. Canviar posició de la icona de preferits.
4. Afegir captures amb text explicatiu a *l'onboarding*
5. Cal afegir l'opció d'indicar la quantitat als elements del rebost amb la unitat desitjada.
6. Poder avaluar les comandes en comptes dels establiments, ja que com diu l'usuari de tipus client estaria millor poder avaluar les comandes que els establiments, pel fet que dues comandes d'un mateix establiment poden tenir dues avaluacions diferents. Per a poder adaptar el disseny es mouria el formulari d'avaluar l'establiment a la pàgina on l'usuari de tipus client veuria l'extracte de la comanda i l'usuari de tipus establiment veuria cada una de les valoracions en la seva vista de la comanda. A més a més, perquè l'usuari de tipus establiment pugui veure la mitjana de les seves avaluacions ho podrà veure directament des de la pàgina de comandes, fent més visible aquesta funcionalitat.

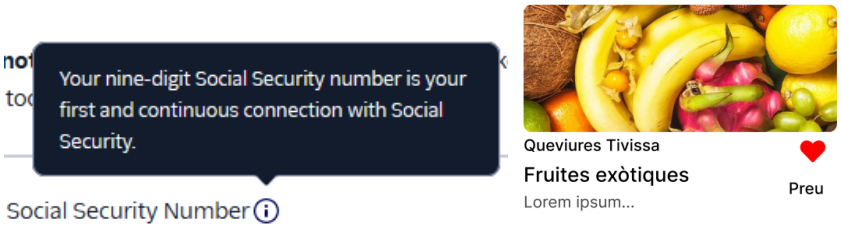


Fig. 38 Exemple de *tooltip* i vista de la targeta amb la icona de preferits canviada

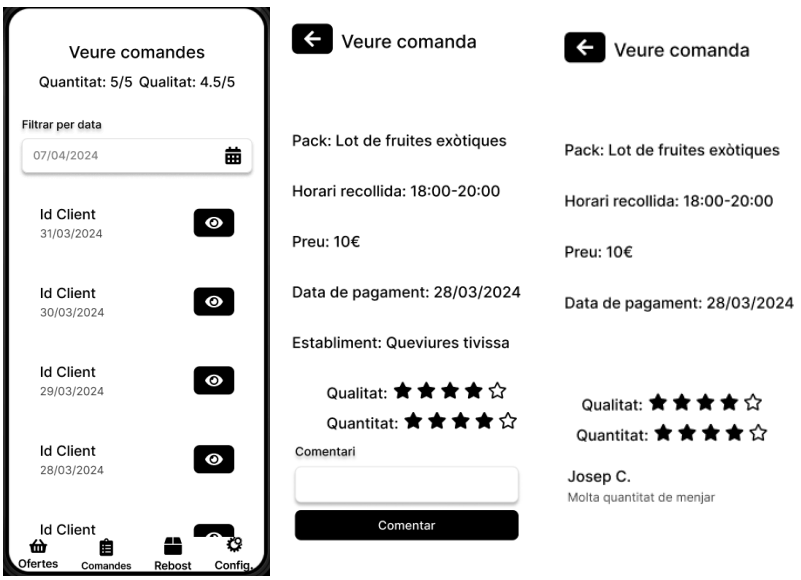


Fig. 39 Vista del canvi de veure comanda pel client i veure comanda per l'establiment

2.3. Disseny tècnic

2.3.1. Casos d'ús

Per a obtenir una visió clara de totes les accions disponibles que tindran els diferents tipus d'usuari, s'aplica la tècnica de casos d'ús. En el següent diagrama, es poden observar els dos tipus d'usuari presents a l'aplicació i les diferents tasques que podrà dur a terme.

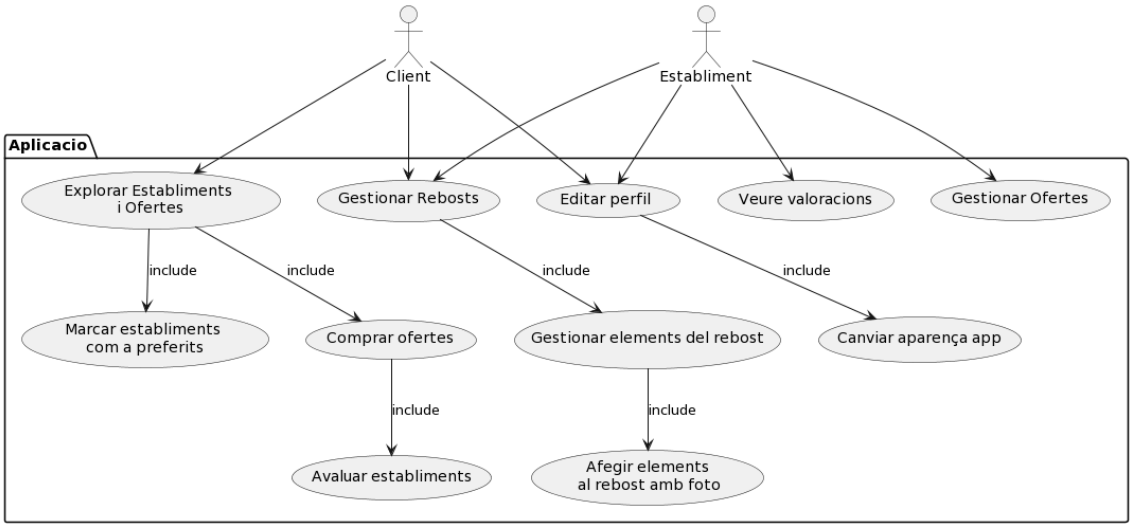


Fig. 40 Diagrama de casos d'ús

Les definicions de cada un dels casos d'ús es pot trobar a l'annex 7.4.

2.3.2. Disseny de la base de dades

Per a poder modelar tota la informació necessària per al funcionament de l'app caldrà modelar una base de dades *NoSQL* amb la següent estructura:

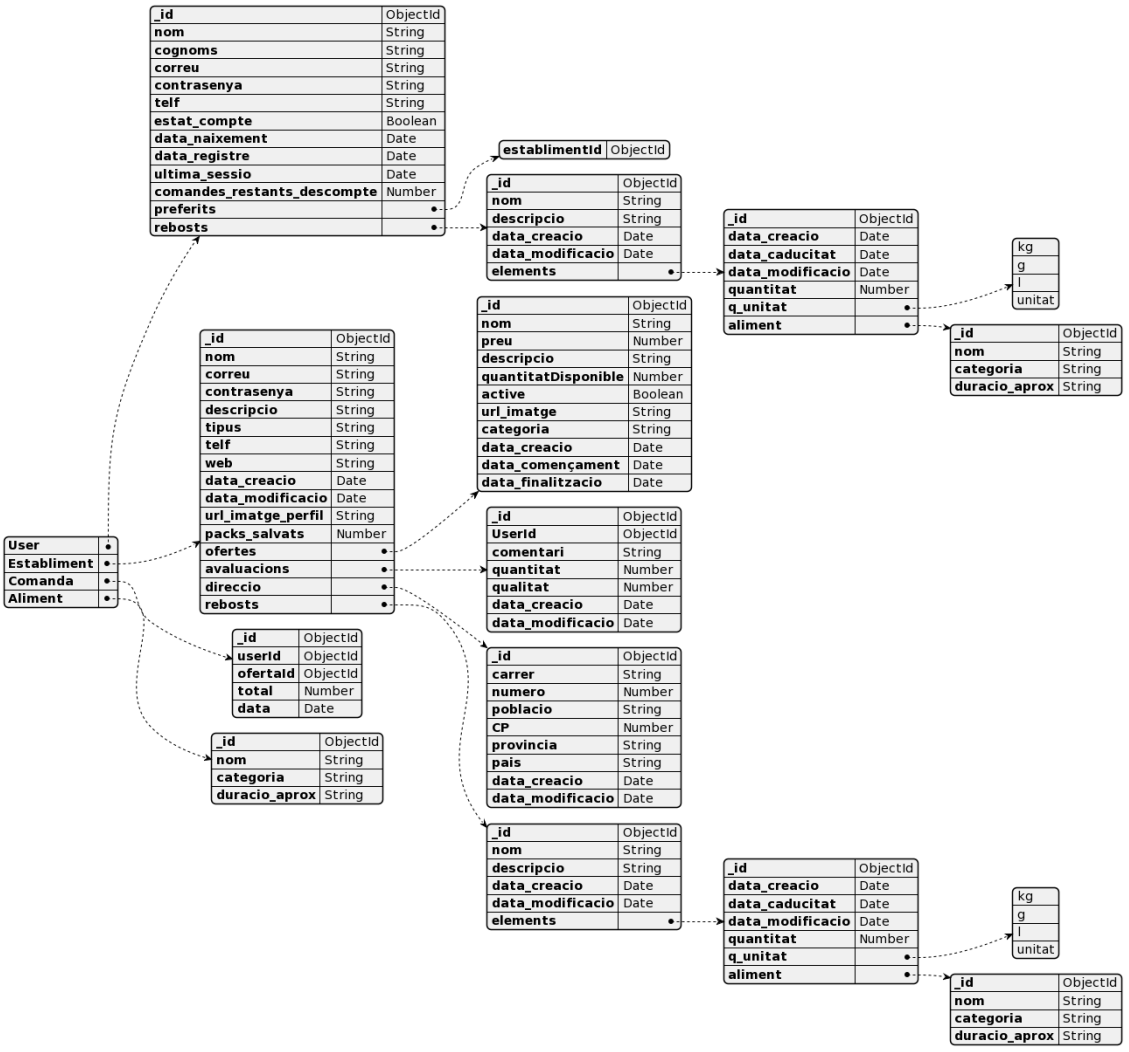


Fig. 41 Model de la base de dades

Com que s'utilitza una base de dades no relacional la informació no s'emmagatzemarà en taules sinó en documents i documents encastats en format **BSON**.

- **Usuari**: En aquest document s'emmagatzemarà tota la informació referent a l'usuari. En el diagrama es pot veure com aquest document té dos atributs que corresponen a dos documents encastats:
  - **Preferits**: En aquest document s'emmagatzemarà una llista de les ID de tots els establiments preferits de l'usuari en qüestió.
  - **Rebosts**: En aquest document s'emmagatzemaran els rebosts de cada un dels usuaris amb un nom i una petita descripció, dins

d'aquests rebosts contindran una llista de documents encastats que representaran els elements del rebost:
    - **Elements**: En aquest document s'emmagatzemaran tots aliments que conté el rebost, així com la informació relacionada, com el nom, dates de compra i caducitat…
- **Establiment**: En aquest document s'emmagatzemarà tota la informació corresponent als establiments. Aquest document contindrà els següents documents:
    - **Rebosts**: Aquest document tindrà la mateixa estructura que el document dels rebosts dels usuaris
    - **Direcció**: Aquest document contindrà els camps necessaris per a representar qualsevol direcció de forma específica, com el nom del carrer, el número, la població, el CP la província i el país.
    - **Oferta**: Aquest document contindrà tota la informació referent a les ofertes creades per l'establiment en qüestió.
    - **Avaluació**: En aquest document s'emmagatzemaran les avaluacions de cada un dels usuaris amb la seva ID.
- **Comanda**: En aquest document s'emmagatzemarà la informació referent a la comanda, l'usuari que la du a terme i l'oferta que compra.
- **Aliment**: Aquest document no podrà ser editat per cap establiment ni cap usuari de l'aplicació. Aquest document contindrà informació dels aliments disponibles en la base de coneixement de *l'OCR*. Quan l'usuari escanegi un tiquet i s'extreguin els caràcters, es cercaran les paraules clau en aquesta taula. Per tant, s'emmagatzemarà informació com el nom de l'aliment, la categoria (Verdura, Fruita, Carn, Peix, Làctics i Fleca) i una durada aproximada abans que arribi la data de caducitat. Aquesta duració serà la proposada quan s'escanegi un tiquet mitjançant la càmera.

2.3.3. Disseny de l'arquitectura global del sistema

L'arquitectura global del sistema es compondrà dels següents elements, segons l'entorn al qual pertanyen, és a dir si són elements els quals el seu objectiu és servir l'aplicació es trobaran en l'entorn de front-end i si són elements el qual el seu objectiu és servir informació a l'aplicació es trobaran en l'entorn de back-end.

En l'entorn de Front-End trobarem:

- L'aplicació compilada per a Android i iOS que permetrà executar l'aplicació en els dispositius mòbils perquè els usuaris puguin accedir a la informació del back-end.
- Servidor web amb nginx que servirà l'aplicació en línia perquè qualsevol usuari amb un navegador compatible pugui utilitzar l'app des de qualsevol dispositiu. Ja que la càrrega del servidor web serà mínima per a servir els fitxers que compondran l'app, es decideix que se servirà l'app en línia en la mateixa màquina física que el servidor de l'API.

En l'entorn de Back-End trobarem

- Com que durant la primera etapa de vida de l'aplicació no s'espera una gran afluència d'usuaris, s'utilitzarà únicament una màquina física que actuarà com a servidor principal. Excepte per a dur a terme les còpies de seguretat de les dades, les quals es faran en un servidor dedicat únicament a emmagatzemar aquestes còpies.
- En aquest servidor s'implementarà un Swarm de contenidors Docker els quals estaran orquestrats mitjançant el programari Portainer.io. El fet d'implementar un clúster de contenidors Docker, permetrà executar diferents contenidors en una mateixa màquina, però que s'executen de manera aïllada, a la vegada que es comuniquen de manera de segura entre ells mitjançant una xarxa interna que ofereix Docker. D'aquesta manera únicament quedaran descoberts a l'exterior, els punts d'entrada de l'API i del servidor web, mantenint segura la base de dades. A més a més, l'orquestrador Portainer detectarà les possibles fallades o caigudes que poden tenir els contenidors i els replicaran de nou perquè pugui continuar el servei. Addicionalment, usar un Swarm de Docker, permetrà afegir més màquines físiques al Swarm per augmentar la capacitat de computació del sistema i afegir més servidors per a satisfer la demanda.
    - Un contenidor del Swarm estarà compost per una imatge oficial de MongoDB v.7 del repositori de Docker. A partir d'aquesta imatge s'especificarà un fitxer YAML per a configurar el port del servidor, la carpeta on s'emmagatzemen els fitxers binaris de la base de dades... Amb aquest fitxer de configuració, es generarà una imatge que es carregarà al clúster mitjançant Portainer.
    - Un altre contenidor del clúster estarà compost per una imatge oficial amb Node V20. En el fitxer de configuració d'aquesta imatge únicament caldrà definir el repositori d'on ha de

descarregar el codi i el fitxer index.js des d'on Node arrancarà l'API del servidor.

- Un tercer contenidor del clúster estarà compost per una imatge oficial de nginx. Per aquest contenidor es proporcionarà un fitxer de configuració de nginx que configurarà el servidor perquè actuï com a *Proxy* invers o equilibrador de càrrega. També es configurarà aquest contenidor perquè serveixi els fitxers estàtics de l'aplicació front-end i pugui ser accessible des de qualsevol navegador.

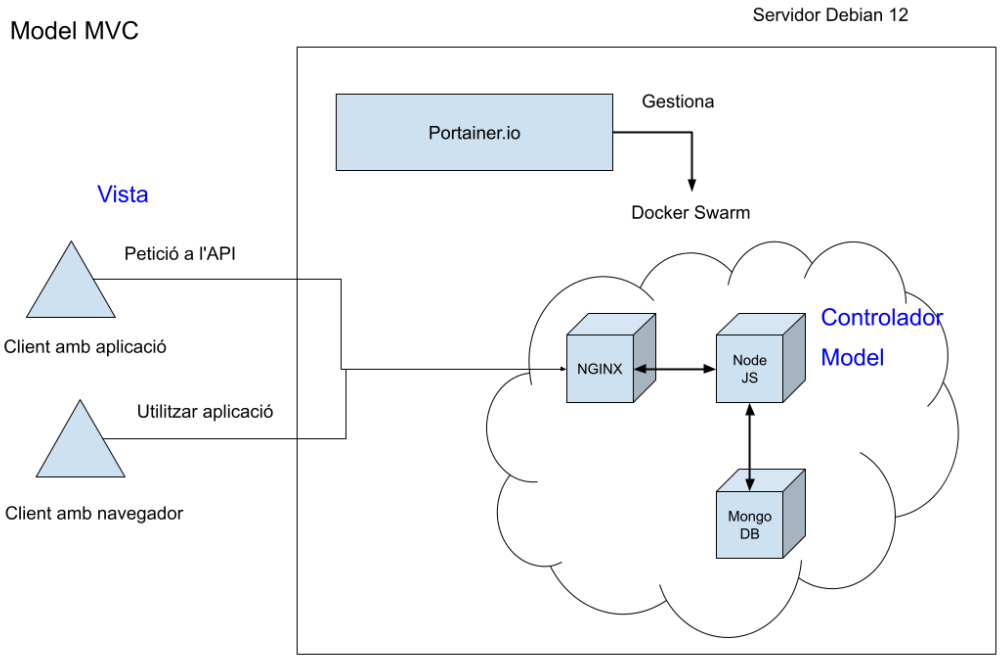


Fig. 42 Arquitectura del servidor

2.3.4. Diagrama de classes

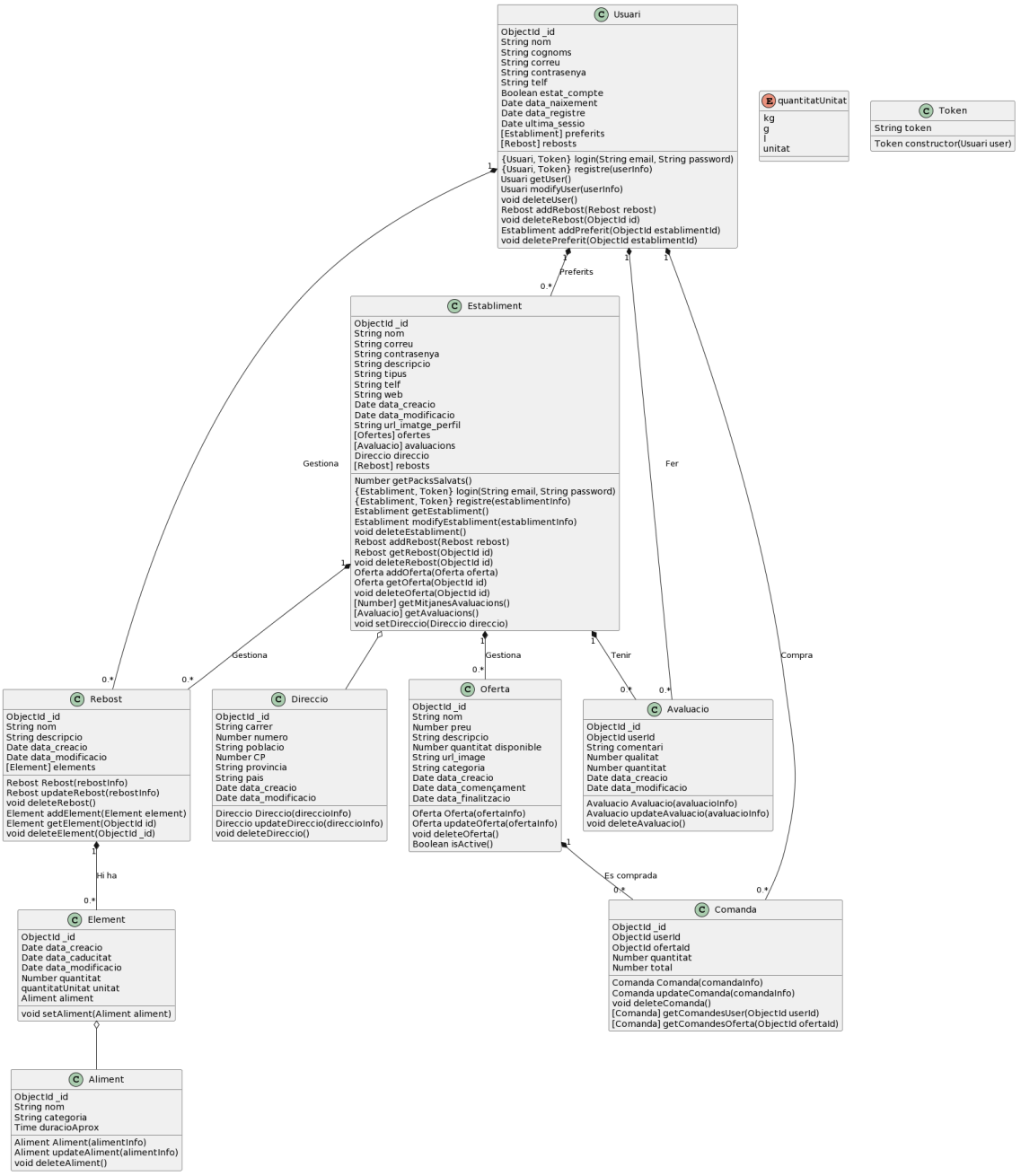


Fig. 43 Diagrama de classes UML

3. Implementació

3.1 Tecnologies utilitzades

Per a poder complir en tots els objectius del projecte han estat necessàries 4 tecnologies principals les quals són:

Firebase: Firebase és una tecnologia de Google la qual ofereix una plataforma completa per al desenvolupament d'aplicacions mòbils i web, inclou servei de notificacions, d'autenticació d'usuaris, bases de dades... En aquest cas s'han utilitzat els serveis de Firebase per a implementar les notificacions dels avisos i l'autenticació dels usuaris mitjançant un compte de Google.

Google Cloud: Google Cloud, és una plataforma de computació en el núvol de Google, que ofereix accés a diferents opcions de virtualització de màquines i infraestructura necessària per al desplegament d'una aplicació i tots els serveis dels quals depèn. En el cas d'aquest projecte s'ha utilitzat la solució Compute Engine, la qual permet tenir una màquina virtualitzada amb l'última versió de Debian. D'aquesta manera es pot muntar un servidor de forma gratuïta (gràcies als 300 $ de crèdit proporcionat per Google per provar Google Cloud) per fer proves durant la implementació i un altre servidor per al desplegament de l'aplicació final.

Cloudflare: Després de fer la compra d'un domini per allotjar l'aplicació i l'API, es tria Cloudflare com a proveïdor DNS. Cloudflare, a part de permetre gestionar els subdominis i un certificat HTTPS per al domini, permet afegir seguretat d'accés a l'app i a l'API. Una d'aquestes mesures de seguretat és definir IP en una llista negra per bloquejar-ne l'accés. També permet limitar l'accés temporalment dels usuaris en cas d'atac de denegació de servei, per a mitigar l'atac.

Node: Aquesta tecnologia és un entorn d'execució de JavaScript amb V8 com a motor d'execució de JavaScript utilitzat principalment pel navegador web Chrome i conegut per la seva rapidesa i eficiència en l'execució de codi JavaScript. Node.js executa aquest V8 en la banda del servidor, podent oferir aplicacions escalables, flexibles i amb un alt rendiment. Per l'altra banda, Node.js permet organitzar el codi en petites parts modulars, fet que ajuda a la reutilització de codi. A més a més, inclou un gestor de paquets anomenat NPM, que permet gestionar i instal·lar llibreries de tercers i reutilitzar aquestes funcionalitats ja implementades per altres desenvolupadors.

Les llibreries principals de NPM utilitzades per al funcionament bàsic del projecte són:

Ionic: Ionic.js es tracta d'una *framework* per a poder programar aplicacions per a diferents plataformes a partir d'un únic codi font. Ionic permet utilitzar tant JavaScript com *TypeScrypt* per al codi font. A més a més, aquesta llibreria proporciona elements UI que no ha fet falta programar, com la vista de les

pestanyes, els *Modals*, les alertes, els formularis... Per l'altra banda Ionic s'integra totalment amb Capacitor, per a poder generar aplicacions totalment compatibles amb Android i iOS, a la vegada que implementes una aplicació per a web.

Capacitor: Capacitor.js és una llibreria la qual permet exportar codi de JavaScript a codi natiu d'altres plataformes, com Android i iOS. A més a més, aquesta llibreria proporciona funcionalitats que faciliten l'accés als sensors i serveis de la plataforma, com el sensor GPS, el servei de notificacions i la càmera. És a dir, proporciona una capa addicional que permet executar codi JavaScript en qualsevol plataforma suportada.

Vue: Vue.js és una *framework* de desenvolupament d'aplicacions SPA (*Single Page Application*) que presenta els següents avantatges que el fan una opció a tenir en compte a l'hora d'haver de desenvolupar qualsevol mena d'aplicació.

- **Facilitat d'ús**: Els *Single File Components* de Vue.js permeten desenvolupar aplicacions web sense haver d'aprendre un nou llenguatge. A més, permeten encapsular l'estructura amb HTML, la lògica amb JavaScript i l'aparença amb CSS d'un mateix component en un sol arxiu (.vue). Permetent així aïllar les diferents parts i responsabilitats en diferents components. A més ofereix funcionalitats extres com directives HTML i declaració de variables JavaScript reactives per a aconseguir una interfície d'usuari dinàmica, de forma còmoda.
- **Reactivitat**: Vue.js ofereix una reactivitat excel·lent, la qual cosa significa que les vistes s'actualitzen automàticament quan les dades canvien, sense necessitat d'intervenció explícita.
- **Components**: Vue.js fomenta l'ús de components reutilitzables, el que facilita la creació d'interfícies d'usuari complexes i escalables. En el cas d'aquest projecte ha permès definir components reutilitzables, com per exemple, tots els formularis utilitzats per a crear un recurs han estat reutilitzats per la vista que permet editar el recurs. A més a més, també s'han reutilitzat components més petits, però no menys complexos, com els botons per a marcar un establiment com a preferit.
- **Integració amb altres llibreries**: Utilitzar Vue també implica poder utilitzar llibreries com VueUse (llibreria d'utilitaris), Vuelidate (per a la validació de formularis), Pinia (per al control d'estats)… Que aporten encara més millores i facilitats en el desenvolupament d'aplicacions amb aquest *framework*.

Express: Express és una *framework* de Node.js que permet desenvolupar de manera ràpida i còmoda un servidor HTTP, el qual ofereix una estructura lleugera i flexible per a la creació d'aplicacions web i API. Amb Express, es poden definir rutes i controladors, gestionar peticions i respostes, utilitzar *middlewares* per a la manipulació de peticions i respostes, i gestionar errors. A més, Express és compatible amb una gran varietat de i llibreries, la qual cosa permet ampliar les seves funcionalitats segons les necessitats específiques del projecte. Per l'altra banda, Express també facilita la integració amb bases de

dades com MongoDB, MySQL, PostgreSQL, entre d'altres, i suporta la creació de sessions, la gestió d'arxius estàtics, la gestió de galetes, i moltes altres funcionalitats que són essencials per al desenvolupament d'aplicacions web modernes.

Vue-router: Vue Router és un complement per a Vue, per a definir quins components es veuran segons l'URL en la qual es troba l'usuari. A més a més, Vue Router proporciona *Route Guards*. Les *Route Guards* són regles que modifiquen el comportament del rúter de forma clara i centralitzada, sense haver de repetir redireccions entre els components. Per exemple, en el cas d'aquest projecte com que un usuari que no ha iniciat sessió no pot accedir a l'aplicació, s'ha definit una *Guard* al rúter que verifica una variable al controlador d'estats (*store*) d'inici de sessió que conté el JWT. Si l'usuari que accedeix a l'aplicació no té inicialitzada aquesta variable, el rúter el reencaminarà automàticament a la vista d'inici de sessió i no podrà utilitzar l'aplicació.

Mongoose: És una llibreria de modelatge d'objectes de MongoDB per a Node.js. Aquesta llibreria simplifica la interacció amb MongoDB des de Node.js proporcionant un conjunt d'eines i una interfície coherent per definir esquemes, validar dades, realitzar consultes, i gestionar les transaccions. Mongoose proporciona una interfície molt potent per a realitzar consultes a la base de dades. Aquesta interfície permet cercar documents basats en diverses condicions, ordenar els resultats, limitar el nombre de documents retornats, agregar informació, calcular-ne de nova a partir de les dades existents...

Dins del projecte, un dels objectius era el de desenvolupar un mètode senzill per introduir els productes comprats a partir d'una fotografia d'un tiquet. Doncs bé, les tecnologies necessàries per a poder assolir aquest objectiu són les següents:

Tesseract.js: Aquesta llibreria és un port per JavaScript del conegut Tesseract-OCR (*Optical Character Recognition*). Aquesta llibreria ofereix funcionalitats per extreure una cadena de caràcters a partir d'una imatge que contingui caràcters. A més a més, aquest port ha estat pensat per a ser executat en navegadors, per la qual cosa ocupa molt poc espai i consumeix pocs recursos a comparació d'altres reconeixedors de caràcters. Per tant, és el candidat perfecte per a integrar-lo en l'aplicació.

Image-js: Per aconseguir els millors resultats en el reconeixement d'imatges per part de *l'OCR* és necessari dur a terme un tractament de les imatges abans de processar-les. Per a poder aconseguir-ho s'utilitza la llibreria image-js que un cop feta la fotografia, image-js aplica filtres per a convertir la imatge en blanc i negre, o aplica filtres per a ressaltar els contorns dels caràcters de les imatges de forma senzilla.

string-similarity-js: També per a millorar encara més els resultats del reconeixement de caràcters s'utilitza aquesta llibreria que proporciona mètodes per a comparar paraules o cadenes de caràcters amb l'algorisme de distància

de Levenshtein. Que amb dues cadenes de text com a punt de partida retorna el quocient de similitud entre 0 i 1.

Per a accelerar el procés de desenvolupament, es reutilitza codi de tercers amb funcions utilitàries que permeten obtenir un codi òptim, ja que són funcions testejades per una gran comunitat i ajuden a mantenir un codi més net i permeten centrar-se en les funcionalitats necessàries.

Vitest: Llibreria de testing integrada amb vite, que permet fer tests unitaris amb la sintaxi de jest. A més a més, proporciona eines per a realitzar simulacions (*mocks*) de mòduls i funcions importades en el component, per a poder dur a terme tests unitaris complexos a tota mena de components.

APIs auxiliars:

Nominatim
Quan un establiment es registra a la plataforma, aquest ingressa al formulari únicament una adreça. Per a poder saber la localització exacta d'una adreça (latitud i longitud) s'utilitza l'API Nominatim, que a part de verificar si una adreça introduïda durant el registre és vàlida o no (i evitar que es creïn establiments falsos), també permet extreure'n les coordenades exactes de forma automàtica que posteriorment s'emmagatzemaran a la base de dades per a cercar establiments propers a partir de coordenades.

https://nominatim.org/release-docs/latest/api/Lookup/

My-ip.io
En el cas que a l'usuari no li funcioni o no activi la localització GPS. L'aplicació utilitza l'API pública de My-ip per a obtenir informació pública de la IP per a obtenir una localització aproximada de l'usuari. Aquesta informació de localització no s'envia ni es monitora des del servidor, per la qual cosa la localització de l'usuari sempre és anònima pel servidor.

https://api.my-ip.io/v2/ip.json

Altres llibreries secundàries, però no menys importants que s'han implementat a l'API són:

Per a emmagatzemar les contrasenyes a la base de dades s'utilitzarà aquesta llibreria que encripta les contrasenyes de manera segura. L'algoritme bcrypt és resistent a atacs de força de bruta i a més aporta funcions senzilles per a encriptar i per a verificar contrasenyes.

La llibreria CORS proporciona una forma còmoda de configurar la política CORS del servidor Express. CORS és un mecanisme de seguretat del navegador que restringeix les sol·licituds HTTP realitzades des d'orígens diferents del del servidor. Amb aquesta llibreria es poden configurar de manera senzilla les regles per permetre o denegar sol·licituds CORS des de dominis

específics, mètodes HTTP i altres paràmetres, facilitant així la comunicació segura entre el client i el servidor.

A part de validar les dades introduïdes a l'aplicació, és important validar les dades que arriben a les sol·licituds de l'API per si es produeix algun atac i s'envien peticions a l'API amb dades errònies de forma intencionada per a produir alguna injecció de codi a la base de dades... Per a evitar-ho s'utilitza express-validator per a sanejar el cos i els paràmetres de les peticions que arriben a Express.

Per a poder enviar notificacions als dispositius dels usuaris és necessari utilitzar la llibreria firebase-admin, que permet enviar notificacions personalitzades als dispositius escollits. A més a més, per a poder programar l'enviament diari de notificacions des de l'aplicació s'ha utilitzat node-cron.

La llibreria google-auth-library, és necessària per a poder validar els tokens OAuth2 generats pels botons de Google de l'aplicació i que s'envia a través d'una crida a l'API per a poder iniciar sessió.

Helmet.js, Morgan i Multer, són 3 llibreries que prorporcionen funcionalitats bàsiques de seguretat, gestió de registres (*logs*) i gestió de fitxers per a Express mitjançant *middlewares*.

Ja que no s'ha pogut implementar un sistema d'inici de sessió amb Passport.js, es decideix implantar un sistema d'inici de sessió per JWT des de 0. Aquesta llibreria, permet generar JWT en el servidor, per a proporcionar-lo al client i el pugui utilitzar per a accedir a recursos privats d'Express. A més, aquesta llibreria permet verificar els tokens de manera ràpida i simple implementant un petit *middleware* en les rutes privades d'Express.

3.2. Reptes

En el següent apartat es detallaran els reptes trobats durant la implantació i les solucions o decisions adoptades per a solucionar o mitigar aquests problemes.

3.2.1. Inici de sessió

Durant la fase de disseny es va proposar Passport.js com a gestor d'autenticació dels usuaris. Tot i això, després de bastants intents no s'aconsegueixen avenços, ja que Passport.js està pensat per aplicacions renderitzades en el servidor i no s'ha aconseguit implementar en aquest projecte. Pel fet que l'aplicació es renderitza en el client i el servidor únicament proporciona les dades i recursos necessaris per al seu funcionament. Per tant, s'ha decidit fer un sistema d'inici de sessió per correu i contrasenya fet de 0 juntament amb un sistema d'autenticació de JWT. En conseqüència, es dissenya de nou el sistema d'autenticació amb correu i contrasenya que funcionarà de la següent manera, seguint les millors pràctiques segons les referències[12 *Carlos Azaustre. 2015*]:

1. Sigui usuari del tipus client o tipus establiment haurà de registrar-se mitjançant un dels dos formularis disponibles des de la pàgina d'inici de sessió.
2. Quan el servidor rebi la petició de registre d'algun formulari valida les dades, les emmagatzema i just abans d'emmagatzemar la contrasenya, es dispara un disparador (*trigger*) de Mongoose que automàticament encripta la contrasenya amb l'algorisme "bcrypt" i l'emmagatzema encriptada a la base de dades.
3. Un cop registrat, quan l'usuari envia la petició d'inici de sessió amb correu i contrasenya, el servidor cerca primer un usuari al document dels clients amb el correu especificat, si no existeix, cerca el correu al document dels establiments. En cas, que no trobi cap correu informarà de l'error a l'usuari.
4. En el cas que el servidor trobi algun usuari de tipus client o de tipus establiment a la base de dades, extraurà, la contrasenya encriptada i la ID de l'usuari. A continuació, mitjançant la llibreria "bcrypt", compararà la contrasenya proporcionada amb la contrasenya encriptada a la base de dades. En cas que no coincideixi la comparació, s'enviarà un missatge notificant a l'usuari.
5. En el cas que la comparació sigui satisfactòria, el servidor, mitjançant la llibreria JWT token crea un token amb informació de l'usuari, com el seu ID, el tipus d'usuari i la marca de temps de quan finalitza la validesa del token. A continuació el servidor firma aquesta informació amb una clau secreta que únicament es troba en les variables d'entorn del servidor. A continuació, el servidor envia el token com a resposta al client.
6. Si el client rep una resposta satisfactòria amb estatus 200 i el token com a resposta, l'emmagatzema en un controlador d'estats (*store*) de Pinia. Gràcies a utilitzar la *store* es pot saber des de qualsevol vista si l'usuari ha iniciat sessió i quin és el token proporcionat pel servidor. Si es vol visualitzar la informació que conté el token, es pot fer des de la pàgina següent, per tant, és important no introduir informació com la

contrasenya. Si volem, des d'aquesta pàgina es podria generar un token modificant el contingut, tot i això, per a generar un token que el servidor reconegui, caldria generar-lo amb la clau secreta que únicament coneix el servidor.

7. Un cop iniciada la sessió i el client ha guardat el token a la *store*, el client ja pot fer peticions que necessiten autenticació, com per exemple, obtenir tots els rebosts, afegir ofertes... La capa de serveis APIService creada per a fer les peticions, automàticament utilitza el token emmagatzemat en la *store* per a incloure el token a la capçalera de les peticions.
8. Quan les peticions arriben al servidor, aquestes executen funcions especificades en el rúter. En aquestes funcions, es pot especificar una per a verificar que el token en la capçalera de la petició és vàlid. D'aquesta manera, quan en una ruta existeixi la funció isAuth, es verificarà el token i s'extraurà informació rellevant com la ID de l'usuari i el tipus d'usuari que serviran per a les funcions següents, on es retornaran els recursos demanats per l'usuari. En aquest pas també es verificarà si el token es troba caducat o no.

3.2.2. Organització del projecte del servidor i reutilització de codi

Per a poder gestionar correctament la gran quantitat de peticions a l'API necessàries per a poder gestionar tots els recursos disponibles, s'ha de dur a terme una organització exhaustiva del projecte per poder separar les obligacions de cada una de les capes del servidor en diferents carpetes i fitxers. Per a organitzar aquest projecte s'ha decidit crear les següents carpetes, seguint les millors pràctiques d'implementació d'una API amb Express [13 *Chris, Kolade. 2021*] [14 *Express. 2017*]:

- **Controladors**: De cada recurs que estarà disponible en l'API es crearà un fitxer amb les funcions bàsiques HTTP que defineixen els verbs (accions) bàsics del protocol HTTP. Addicionalment, també es creen els recursos necessaris, com per exemple, POST /search, per a poder dur a terme cerques amb filtres a la base de dades. S'ha de tenir en compte que aquests fitxers no accedeixen a la base de dades directament, d'això se n'encarreguen els serveis, d'aquesta manera es separen les responsabilitats en diferents funcions.
    - GetAll (GET /recurs)
    - GetOne (GET /recurs/id)
    - Create (POST /recurs)
    - Update (PUT /recurs/id)
    - Delete (DELETE /recurs/id)

Fig. 44 Vista de la carpeta controladors i una funció controladora de l'API que crida a un servei de la BD

- **Models**: En aquesta carpeta s'emmagatzemen els models dels documents de la base de dades definits amb Mongoose. Aquests models permetran fer consultes i canvis a la base de dades. Aquests objectes únicament seran accedits des dels fitxers de la carpeta Serveis. D'aquesta manera, si alguna vegada canvia el motor de la base de dades, únicament caldrà adaptar les funcions de la capa de serveis i proporcionar els mateixos resultats i funcions a la capa de controladors. Facilitant així possibles canvis futurs i fent el projecte escalable.

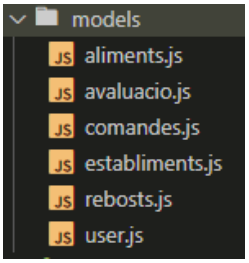


Fig. 45 Vista de la carpeta models i vista del model de l'usuari de tipus client

- **Middlewares**: En aquesta carpeta es troben totes les funcions que s'executen abans que s'executi qualsevol dels controladors. En aquesta carpeta es pot trobar el *middleware* encarregat de verificar el JWT, o el *middleware* d'emmagatzemar les imatges organitzades en carpetes i noms predefinits... També es troba el *middleware* que valida les peticions POST i PUT de l'API.

- **Serveis**: En aquesta carpeta es troba tota la lògica del servidor, en aquests arxius es troben les funcions que realitzen les consultes a la base de dades, el servei de generació de tokens i el servei de les notificacions.

Fig. 46 Vista de la carpeta de serveis i la funció que es crida des del controlador anterior

- **Router**: En aquesta carpeta es troba el fitxer principal de la definició de les rutes del servidor. En aquest fitxer s'indiquen les rutes a les quals el servidor HTTP ha de respondre i quins *middlewares* i controladors han d'executar-se en cada ruta.

```
router
  .get('/rebosts/:rebostId/elements', isAuth, elementsCtrl.getAllElements)
```

Fig. 47 Vista de la definició d'una ruta amb el *middleware* d'autenticació i el controlador.

3.2.3. Organització del projecte de l'aplicació

De la mateixa manera que el projecte del servidor, el projecte del client també requereix una bona organització a causa de la gran quantitat de vistes i components que cal mostrar. Per aquest motiu durant el transcurs de la implementació s'ha organitzat el projecte amb la següent estructura, seguint les millors pràctiques basant-me en les següents referències [15 *Zanini, Antonello. 2023*]:

- **APIService**: Carpeta únicament dedicada a la definició de les operacions amb la llibreria Axios, on es troben totes les peticions necessàries perquè l'aplicació es comuniqui correctament amb el servidor.

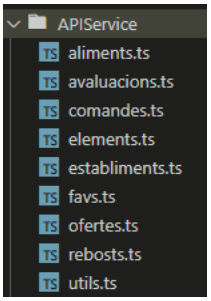


Fig. 48 Vista de la carpeta APIService

A més a més, en aquesta **APIService**, s'ha creat un *wrapper* anomenat *AxiosWrapper*. Un *wrapper*, també és una funció que envolta una altra funció i s'executa abans i després de l'altra funció passada com a paràmetre. Aquest *wrapper* encapsularà totes les peticions que es facin a l'API i mostrarà els missatges d'error corresponents en cas que faci falta i executarà una funció de retorn (*callback*), després de dur a terme la petició, independentment de si hi ha hagut un error o no.

```
export const axiosWrapper = (fn: Function,cb: CallbackFunction,...options: any) => {
  fn(...options)
    .then((response: any) => {
      cb(null, response.data);
    })
    .catch(async (error: any) => {
      if (error.response) {
        // La request s'ha fet pero s'ha obtingut un status diferent a 2xx
        console.log(error.response.data);
        console.log(error.response.status);
        console.log(error.response.headers);
        cb({
          status: error.response.status,
          message: error.response.data,
        });
        if (error.response.status == 401 || error.response.status == 400) {
          let alert = await createErrorAlert(
            error.response.data.message,
            error.response.status
          );
          alert.present();
        }
        if (error.response.status == 500) {
          console.log(
            `Error status: ${error.response.status} Error message: ${error.response.data.message}`
          );
```

Fig. 49 Definició del *wrapper*

Sembla molt codi innecessari, però realment el que s'està aconseguint és assegurar-se que totes les peticions a l'API s'executen correctament o que en cas contrari mostrin automàticament el missatge d'error amb l'error rebut. I a més permet definir les operacions de manera molt fàcil.

```
import { instance, getHeaders, axiosWrapper } from './utils';
import {CallbackFunction} from '../types'
//Rebosts

export const getRebosts = (cb: CallbackFunction) =>
  axiosWrapper(instance.get, cb, '/rebosts', getHeaders());
export const getRebost = (id_rebost: any, cb: CallbackFunction) =>
  axiosWrapper(instance.get, cb, `/rebosts/${id_rebost}`, getHeaders());
export const crearRebost = (data: any, cb: CallbackFunction) =>
  axiosWrapper(instance.post, cb, '/rebosts', data, getHeaders());
```

Fig. 50 Definició de les operacions possibles a realitzar a l'API sobre el rebost

I en la següent captura es pot observar amb la facilitat en la qual es pot fer una crida a l'API, on únicament cal especificar la funció de retorn per a saber com tractar les dades. Ja que si dona error, el *wrapper* definit anteriorment mostrarà els avisos automàticament.

```
const fillRebosts = async () => {
  let loader = await showLoading("Carregant rebosts")
  loader.present()
  getRebosts((err: any, data: any) => {
    loader.dismiss()
    if (err) return
    rebosts.value = data.rebosts;
  })
}
```

Fig. 51 Crida d'una funció de l'API amb el *wrapper*.

- **Components**: Carpeta dedicada als SFC (*Single File Components*) dels components reutilitzables, en aquesta carpeta es pot trobar el botó de preferits, els components de les targetes dels establiments, les targetes de les ofertes, les targetes dels rebosts, les targetes dels elements del rebost. L'element de *l'onboarding*, el component de comentar, la targeta per a veure les avaluacions a la vista de l'establiment, les targetes de les comandes, el component del missatge d'error quan es valida un formulari, el botó de preferits i dos badges que permeten mostrar l'horari i el tipus d'establiment de forma més còmoda i de forma predeterminada.
- **Composables**: Segons la documentació oficial de Vue, els *composables* són fitxers de codi de Vue que no necessiten una plantilla (*template*) i que són reutilitzables en una gran part del projecte. Els fitxers d'aquesta carpeta són: els *composables* de les alertes i els "loaders", el de la càmera, és a dir, el codi encarregat d'engegar la càmera i processar les fotografies estigui en la plataforma que estigui. En aquesta carpeta també es troba el codi encarregat d'inicialitzar els serveis de les notificacions de Firebase per a la plataforma web i el de Capacitor.
- **Router**: En aquesta carpeta es troba el fitxer principal del rúter. En aquest fitxer es troben les rutes les quals l'aplicació respondrà per a mostrar una vista i el fitxer de la vista que haurà de renderitzar el rúter. En aquest fitxer, s'han configurat les importacions de les vistes amb càrregues mandroses (*lazy imports*). Aquesta tècnica permet carregar únicament les vistes renderitzades en memòria a mesura que l'usuari va visitant les rutes, d'aquesta manera s'augmenta el rendiment general de l'aplicació. A més a més, en aquest fitxer s'especifiquen les Route Guards mencionades anteriorment. Únicament declarant un disparador "beforeEach" de l'objecte rúter, es podrà declarar una funció que s'executarà cada vegada que el rúter canviï de ruta. En conseqüència, es podrà verificar en cada canvi de ruta, si l'usuari està autenticat i si la ruta a la qual es dirigeix està protegida o no...
- **Stores**: En aquesta carpeta es troben els fitxers controladors de l'estat que solucionen diversos problemes de persistència de dades entre els components. Les variables que es declaren dins d'un component de Vue són exclusives d'aquell component i cap dels altres components pot accedir fàcilment a les variables. Les úniques maneres en les quals es poden enviar dades d'un component a un altre és mitjançant atributs (*props*) quan s'invoca un component fill o mitjançant disparadors personalitzats, tot i això, és necessari que el component s'hagi renderitzat per a poder rebre aquest disparador. Existeixen altres

mètodes com *provide* i *inject*, però Pinia.js ofereix una implementació molt simple i s'integra molt bé amb Vue, ja que utilitza les mateixes propietats reactives. En aquesta carpeta es troben els controladors d'estat (*stores*) per a emmagatzemar en local l'estat dels establiments marcats com a favorits, per a emmagatzemar l'estat de l'inici de sessió, per a emmagatzemar el token del dispositiu...

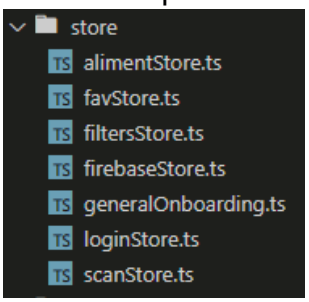


Fig. 52 Vista de la carpeta *Store*

- **Theme**: En aquesta carpeta es troben els fitxers d'estil CSS referents als colors dels 3 temes disponibles (clar, fosc i accessible) a l'aplicació.
- **Views**: En aquesta carpeta es troben les vistes principals de l'aplicació. En l'arrel d'aquesta carpeta, es troben les 4 pestanyes principals, *HomePage*, Explorar, Rebost i Configuració, i les vistes d'inici de sessió i registre. Dins d'aquesta carpeta es poden trobar subcarpetes que fan referència a vistes secundàries de les pestanyes anteriorment mencionades.

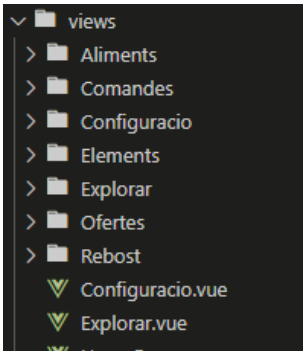


Fig. 53 Vista de la carpeta *Views*

- Finalment, en **l'arrel** de la carpeta **/src**, es troba el component principal anomenat App.vue i les declaracions de tipus de TypeScript per al projecte.

3.2.4. Diferències entre el prototip i el disseny final

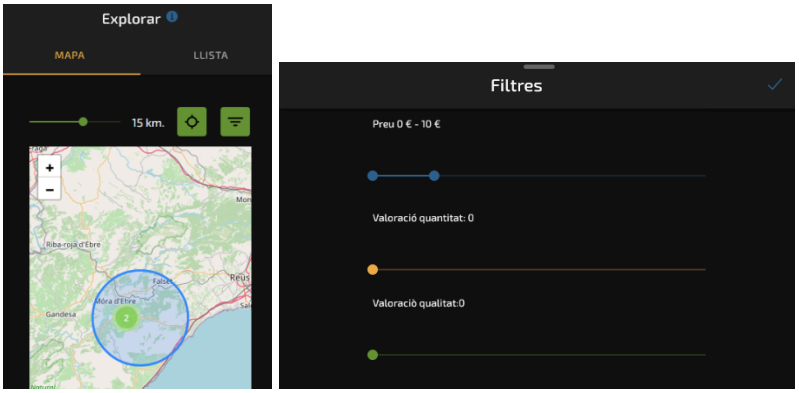


Fig. 54 Vista de la pestanya explorar i dels filtres

Les principals diferències entre el prototip i el disseny final es donen bàsicament al mapa. Ja que el controlador del radi del cercle de cerca i el botó d'anar a la ubicació detectada no estaven pensats. Pel fet que no s'havia implementat el mapa interactiu en el prototip. Per altra banda, als filtres, s'afegeixen controladors diferents per als filtres numèrics, que permeten aplicar els filtres de manera més còmoda. La vista de veure el rebost també canvia, ja que s'ha introduït un controlador a la part superior que permet ordenar els resultats de l'inventari. La vista de les comandes també ha canviat lleugerament. En l'aplicació, s'agrupen les comandes per dies per a poder ajudar l'usuari a localitzar comandes de manera més ràpida. I a més, s'implementa la idea extreta en els tests d'usuaris de valorar les comandes en comptes dels establiments.

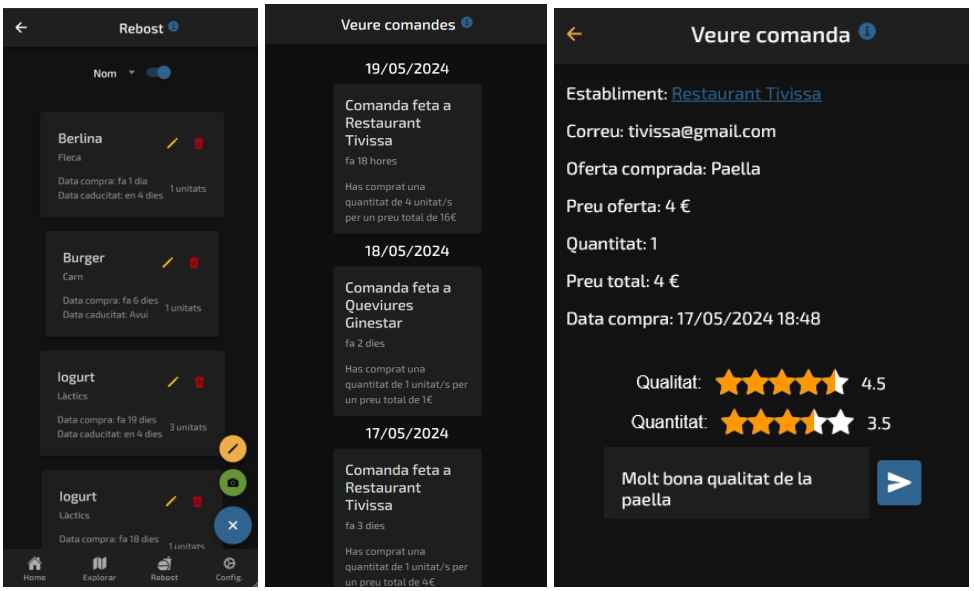


Fig. 55 Vista del contingut d'un rebost comandes i veure el detall d'una comanda

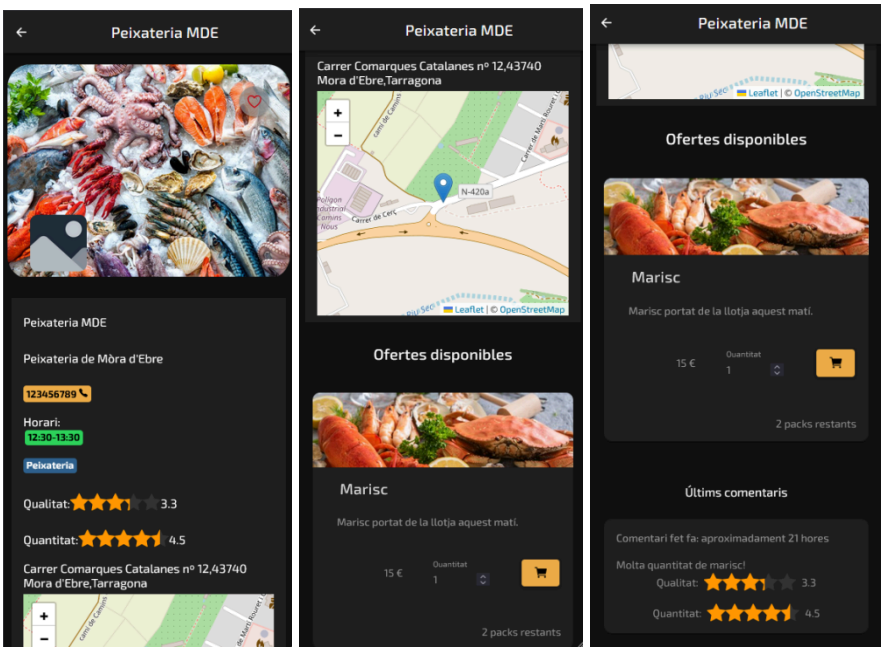


Fig. 56 Detalls de la vista de veure un establiment

Una altra de les vistes que s'han modificat substancialment és la vista de veure un establiment i les seves ofertes. Tal com es pot veure en les imatges, a part de la informació de l'establiment es mostra la localització exacta de l'establiment mitjançant un mapa. També es mostren les ofertes disponibles i es pot comprar directament des d'aquesta vista sense haver d'entrar a la pàgina de l'oferta. A més, en aquesta vista es mostren els últims 5 comentaris que han fet en alguna oferta de l'establiment.

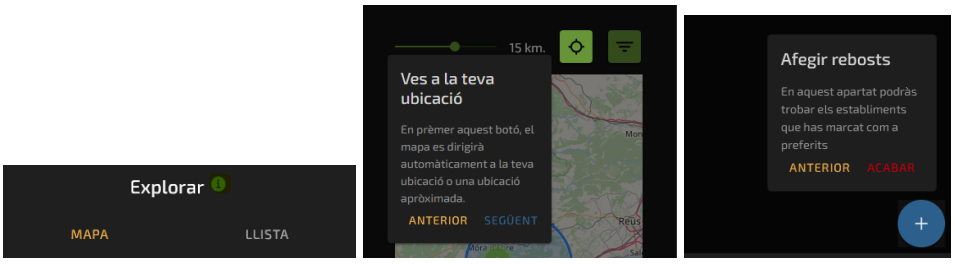


Fig. 57 Vista de la icona que executa *l'onboarding* amb dues mostres de *l'onboarding*

Per altra banda, també s'ha substituït *l'onboarding* que es mostrava en iniciar sessió, per un *onboarding* en cada una de les vistes principals. D'aquesta manera, sempre que ho necessiti, l'usuari podrà accedir a l'ajuda específica de la vista en la qual es troba, tal com es va establir en les proves d'usuaris del prototip.

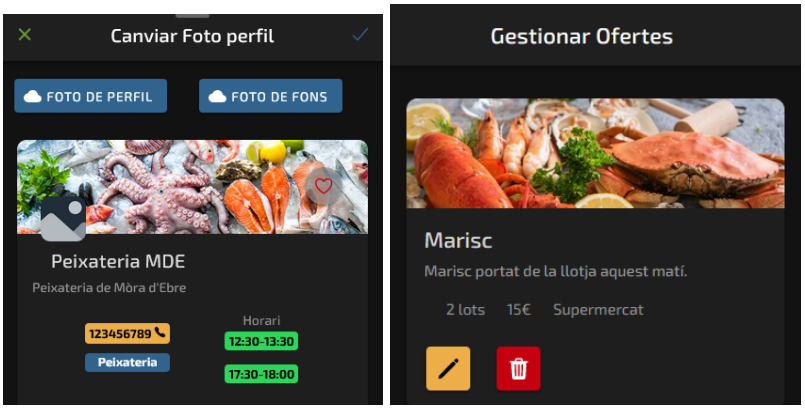


Fig. 58 Vista de canviar la foto de perfil de l'establiment (esquerra) i vista de la gestió d'ofertes (dreta)

Finalment, s'ha afegit una vista a part per a canviar la foto de perfil i la foto de fons de l'establiment. Aquesta vista no estava pensada, ja que l'acció de canviar la fotografia s'esperava integrar al formulari per canviar el perfil. Però, a causa d'alguns errors al moment de processar la sol·licitud a l'API, s'ha decidit fer-ho en una vista nova i s'ha aprofitat per afegir una previsualització de com veuria l'usuari la targeta de l'establiment.

3.2.5. Obtenció de les coordenades en el registre de l'establiment

Durant la implementació de la cerca dels establiments trobo la necessitat d'obtenir les coordenades dels establiments perquè posteriorment els usuaris puguin cercar establiments mitjançant la localització geogràfica. He pensat a demanar directament aquesta dada a l'establiment quan es registra. Tot i que és l'opció més fàcil, es pot donar peu a la creació d'establiments en llocs inexistents i no existiria una manera fàcil de verificar que les dades proporcionades són correctes.

Per aquest motiu es pensa en una alternativa com la possibilitat d'utilitzar l'API **Nominatim**. Aquesta API, mitjançant el recurs **/lookup**, permet realitzar una consulta amb una direcció i l'API retorna les coordenades exactes de la direcció o un error en cas que no trobi la direcció proporcionada. D'aquesta manera, amb una única crida, es verifica que la direcció és correcta i a més a més, s'obtenen les coordenades per emmagatzemar-les directament amb la petició del registre.

Per tant, després de la validació del formulari de l'establiment i abans d'enviar la petició de registre a l'establiment, s'implanta aquesta crida a l'API de **Nominatim** i en cas que es verifiqui la direcció s'enviarà la petició de registre amb les coordenades extretes. En cas que no es trobi la direcció, s'informarà l'usuari perquè revisi la informació de la direcció abans de continuar.

3.2.6. Mapa interactiu i cerca d'establiments propers

Un cop s'emmagatzemen les coordenades dels establiments de forma automàtica, es procedeix a implantar el complement de Capacitor que permet

obtenir les coordenades dels dispositius GPS i d'aquesta manera es pot fer la cerca automàticament a la pàgina principal i es pot mostrar una localització propera quan s'inicialitza el mapa. Tot i això, si el complement de geolocalització no funciona o es bloqueja la localització exacta, l'aplicació fa una crida a l'API mencionada anteriorment My-ip.io. D'aquesta manera l'aplicació utilitza la informació pública de l'adreça IP del client a obtenir una localització aproximada de l'usuari i oferir almenys un servei mínim. Es recorda que la localització obtinguda en aquesta crida no es comparteix amb el servidor ni s'emmagatzema garantint així, la privacitat de la localització dels usuaris.

```
//Funció per a obtenir la posició
const getCurrentPosition = async () => {
    try {
        //En primera instancia intenta rebre les coordenades de capacitor
        const coordinates = await Geolocation.getCurrentPosition();
        //En cas de que tot i que funcioni pero ens retorni la localització 0,0. Executarem el catch
        if (coordinates.coords.latitude == 0 || coordinates.coords.longitude == 0)
            throw new Error("GPS no funciona")
        //En cas de que capacitor ens retorni les coordenades, actualitzem les variables reactives
        mapCoordinates.value[0] = coordinates.coords.latitude
        mapCoordinates.value[1] = coordinates.coords.longitude
        myLocation.value = mapCoordinates.value

    } catch (err) {
        //En cas d'un error, es realitza una crida a la API MyIp.io per a obtenir les coordenades
        //A partir de la ip i mostrar una localització aproximada
        doIPLocation((err: any, data: any) => {
            if (err) return
            myLocation.value = [data.location.lat, data.location.lon]
        })

    }
    //Finalment es canvia la posició en el mapa amb la localització obtinguda
    map.value?.flyTo(myLocation.value)
    //Es realitza la cerca
    fillEstabliments()
};
```

Fig. 59 Codi de com s'obté la localització de l'usuari en cas que Capacitor no funcioni

Seguidament, l'aplicació realitza una petició a l'API per a cercar els establiments propers a la localització. Per a cercar els establiments a partir d'unes coordenades, s'utilitzen les coordenades obtingudes durant el registre dels establiments.

Quant a la implementació del mapa, al principi de la fase d'implementació es presenta la idea d'utilitzar Google Maps o alguna llibreria alternativa que utilitzi Open Street Maps. Una alternativa a Google Maps gratuïta i sense límit d'utilització, que té una potent llibreria per a JavaScript anomenada Leaflet. Fins i tot, al principi de la implantació s'utilitza una portabilitat de Leaflet per a Vue, que proporciona els components de Vue i els mètodes necessaris per a implantar un mapa interactiu de manera molt còmoda. Per tant, es decideix al principi utilitzar aquesta portabilitat de Leaflet per a Vue.

Al principi funciona correctament, però més endavant, quan comencen a amuntegar-se els marcadors en el mapa, no s'aconsegueixen diferenciar els establiments i acaba sent possible utilitzar el mapa. Per aquest motiu, se cerca una manera d'implementar el complement de MarkerCluster [16 *Leaflet. 2024*], que automàticament, detecta els marcadors del mapa que es poden agrupar i forma grups que quan es prem damunt, el mapa s'amplia automàticament. Durant unes quantes hores de proves, es troba impossible trobar alguna funció

al complement per a aconseguir fer els MarkerCluster. Per la qual cosa, es cerca una alternativa.

Es troba una altra portabilitat de Leaflet per a Vue 2 que conté el complement de MarkerCluster integrat, tot i això, es descarta perquè s'està utilitzant Vue 3, ja que Vue 2 ja no té suport. Finalment, es decideix utilitzar la llibreria oficial de Leaflet i adaptar-la per a mostrar-la a Vue amb el complement oficial de MarkerCluster. Amb una mica més de feina s'aconsegueix implementar de nou el mapa i posteriorment, s'aconsegueix afegir el complement correctament.

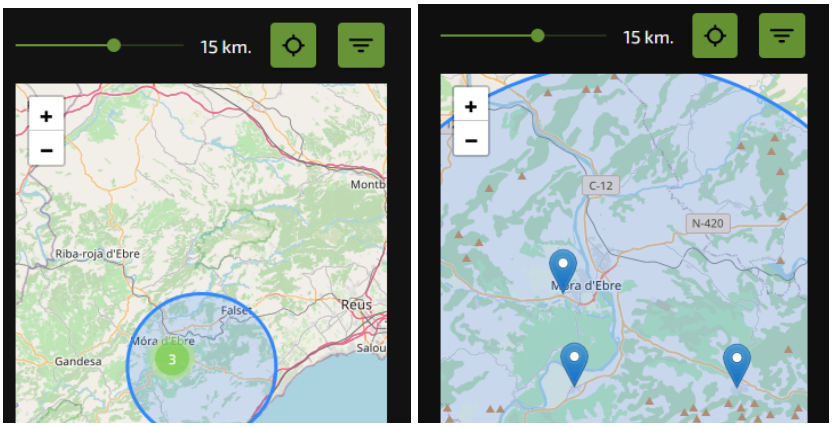


Fig. 60 Vista dels *marker clusters* implementats.

Després de solucionar-lo apareix el següent problema. Amb la llibreria oficial de Leaflet cal crear els marcadors i els *popups* dels marcadors a partir de codi JavaScript i és necessari especificar-li el codi HTML que es mostrarà en el *popup*. Quan es duen a terme les primeres proves els *popups* no es renderitzen correctament i el codi i les etiquetes de Vue tampoc funcionen. A més els disparadors que es creen dins del *popup* no són accessibles des de Vue i els disparadors de Vue tampoc són accessibles des del *popup*. Així doncs, és impossible d'aquesta manera fer *popups* que mostrin informació referent al marcador o poder fer botons dins del *popup* que per anar a la vista de l'establiment. Per aquest motiu, el mapa torna a ser inservible, però ara per aquest motiu.

Després de dur a terme una profunda recerca es descobreix que Vue pot obtenir un component renderitzat de la plantilla en un objecte de JavaScript i integrar-lo al *popup* de Leaflet [17 *Abrab. 2019*]. El problema és que per a fer-ho cal declarar una variable al script amb el mateix nom que l'atribut "**ref**". En el cas que sol calgués mostrar una quantitat fixa de marcadors, aquesta opció solucionaria el problema. Tot i això, la quantitat de marcadors són variables segons la localització del mapa en aquell moment. Per la qual cosa s'intenta fer una llista de referències a elements HTML, però no s'aconsegueix cap resultat, ja que no es pot fer una llista d'elements Ref.

Posteriorment, durant la implementació dels temes i colors de l'aplicació s'utilitza VueUse i durant la cerca per la documentació de la llibreria es troba la

funció "useTemplateRefsList", s'analitza la documentació [18 *Fu, Anthony. 2024*] i resulta que aquesta funció juntament amb un component que proporciona VueUse, permet implementar la idea de tenir una llista de TemplateRefs que havia intentat implementar jo mateix amb anterioritat. Es duen a terme proves amb aquesta funció i finalment s'aconsegueixen crear elements reactius que s'aconsegueixen injectar en l'HTML de la llibreria Leaflet.

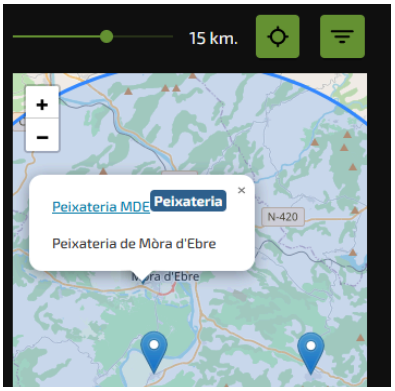


Fig. 61 Vista d'un *popup* de Leaflet amb components de Vue injectats

Un cop es soluciona aquest problema i el mapa acaba funcionant amb el complement dels clústers dels marcadors i els *popups* amb components de Vue. Es procedeix a crear un cercle en el mapa, que representarà l'àrea on es cercaran els establiments, juntament amb un slider, que permet augmentar o disminuir el radi del cercle. Finalment, per a cercar els establiments en cada moviment del mapa, es crea un *trigger* que es dispararà cada vegada que el mapa deixi de moure's. Quan es dispari aquest *trigger*, l'aplicació agafa les coordenades del mapa i el radi del cercle i realitza la petició de cerca a l'API. Quan rep la resposta de l'API amb els resultats, aquests resultats sempre es trobaran dins del cercle dibuixat en el mapa.

Tot i això, després d'implementar-ho, s'observa que el disparador que detecta el moviment del mapa s'executa de forma excessiva i provoca una gran quantitat de sol·licituds inútils al servidor. Per aquest motiu la possible solució que es pensa és en utilitzar la funció "useDebounceFn", també de VueUse. Aquesta funció permet especificar un temps d'espera des de l'última vegada que es crida la funció fins que s'executa. És a dir, si una funció té un *debounce* de 500 ms, i es crida una altra vegada abans que passin aquests 500 ms, la funció no s'executarà i reiniciarà el comptador de temps. La funció que s'executarà serà l'última crida realitzada de la funció passats els 500 ms de *debounce*. D'aquesta manera, s'aconsegueix reduir dràsticament la gran quantitat de peticions que es duen a terme a l'API a causa de la navegació amb el mapa.

Per altra banda, també és necessari aplicar filtres en la cerca. Per a aconseguir la integració de la vista dels filtres (per escollir els filtres) amb les vistes del mapa i de les llistes (per a observar els resultats de la cerca), s'implanta una *store* amb la informació dels filtres escollits en la vista dels filtres. I les vistes dels resultats que realitzen la cerca amb els filtres indicats quan aquests

canvien, ja que recordem que les *store* de Pinia també utilitzen variables reactives.

3.2.7. Reconeixement d'aliments en fotografies de tiquets

Durant el principi de l'etapa d'implementació es va provar de forma ràpida que la càmera pogués funcionar en totes les plataformes, per a descartar la possibilitat d'haver de buscar alternatives o directament descartar la funcionalitat. Després de les proves s'aconsegueix implementar el complement de Capacitor per a utilitzar la càmera en tots els dispositius. Posteriorment, es prova d'integrar la llibreria tesseract.js i sorprenentment funciona sense cap mena de problema tant en les proves de la versió web com en les proves en Android. Inclús el rendiment d'una llibreria com aquesta en l'aplicació d'Android sorprèn, i es pot continuar endavant amb la funcionalitat. Tot i això, en ser fotografies fetes amb un mòbil, Tesseract detecta molts més caràcters i símbols estranys que no es troben en la fotografia. A més a més, s'observa que el Tesseract té problemes per a diferenciar entre diferents caràcters com la i majúscula **i** la **t** majúscula. Fet que no beneficia gens, ja que pot confondre "Iogurt" per "Togurt".

Per a poder solucionar aquests problemes, primerament es pensa a fer un preprocessament mínim a les imatges per aconseguir millor resultats i reduir el soroll de les imatges, seguint les recomanacions generals per a tractar les imatges d'entrada d'un *OCR* [19 *KlearStack. 2024*]. Per a aconseguir-ho, es cerca una llibreria que es pugui executar al navegador, no utilitzi programes externs i a més sigui tan lleugera com sigui possible. Aquesta llibreria es tracta de image-js, gràcies a aquesta llibreria s'aconsegueix implementar un filtre per a canviar la imatge a blanc i negre i s'aconsegueix implementar un filtre de mitjana per a poder ressaltar les cantonades de les lletres:

```
const preprocessImage = async (imagePath: any) => {
  let img = await Image.load(imagePath);
  img = img.grey();
  img=img.medianFilter()
  img=img.resize({ width: 1200 });

  return await img.toDataURL();
};
```

Fig. 62 Codi de transformació de la imatge

Després d'aquestes millores els resultats del tesseract.js són més nets i no apareixen tants caràcters estranys. Tot i això, el tesseract.js continua sent incapaç de diferenciar alguns caràcters en algunes paraules i, per tant, en aquesta etapa, en els millors dels casos els resultats que detecta durant l'escaneig són com a màxim 1 o 2.

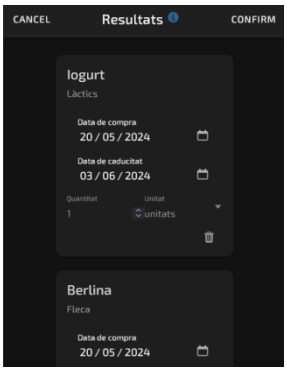
Això es deu al fet que un cop l'*OCR* retorna la cadena de caràcters, l'aplicació cerca cada paraula de la sortida a una llista on es troben els noms dels aliments per saber que una paraula referència surt en el tiquet. Quan l'aplicació realitza la cerca, les comparacions les fa d'igualtat, i moltes vegades compara

cadenes com "Iogurt"=="Togurt" i és fals, és clar. Però nosaltres com humans podríem arribar a raonar que realment la paraula "Togurt" significa "Iogurt". Per aconseguir arribar a aquest raonament a l'aplicació es decideix implementar la llibreria string-similarity. Que permet comparar dues cadenes mitjançant la distància de Levenshtein [20 *Nam, Ethan. 2021*]. Aquesta comparació retorna un nombre entre 0 (No s'assemblen gens) i 1 (Són iguals). A partir d'aquesta comparació, es defineix un llindar del 0.7, perquè quan es detecti una paraula semblant a un aliment en més de 0.7, s'afegirà l'aliment als resultats de l'escaneig. D'aquesta manera s'aconsegueix millorar moltíssim la qualitat de l'escaneig de tiquets.

```
//Funció per a cercar els aliments
// de la base de dades si estan presents en el text
//Accepta un paràmetre acceptance per a establir
//el mínim d'acceptació per a afegir l'aliment al rebost
const searchAliment = (text: string, acceptance: number) => {
    //Per cada un dels aliments de la base de dades
    let aliments = getAllNoms.value?.map((val, idx, arr) => {
        let mx = 0
        //Busca l'acceptació de cada paraula del text amb l'aliment
        text.split(' ').forEach((val2, idx2, arr2) => {
            let res = stringSimilarity(val, val2)
            if (res > mx)
                mx = res
        })
        return { similarity: mx, val }
    })
    //Finalment es retornen unicament els aliments de la base de dades
    // que obtenen una similaritat major a al límit establert
    return aliments?.filter((val, idx, arr) => val.similarity > acceptance)
}
```

Fig. 63 Vista del codi que cerca la similitud de les paraules del text escanejat amb els aliments de la base de dades

En una de les proves fetes, s'escaneja un tiquet en castellà per a testejar si aconsegueix escanejar Iogurt, escrit d'una altra manera.



2	YOGUR NATURAL X6	1,05	2,10
1	JAMON EXTRAFINO TREV		3,24
1	JAMONCITO CERTIFICA		4,99
1	12 HUEVOS SUPER GRAN		2,98
3	LECHE ENTERA	0,91	2,73
1	BERLINA CACAO 4 UDS		1,35
1	FRESH 0%ALCOHOL ENJ.		1,35
1	CAFE MOLIDO MEZCLA		1,60
1	CAP. EXTRAFORTE ALUM		3,60
2	ROLLO HOGAR DOBLE	2,35	4,70
1	BANANA		
	0,802 kg	1,45 €/kg	1,16
1	BANANA		
	1,320 kg	1,45 €/kg	1,91

Fig. 64 Vista dels resultats de l'escaneig (esquerra), tiquet utilitzat (dreta)

I tal com es pot observar en la captura anterior els resultats són els esperats. Per tant, es pot donar per acabada i com a satisfactòria la implementació d'aquesta funcionalitat.

3.2.8. Notificacions

Per a poder implementar notificacions push que es mostrin a dispositius Android i navegadors web inclús quan l'aplicació es troba tancada o en segon pla, és necessari la utilització de la plataforma Cloud Messaging de Google Firebase. Fins aquest projecte no havia tingut l'oportunitat de treballar amb Firebase i tot i que al principi ha resultat una mica confós, finalment, s'ha aconseguit implementar el sistema de notificacions sense cap mena de problema. Per a fer-ho, s'ha hagut de crear un projecte de Firebase amb dues aplicacions. Una configurada com a web i l'altra configurada com a Android.

En el moment d'implementar les notificacions amb Capacitor i Firebase es troba un problema no esperat. El complement per a rebre les notificacions no funciona en la versió web, per aquest motiu, s'ha de cercar una solució que funcioni en totes les plataformes. Així doncs, després d'hores de recerca es decideix implementar dos sistemes per a rebre notificacions en l'aplicació. Per als dispositius mòbils es continua utilitzant el complement de Capacitor [21 *Capacitor. 2024*] per a les notificacions i per la versió web s'utilitza la llibreria de Firebase per a JavaScript [22 i 23 *Google. 2024*].

És per aquest motiu que a Firebase és necessari crear dues aplicacions, ja que les notificacions en tots dos casos funcionen diferent. Per a diferenciar entre les dues plataformes, l'aplicació primer inicialitza el servei de Capacitor. En cas que es produeixi un error, l'aplicació el detecta i inicialitza la llibreria de Firebase.

D'aquesta manera, tant si s'executa l'aplicació en una plataforma o en una altra, s'executarà la llibreria necessària per a generar un token que farà referència al dispositiu d'un usuari on ha iniciat sessió, que s'enviarà al servidor quan aquest comenci sessió. Aquest token s'enviarà a l'API i es guardarà a la base de dades, de manera que quan el servidor necessiti enviar una notificació a un usuari en concret, trametrà una notificació als tokens emmagatzemats en un usuari.

Es notificarà a un usuari quan:

- Es notifica als dos tipus d'usuaris, per a informar-lo que té algun aliment caducat en el seu rebost. Com que els dos tipus d'usuari poden tenir rebost, es notificarà als dos tipus d'usuari per igual.
- Es notifica als usuaris de tipus client quan un establiment que tenen marcat com preferit, ha creat una nova oferta o ha modificat les dades d'alguna oferta ja existent.
- Es notifica als establiments, quan un client fa la compra d'alguna de les seves ofertes.

Quan es notifica als establiments quan un client fa la compra d'alguna de les seves ofertes, s'envia la notificació quan es produeix l'acció de la qual avisa. Tot i això, per a notificar els dos tipus d'usuaris quan un element es troba caducat, cal executar periòdicament una funció que cerqui a la base de dades tots els usuaris que tenen algun element a algun rebost que estigui caducat. I

posteriorment els enviï la notificació. Per a executar una funció periòdicament que pugui cercar a la base de dades, s'utilitza node-cron i es programa un enviament els enviaments a les hores següents 18:00, 20:00, 22:00 i 23:00, mitjançant la notació clàssica del crontab de Linux. Aquesta funció executa la funció "getAllUsuarisAmbAlgunElementCaducat", que retorna una primera llista amb els usuaris de tipus client i una segona llista amb els usuaris de tipus establiment. Posteriorment, s'utilitza la funció "sendMessageToUser", que automàticament cerca els tokens dels dispositius de l'usuari al qual s'ha d'enviar la notificació.

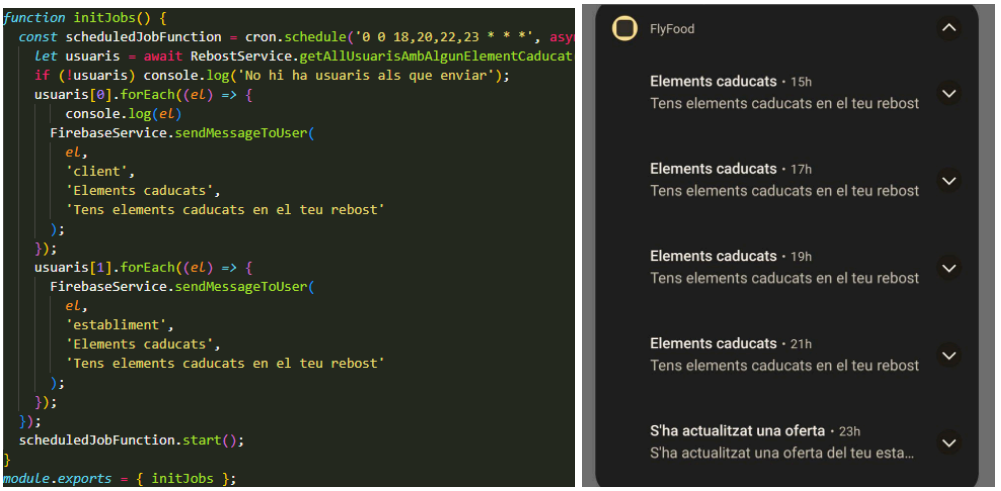


Fig. 65 Captura de les *rutines programades* per enviar notificacions i vista de les notificacions

3.2.9. Tema de l'aplicació

Durant la implementació de l'aplicació amb Ionic i revisant la documentació es detecta que Ionic proporciona una manera fàcil per a definir els colors de les aplicacions, sempre que els temes siguin fosc i clar. En quant és necessari implementar altres temes, Ionic no està preparat, per a fer-ho. Per tant, per aconseguir implementar un tercer tema amb contrast s'ha utilitzat la funció "useColorMode" de VueUse. Aquest petit complement de VueUse, permet definir la quantitat de temes que es vulguin sense cap mena de límit. I a més permet canviar entre els diferents temes de forma fàcil i còmoda. A més, es pot combinar amb les classes CSS dels colors de Ionic i crear temes amb diferents colors, de manera molt fàcil i ràpida.

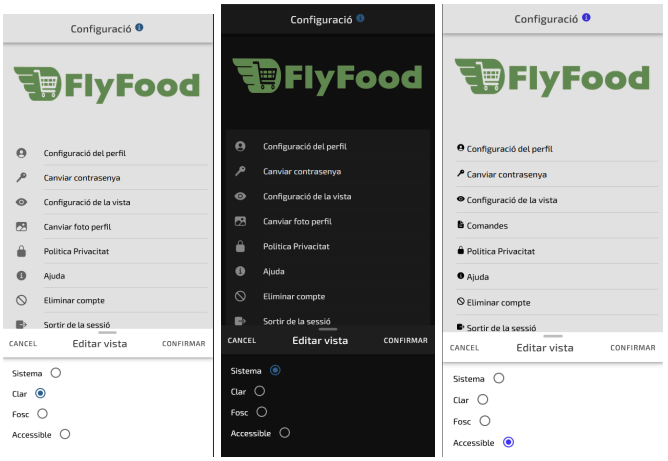


Fig. 66 Vista del formulari per a canviar el tema de l'aplicació

3.2.10. Preferits

Per a gestionar els establiments marcats com a preferits s'ha creat un component de Vue personalitzat que es tracta d'un botó amb la icona d'un cor. Addicionalment, aquest component utilitza una *store* que emmagatzema la llista d'ID dels establiments que l'usuari ha marcat com a preferits. Per altra banda, aquest botó rep com a paràmetre la ID de l'establiment que representa. Llavors únicament el que fa el botó és buscar si l'establiment que li passen com a atribut es troba a la *store* dels favorits. En cas afirmatiu, el botó mostrarà la icona del botó un cor ple. En cas negatiu la icona mostrada serà un cor buit. D'aquesta manera l'usuari podrà diferenciar si un establiment es troba entre els seus preferits o no.

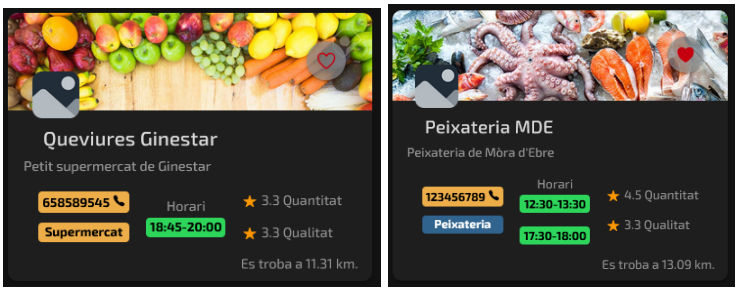


Fig. 67 Vista de dues targetes d'establiments un no preferit (esquerra) i l'altre marcat com a preferit (dreta)

A més a més, quan es premi aquest botó, segons l'estat que tingui, afegirà o eliminarà automàticament l'establiment en qüestió de la *store*. A més, s'enviarà una petició a l'API automàticament, per a mantenir actualitzada la llista d'establiments preferits d'un usuari a la base de dades. D'aquesta manera, quan l'usuari tanqui l'aplicació i torni a iniciar-la obtindrà automàticament la llista d'establiments preferits.

A part de trobar-se en la targeta de visualització de l'establiment, també es trobarà a la vista de visualització d'un establiment concret.

Per altra banda, s'ha de comentar un problema en la visualització dels establiments preferits en la pantalla principal de l'aplicació.

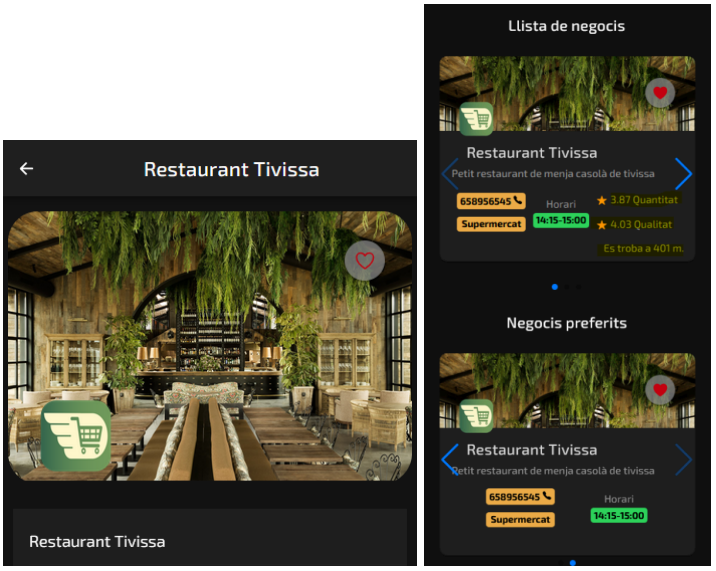


Fig. 68 Vista del botó preferits i de les diferències entre els resultats dels preferits i els resultats de cerca.

Tal com es pot veure en la captura anterior, en la targeta de cerca (la superior) apareix més informació que en la targeta de preferit (la inferior) això és degut perquè quan es fa la consulta de cerca, la distància surt calculada de les coordenades utilitzades, en canvi, en els preferits això no passa. Una cosa semblant passa en les valoracions. Quan es realitza la consulta dels establiments preferits ha resultat impossible recuperar la informació de les valoracions a causa de l'estructura de la base de dades i la consulta que acabaria sent massa costosa. És per aquest motiu la diferència entre aquestes dues targetes. Tot i això, si l'usuari desitja veure aquestes dades, sempre ho podrà fer accedint a la pàgina d'informació de l'establiment clicant damunt de la targeta.

3.2.11. Animacions

Vue Router ofereix una gran quantitat de funcions i Route Guards per a poder aplicar diferents animacions i diferents llibreries d'animacions de manera fàcil i còmoda. Tot i això, passada la meitat de l'etapa d'implementació, m'adono que s'està utilitzant l'etiqueta ion-router-outlet de Ionic, per a renderitzar els components. En intentar canviar l'etiqueta del rúter de Ionic per a utilitzar el Vue Router, apareixen errors en la visualització de les pestanyes. De fet, després d'una mica de recerca [24 *Ionic. 2024*], a la documentació s'especifica que per a utilitzar el component de les pestanyes, s'obliga a utilitzar el component ion-router-outlet. Per tant, es descarta utilitzar vue-router per a les animacions, ja que per molt que es pugui utilitzar per a definir les rutes de ion-router-outlet, ion-router-outlet utilitza una altra manera d'animar els components.

Per animar els components no es poden utilitzar Route Guards i és necessari utilitzar una funció interna de Ionic per a crear les animacions. A més a més, no resulta fàcil aplicar animacions a components sencers, estan pensades bàsicament per a botons, textos... [25 *Ionic. 2024*] Per la qual cosa, no s'han pogut integrar les animacions horitzontals de les pestanyes. Tot i això, s'han aconseguit implementar les animacions dels *Modals* de manera correcta i es veuen correctament.

A més a més, també s'ha creat una animació amb la funció de Ionic. Que es mostra quan s'inicia sessió, quan es tanca sessió, i quan es mostra la fitxa d'un establiment. D'aquesta manera s'intenta animar la majoria de les transicions possibles de la manera més consistent possible tal com s'especifica en l'avaluació heurística.

3.2.12. Inici de sessió de Google

Després del contratemps del principi relacionat amb l'inici de sessió descrit anteriorment, l'inici de sessió de Google es troba en descobert i no sembla haver-hi una alternativa viable. Després d'unes hores de recerca es troba la possibilitat d'afegir un component de Vue a l'aplicació que a partir d'una ID de Google WebOauth renderitza el botó de Google [26 *Wavezync. 2024*]. Aquesta llibreria es tracta de Vue3-Google-Signin.

Aquesta llibreria tot i funcionar en la versió web, un cop compilada per Android el botó no es renderitza, fet que impossibilita iniciar sessió amb Google des de l'aplicació d'Android.

Després de realitzar una profunda recerca en paral·lel al desenvolupament d'altres funcionalitats, es troba un complement no oficial de Capacitor [27 *CodetrixStudio. 2019*]. Aquest complement, permet configurar-lo de manera molt senzilla i permet enllaçar qualsevol botó de l'aplicació perquè funcioni com a botó per a iniciar sessió. S'implementa a la pàgina d'inici de sessió i queda de la següent manera:

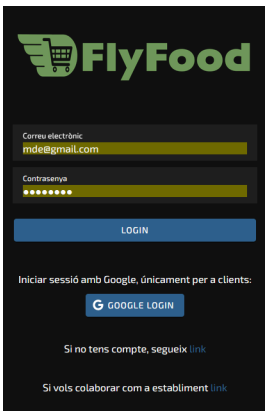


Fig. 69 Aparença de la vista d'inici de sessió amb el botó de Google

Quan es prem sobre aquest botó, la funció enllaçada fa aparèixer un desplegable amb els comptes de Google disponibles per a iniciar sessió. Posteriorment en triar un compte de la llista, la funció del complement rep un token de Google, el qual l'aplicació envia automàticament a l'API.

Quan l'API rep un token d'autenticació de Google, utilitza la llibreria Google Auth, que proporciona un mètode per a desencriptar aquests tokens i obtenir la informació d'inici de sessió. Com el nom, el correu... Quan l'API desencripta aquesta informació cerca a la base de dades el correu obtingut. En cas, que existeixi, genera un token JWT perquè el client pugui utilitzar l'API com si hagués utilitzat correu i contrasenya. En el cas que l'usuari no existeixi a la base de dades, es crea l'usuari i posteriorment es genera el JWT i s'envia al client.

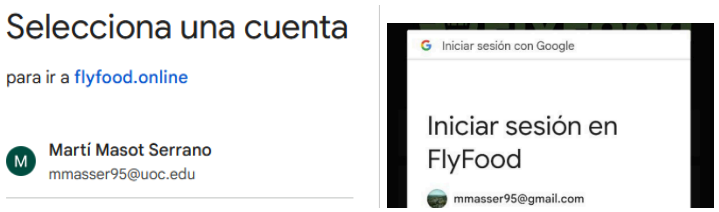


Fig. 70 Vista de l'inici de sessió de Google a Windows (esquerra) i a Android (dreta)

3.2.13. Persistència de la sessió quan es tanca l'aplicació

Quan es tanca l'aplicació, les variables inicialitzades a les *stores* es perden i, per tant, el token d'inici de sessió dels usuaris es perd. En conseqüència, per a mantenir la sessió iniciada s'ha ideat el següent procediment perquè l'aplicació pugui obrir sessió automàticament, un cop torna a executar-se.

1. A partir d'ara, quan s'inicia sessió, a part d'emmagatzemar el token a la *store*, s'emmagatzemarà també a l'emmagatzematge local (*local storage*) de l'aplicació.
2. Quan l'aplicació es posa en marxa, buscarà al *local storage*, el token emmagatzemat. Si existeix, és que l'última vegada que es va engegar l'aplicació es va tancar, sense tancar sessió, per la qual cosa quedarà emmagatzemat i s'utilitzarà aquest token per a renovar-lo i iniciar sessió. En cas que no existeixi el token al *local storage*, es demanarà obrir sessió, ja que s'haurà tancat la sessió o no s'haurà accedit mai.
3. Quan el client envia el token guardat, el servidor el verifica i verifica que no estigui caducat. En cas que no sigui vàlid es retornarà una resposta que obligarà l'usuari a tornar a iniciar sessió a l'aplicació.
4. En cas que el token sigui vàlid, el servidor generarà un nou token amb la data de caducitat renovada i s'enviarà una resposta igual com si l'usuari hagués obert la sessió amb correu i contrasenya, d'aquesta manera l'aplicació obrirà sessió automàticament, quan rebi la resposta.

3.2.14. Aplicació iOS

Tot i que desenvolupar aplicacions per múltiples plataformes pot ser un repte, el fet d'haver triat Ionic i Capacitor ha estat una decisió estratègica que ha facilitat enormement el procés de creació de l'aplicació per Android i Web. Aquesta elecció ha permès superar problemes com la integració del botó d'inici de sessió de Google i la implementació de notificacions, oferint una base sòlida i consistent per a ambdues plataformes.

Pel que fa al desenvolupament per iOS, tot i que actualment no disposo d'un dispositiu Mac ni d'un compte de desenvolupador, l'ús de Capacitor fa que la transició sigui relativament senzilla. Una vegada es disposi d'aquestes eines, les passes necessàries per aconseguir un binari per iOS són mínimes gràcies a la integració nativa que ofereix Capacitor.

3.2.15. Onboarding

Després dels contratemps ocasionats en els reptes anteriors es decideix implementar una llibreria *d'onboarding* per a agilitzar el procés i poder tenir una petita guia de les funcionalitats de l'aplicació, de manera ràpida. Tot i això, apareixen alguns errors com els que es poden veure a continuació.

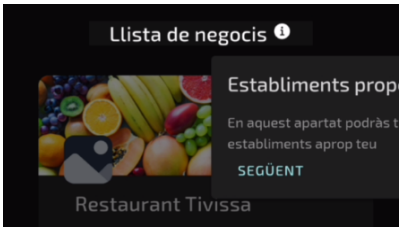


Fig. 71 Vista de l'error de *l'onboarding* en versió mòbil

Tot i això, fixant unes quantes propietats de CSS com l'amplada màxima de les targetes i fixant la posició del *popup* s'aconsegueixen mitigar alguns dels errors de visualització mostrats anteriorment. Per tant, finalment es canvia *l'onboarding* que hi havia a l'inici de la sessió en el disseny, per aquest disseny, que permet assenyalar i explicar components de la interfície, de manera que els usuaris puguin visualitzar correctament i intuïtivament tots els controls els quals se'ls expliquen.

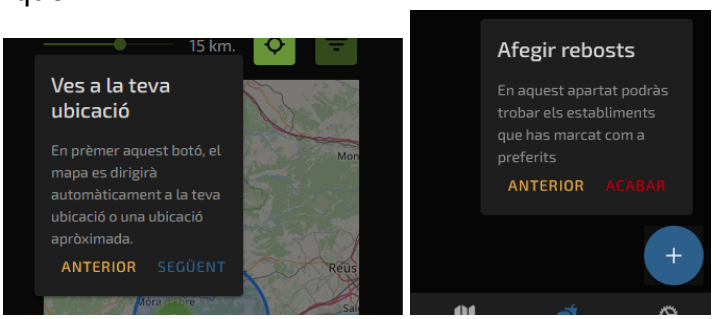


Fig. 72 Vista de *l'onboarding* amb els canvis fets amb CSS.

3.2.16. Mostrar l'aparença del perfil i de l'anunci als establiments

Durant el desenvolupament de la interfície dels establiments m'adono que els establiments no poden veure l'aparença de l'oferta o de la targeta de l'establiment des de les vistes on poden accedir. És per això que es crea una vista addicional a la configuració on ells mateixos poden pujar la fotografia de perfil i la fotografia de fons de la targeta. En aquesta vista es mostra una previsualització de la targeta de l'establiment, on es pot veure l'aparença que tindrà l'establiment als usuaris.

A més a més, a la vista de gestió d'ofertes es mostra també una previsualització de com es veurà cada una de les ofertes al perfil de l'establiment.

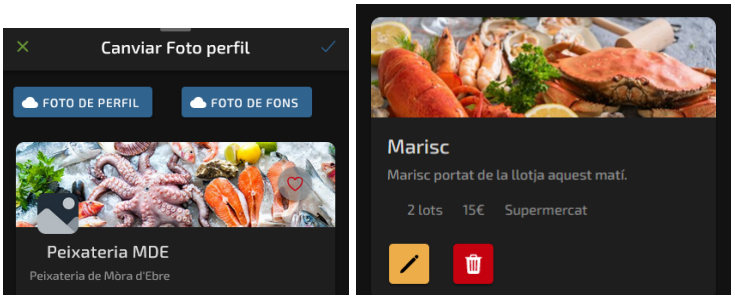


Fig. 73 Vista de canviar la foto de perfil de l'establiment (esquerra) i vista de la gestió d'ofertes (dreta)

3.2.17. Passarel·la de pagament

Acabant la fase d'implementació queda implementar la passarel·la de pagament de les comandes, tot i ser una funcionalitat clau per una aplicació d'aquest tipus, el poc temps disponible per a la seva implementació i la no previsió d'ús comercial de l'aplicació de l'app es descarta. Tot i que en el cas que posteriorment l'app funcionés per a usos comercials reals, es podria implementar sense cap mena de problema, ja que únicament caldria adaptar el procés entre confirmar la comanda a l'aplicació i confirmar la comanda al servidor i afegir-hi la passarel·la de pagament.

3.2.18. Verificar usuaris i recuperació de contrasenya

Durant la implementació de l'aplicació es troba la necessitat d'implementar un sistema per a verificar que els usuaris que es registren a la plataforma, siguin usuaris amb correus reals. També es troba la necessitat d'implementar un sistema perquè qualsevol usuari pugui canviar la contrasenya d'accés si no se'n recorda d'aquesta. Tot i això, per a la implementació d'aquestes funcionalitats implica que l'API pugui enviar correus, i per tant seria necessari implementar un servei SMTP, el codi necessari a l'API per a consumir aquest servei, així com la lògica de les funcionalitats, dissenyar una nova vista per a la recuperació de la contrasenya, fer proves d'integració...

Per aquest motiu i com que no ha estat planejat durant les fases inicials, aquest objectiu es descarta en aquesta fase, però es deixa constància que queda com a objectiu principal en la pròxima iteració del projecte.

3.2.19. Empaquetament, desplegament i posada en marxa

Durant tot el transcurs de la fase d'implementació, cada dia o cada dos dies, s'han fet proves d'empaquetament de l'app per a Android i s'han pogut fer proves exhaustives del comportament de l'App en Android envers l'app web. Per aquest motiu, tots els reptes plantejats no han impedit poder tenir una versió actualitzada de l'App, empaquetada per Android en el moment d'acabar la fase d'implementació.

Tot i això, el desplegament de l'API i l'aplicació web sí que s'ha vist modificat, a causa també de la falta de temps. Ja que no s'ha pogut desenvolupar un Dockerfile per a definir la imatge personalitzada dels dos projectes i el docker-compose que munta el sistema de contenidors automàticament. Tot i això, s'ha realitzat el desplegament en una màquina virtual de Google, la qual també ofereix les mateixes característiques que els contenidors, com l'escalabilitat, la tolerància a fallades... Però com a conseqüència, el desplegament del servidor és manual i, per tant, no es pot fer automàticament com en el cas dels contenidors. Per aquest motiu, l'arquitectura final del desplegament de l'API és el següent:

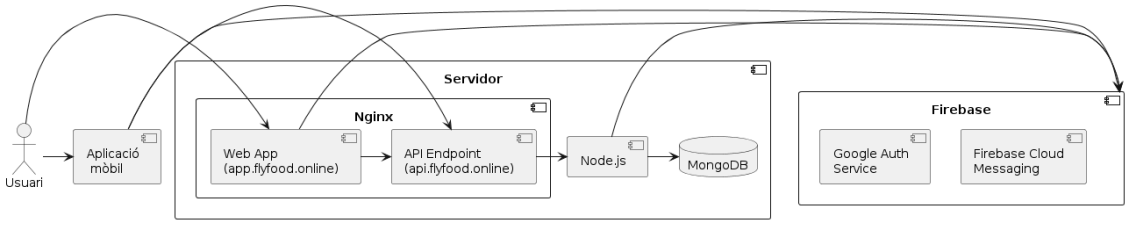


Fig 74. Esquema de l'arquitectura general de l'aplicació després de la implementació.

3.3 Tests

Els tests unitaris permeten assegurar que cada unitat individual de codi, com ara funcions, mètodes o components, funciona correctament de manera aïllada i compleix amb les especificacions establertes. D'aquesta manera, s'automatitzen les proves i es poden detectar fallades de forma molt ràpida, millorant l'eficiència i la fiabilitat del desenvolupament del programari. Per tant, per a fer uns tests unitaris on es puguin provar diferents aspectes dels components de Vue, s'utilitza Vitest de Vite juntament amb Vue-test-utils.

Aquestes eines proporcionen un entorn de test de components on es poden renderitzar els components encapsulats i provar la lògica de cadascun per separat. A més a més, Vitest ofereix una funcionalitat per a fer *mock* [28 *Vladimir, Anthony Fu, Patak…*] a mòduls sencers. És a dir, quan es fan les proves unitàries d'un component que necessita dades de l'API, per a mostrar informació, es pot fer *mock* a les crides a l'API per a simular-les i no haver de dependre de l'estat de les dades de la base de dades per a poder passar els tests satisfactòriament i no haver d'anar-los editant a mesura que canvia la base de dades.

D'aquesta manera, es garanteix que els tests sempre rebran les mateixes dades i, per tant, es podrà testejar, si es renderitzen la quantitat de components segons les dades simulades i si s'ha renderitzat amb la informació esperada i en el lloc esperat.

Per a exemplificar el procés de *testing* es testeja un element simple com el següent:

```
<template>
    <div class="horari" v-if="horaris.length > 0">
        <!-- Bucle que genera un ion-badge per cada interval de l'horari que
            es troba a props.horaris-->
        <ion-badge color="success" data-test="badge" v-for="horari in horaris"
        :key="horari._id">
            {{ horari.inici }}-{{ horari.final }}
        </ion-badge>
    </div>
</template>
<script setup lang="ts">
import { IonBadge } from '@ionic/vue'
import { Horari } from '../types'
const props = defineProps<{
    horaris: Horari[]
}>()
</script>
<style scoped>
.horari {
    display: flex;
    flex-flow: column wrap;
    margin: 2px;
    gap: 15px;
}

</style>
```

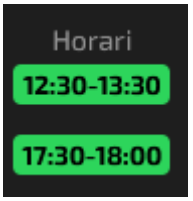


Fig. 75 Captura del codi i vista del component a testejar

Aquest component es mostra en les targetes dels establiments i en la vista de l'establiment per a mostrar l'horari de recollida dels establiments. Aquest component rep una llista d'objectes del tipus:

```
{
  inici:string,
  final:string
}
```

On l'inici correspon a l'hora d'inici d'un interval i final, correspon a l'hora final de l'interval. Aquestes dades el component les rep mitjançant propietats (*props*)

```
const myBadgeFactory = (horaris) => {
  return mount(badgeHorari, {
    props: { horaris },
  });
};

describe("Tests del badge de l'horari", () => {
  test('Es renderitza correctament', () => {
    const wrapper = myBadgeFactory([{ inici: '18:00', final: '21:00' }]);
    expect(wrapper.find('[data-test="badge"]').exists()).toBe(true);
  });
  test('Comprovacions de que no es renderitza quan no cal', () => {
    const wrapper = myBadgeFactory([]);
    expect(wrapper.find('.horari').exists()).toBe(false);
  });
  test('Comprovacions de que es renderitzen correctament dos badges', () => {
    const wrapper = myBadgeFactory([
      { inici: '12:30', final: '13:00' },
      { inici: '18:00', final: '21:00' },
    ]);
    expect(wrapper.find('[data-test="badge"]').exists()).toBe(true);
    expect(wrapper.findAll('[data-test="badge"]').length).toBe(2);
  });
  test('Comprovacions de que es renderitzen els dos badges amb el text esperat',()
    const wrapper = myBadgeFactory([
      { inici: '12:30', final: '13:00' },
      { inici: '18:00', final: '21:00' },
    ]);
    expect(wrapper.findAll('[data-test="badge"]')[0].text()).toBe("12:30-13:00");
    expect(wrapper.findAll('[data-test="badge"]')[1].text()).toBe("18:00-21:00");
  })
});
```

Fig. 76 Vista del codi dels tests

```
 tests/unit/components/badgeHorari.spec.ts (4) 724ms
   Tests del badge de l'horari (4) 719ms
     Es renderitza correctament 446ms
     Comprovacions de que no es renderitza quan no cal
     Comprovacions de que es renderitzen correctament dos badges
     Comprovacions de que es renderitzen els dos badges amb el text esperat

Test Files  1 passed (1)
     Tests  4 passed (4)
  Start at  18:00:54
  Duration  11.51s
```

Fig. 77 Vista de la validació dels tests

En la captura es pot observar el procediment seguit per a dur a terme els tests. Primerament, es crea una funció, que muntarà el component amb les dades que es proporcionin com a arguments.

Seguidament en cada test es munta el component amb els arguments desitjats i es testegen si s'han renderitzat correctament els elements.

- En primer lloc, es pot veure que es testeja que el component es renderitza quan se li envia almenys un element.
- En segon lloc, es testeja que no es renderitza quan no se li proporcionen dades

- En tercer lloc, es munta el component amb dos intervals d'horaris i es testeja que efectivament es renderitzen dos elements.
- En quart lloc, es munta el component amb dos intervals i es testeja que el text renderitzat és l'esperat.

Tot i això, com s'ha pogut comprovar en aquest component no és necessari fer un *mock* de cap llibreria, ja que el component no depèn ni de *stores* ni de crides a l'API. Per a exemplificar el procés de realització d'un *mock* [28 *Vladimir, Anthony Fu, Patak…*] d'una llibreria, en el test d'un component. Es pot dur a terme el test a la *store* que controla els favorits emmagatzemats per un client, per a dur a terme aquest test, s'han seguit les bones pràctiques de la referència [29 *Vue.js. 2024*].

```
import { createFav, deleteFav, getFavs } from '@/APIService/favs';

export const useFavStore = defineStore('fav', () => {
  const favorites: Ref<string[]> = ref([]);

  const setFavs = (favs: string[]) => {
    favorites.value = favs;
  };

  const setLoginFavs = () => {
    getFavs((err: any, data: any) => {
      if (err) return;
      if (data.preferits) setFavs(data.preferits);
    });
  };

  const addFav = (fav: string) => {
    createFav(fav, (err: any, data: any) => {
      if (err) return;
      favorites.value.push(fav);
    });
  };

  const isFav = (fav: string) => {
    return favorites.value.includes(fav);
  };

  const removeFav = (fav: string) => {
    deleteFav(fav, (err: any, data: any) => {
      if (err) return;
      let indexEliminar = favorites.value.indexOf(fav);
      favorites.value.splice(indexEliminar, 1);
    });
  };
```

Fig. 78 Captura del codi de la *store* a testejar

Com es pot observar, aquest component utilitza les funcions **createFav**, **deleteFav** i **getFavs** de l'API per a verificar els canvis a la *store*. Per tant, per a poder testejar la *store*, sense dependre de l'API, es durà a terme un *mock* d'aquestes funcions.

```
import { describe, expect, test, beforeEach, beforeAll, vi } from 'vitest';
import { setActivePinia, createPinia } from 'pinia';
import { useFavStore } from '@/store/favStore';

const myData = { preferits: ['ID-001', 'ID-003'] };

describe('Tests la store dels preferits', () => {
  //Com que la store realitza aquestes crides a la API es realitza el mock
  //Per a obtenir sempre la data definida anteriorment
  beforeAll(() => {
    //S'inicialitza la store de pinia per a poder provar la persistencia de les dades
    setActivePinia(createPinia());
    //Es realitza el mock de la llibreria que fa les peticions a la API
    vi.mock('@/APIService/favs', () => ({
      getFavs: vi.fn((cb) => {
        cb(null, myData);
      }),
      createFav: vi.fn((favs, cb) => {
        cb(null, null);
      }),
      deleteFav: vi.fn((favs, cb) => {
        cb(null, null);
      }),
    }));
  });
```

Fig. 79 Inicialització dels tests

Totes les funcions de la APIService de l'aplicació reben com a paràmetre una funció de *callback* que com a primer paràmetre rep un error i com a segon paràmetre rep la data. Com que en la *store*, l'única funció que retorna dades és la de **getFavs**, es defineix la funció i dins la funció, es crida al *callback* amb les dades que hauria de rebre la *store*, com a segon paràmetre. Les altres dues funcions no utilitzen el paràmetre data, així doncs, es crida la funció *callback* amb null als dos paràmetres.

```
test("S'inicialitza la store (En principi buida)", () => {
  const store = useFavStore();
  expect(store.favorites).toStrictEqual([]);
});

test("S'afegeix una id com a favorit a la store mitjançant el mètode addFav()", () => {
  const store = useFavStore();
  store.addFav('ID-001');
  expect(store.favorites).toStrictEqual(['ID-001']);
});
test("Es comprova amb isFav() que s'ha afegit correctament", () => {
  const store = useFavStore();
  expect(store.isFav('ID-001')).toBe(true);
});
test("Es comprova que funciona l'eliminació", () => {
  const store = useFavStore();
  store.removeFav('ID-001');
  expect(store.favorites).toStrictEqual([]);
});
test("I que a continuació, es retorna false perque la ID s'ha eliminat correctament", () => {
  const store = useFavStore();
  expect(store.isFav('ID-001')).toBe(false);
});
test("Es comprova que s'inicialitza correctament la store amb el mètode setLoginFavs", () =>
  const store = useFavStore();
  store.setLoginFavs();
  expect(store.isFav('ID-001')).toBe(true);
  expect(store.isFav('ID-002')).toBe(false);
  expect(store.isFav('ID-003')).toBe(true);
});
```

Fig. 80 Implementació dels tests

```
 tests/unit/stores/FavStore.spec.ts (6)
   Tests la store dels preferits (6)
     S'inicialitza la store (En principi buida)
     S'afegeix una id com a favorit a la store mitjançant el mètode addFav()
     Es comprova amb isFav() que s'ha afegit correctament
     Es comprova que funciona l'eliminació
     I que a continuació, es retorna false perque la ID s'ha eliminat correctament
     Es comprova que s'inicialitza correctament la store amb el mètode setLoginFavs

Test Files  1 passed (1)
     Tests  6 passed (6)
  Start at  17:47:02
  Duration  1.01s
```

Fig. 81 Verificació dels tests

Finalment, s'escriuen els tests per a verificar que:

- En primer lloc, s'inicialitza la *store* buida.
- En segon lloc, es verifica que quan s'afegeix un preferit, la llista de preferits tindrà un element amb la ID de l'establiment.
- En tercer lloc, es verifica que la funció **isFav** funciona correctament
- En quart lloc, es verifica que s'elimina correctament la ID de l'establiment
- Finalment, es comprova el correcte funcionament de **setFavs**.

Després de l'exemplificació de la feina requerida per a dos dels components més simples es pot arribar a la conclusió que el temps requerit per a dur a

terme tests a l'aplicació ha estat insuficient. A més a més, en aquest punt, la gran quantitat de dependències gràfiques (Elements UI de Ionic, Leaflet, *Onboarding*...) i de llibreries que proporcionen funcionalitats o dades externes als components, resulta molt complicat poder fer tests unitaris dels components més complexos. Per aquest motiu, en aquesta aplicació seria necessari fer tests d'integració o tests "end to end". Aquests tests permetrien fer les proves necessàries per a comprovar de forma automàtica que els components assoleixen els requisits demanats, en tots els casos possibles. A continuació es pot veure una captura dels resultats de tots els tests unitaris implementats:

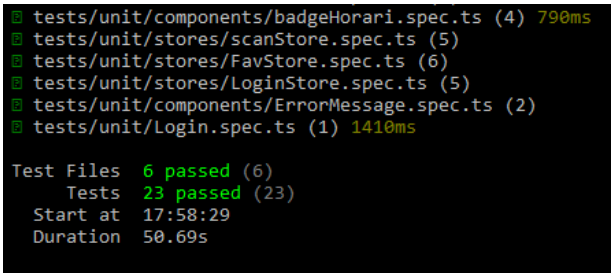


Fig. 82 Vista de la verificació de tots els tests unitaris

4. Conclusions

Un cop acabada la fase d'implementació del treball s'extreuen les conclusions i les línies de treball futures en aquest projecte. La màxima lliçó apresa durant el desenvolupament del projecte ha estat que el més important no és definir una gran quantitat d'objectius, sinó, que el més important és definir objectius assequibles per a poder-los implementar de manera segura tenint temps suficient per a provar l'aplicació. En aquest projecte precisament el que ha passat ha estat això des d'un bon principi es van fixar una gran quantitat d'objectius que encara que una bona part s'hagin assolit, el temps ha sigut massa just i finalment no s'han pogut dur a terme tests al nivell de complexitat en el qual es troba l'app ni altres objectius fixats. Tot i això, aquesta gran quantitat i diversitat d'objectius, m'han enriquit en diferents aspectes del desenvolupament d'aplicacions multiplataforma. En cada una de les etapes de planificació, disseny i implementació he après de primera mà, des de planificar un projecte i definir-ne els objectius, passant per dissenyar cada un dels aspectes claus fins a arribar a obtenir el producte final després de la implementació. A més a més, s'han consolidat molts dels conceptes apresos durant el Grau d'Enginyeria Informàtica.

Però tal com s'ha comentat durant tot el projecte, aquesta gran quantitat d'objectius ja va provocar retards en les tasques de la fase 2. Aquestes tasques estaven planejades per a ser acabades el 3 d'abril, tot i això, es va acabar aquesta fase el 10 d'abril, a causa de la complexitat del disseny del prototip i l'extensa documentació que això comporta. Després d'aquest retard i el retard produït per haver d'implementar un sistema d'inici de sessió des de 0, ja que Passport.js no complia amb els requisits. La implementació del disseny es veu endarrerit cinc dies. A més a més, altres reptes com el mapa interactiu, les notificacions i l'inici de sessió, no estaven contemplades com tasques molt extenses i que finalment, han provocat més retards durant la fase d'implementació de la lògica de negoci, tant del servidor, com de l'aplicació. Havent sofert aquests retards, quan s'intenta implementar la funcionalitat de llegir les fotografies dels tiquets, aquesta s'implementa amb menys temps del planejat. Per aquest motiu, es guanya una mica de temps extra que serà aprofitat per a fer els tests i el desplegament de l'aplicació. Encara així no es poden dur a terme els tests prou complexes que permetin testejar automàticament totes les funcionalitats de l'aplicació i altres objectius que es detallen més endavant en aquesta secció.

Així doncs, es pot concloure que tot i seguir una planificació que pot semblar raonable a primera vista, una gran quantitat d'objectius o definir un abast massa extens, pot comportar perillosos retards en la planificació que poden comprometre l'assoliment final del projecte. Tot i això, s'han assolit la majoria dels objectius:

- Que els clients puguin cercar establiments propers a la seva ubicació.
- Que els clients puguin cercar establiments a partir d'una llista.
- Que els clients puguin cercar establiments a partir d'un mapa interactiu.

- Que els clients puguin aplicar filtres de cerca als establiments.
- Que els clients puguin marcar certs establiments com a preferits.
- Que els establiments puguin publicar i gestionar totes les ofertes que trobin necessàries.
- Comprar ofertes dels establiments (Tot i no haver pogut implementar la passarel·la de pagament, s'ha implementat la simulació de la compra d'una oferta per a veure el comportament de l'aplicació).
- Valorar les comandes fetes als establiments.
- Que els establiments puguin veure les valoracions fetes pels clients.
- Veure l'historial de comandes.
- Poder iniciar sessió amb un compte de Google.
- Que un client sigui notificat quan un establiment afegeixi una nova oferta al sistema (en aquest cas s'ha implementat si l'establiment es troba a la llista de preferits del client).
- Que un establiment sigui notificat quan un client faci una comanda d'una oferta publicada.
- Que tant clients com establiments puguin portar l'inventari dels seus rebosts i puguin ser notificats de quan tenen aliments en els rebosts a punt de caducar.
- Que tant clients com establiments puguin agilitzar la introducció d'elements dels seus rebosts mitjançant la fotografia d'un tiquet.

Tot i haver seguit la planificació al màxim i haver complert els objectius anteriors, malauradament no s'han complert els objectius següents:

- **Implementar l'aplicació per a iOS**. Pels motius ja explicats en la secció 3.2.14.
- **Proves amb usuaris**: Les proves amb usuaris de la fase 3 també s'han vist cancel·lades a causa dels endarreriments i no s'han pogut dur a terme per a obtenir una opinió des de la visió d'un usuari.

Les línies de treball futur que no s'han pogut explorar en aquest treball han estat:

- Implementar una **passarel·la de pagament**, per a poder utilitzar l'aplicació comercialment.
- Dur a terme **tests d'integració i "end to end"**, per a comprovar automàticament que tot el conjunt de l'aplicació funciona correctament.
- Realitzar el desplegament amb **Kubernetes** a Google, per a millorar la tolerància a fallades i automatitzar el desplegament automàtic de l'API o de l'aplicació.
- Implementar la **verificació de correus** quan un usuari es registra a la plataforma, enviant un codi per a verificar posteriorment el compte a l'aplicació.
- Implementar funcionalitats per a poder **canviar la contrasenya** o recuperar el compte en cas d'haver-la perdut.
- Adaptar l'aplicació i implementar el codi necessari per a poder compilar l'aplicació per a **iOS**.
- Adaptar el codi semànticament perquè un **lector de pantalla** pugui reconèixer adequadament totes les accions disponibles a la pantalla i pugui utilitzar l'aplicació com qualsevol usuari.

5. Glossari

API (*Application Programming Interface*): Conjunt de protocols, rutines i eines que permeten la comunicació i la integració entre diferents aplicacions de programari. Una **API** defineix les funcions disponibles per ser utilitzades per altres programes, facilitant la interoperabilitat i l'extensibilitat de les aplicacions.

Avaluació heurística: Aquesta metodologia es basa en l'establiment d'un conjunt de bones pràctiques de disseny, conegudes com a heurístiques. L'objectiu és assegurar que aquestes heurístiques es compleixin en totes les fases del disseny. D'aquesta manera, es poden considerar uns estàndards mínims en el disseny, millorant així la usabilitat i l'experiència de l'usuari.

Benchmarking: És un procés sistemàtic que consisteix a comparar els productes, serveis o processos de l'organització amb els dels competidors o empreses líders en el sector. L'objectiu és identificar les millors pràctiques i establir objectius per millorar el rendiment. El **Benchmarking** pot ser utilitzat en qualsevol fase del disseny o desenvolupament per assegurar que es compleixen uns estàndards mínims i per identificar àrees de millora.

DCU (Disseny centrat en l'usuari): Aquestes són un conjunt de metodologies que posen l'usuari al centre del procés de disseny. L'objectiu és comprendre les necessitats, expectatives i comportaments de l'usuari per crear productes que proporcionin una experiència d'usuari òptima. Aquestes tècniques inclouen la recerca d'usuaris, la creació de persones, l'avaluació heurística, els tests d'usabilitat, entre d'altres.

Framework: Estructura de programari de suport sobre la qual es poden desenvolupar programes. Proporciona una base predefinida sobre la qual es pot construir i ampliar aplicacions, oferint eines i llibreries que ajuden a simplificar i estandarditzar el procés de desenvolupament.

Middleware: Programari que actua com a intermediari entre diferents aplicacions o components, facilitant així la comunicació, la gestió de les dades i la interoperabilitat. S'utilitza sovint per a la integració de sistemes, la gestió de transaccions i en la connexió de serveis web.

Mock: És un objecte d'imitació que s'utilitza en proves unitàries per a substituir components reals del sistema que estan sent provats. Els **mocks** permeten aïllar el component que es vol provar de les seves dependències externes.

Modal: Els **Modals** són un component UI de Ionic que són un tipus de diàleg que apareix des de la part inferior de la pantalla i un cop apareixen, el focus d'acció es troba limitat únicament als components dins del **Modal**. Això ajudarà a enfocar tota l'atenció de l'usuari al **Modal** i a més, no es permetrà a l'usuari sortir del **Modal** si no omple correctament el formulari o cancel·la l'acció.

OCR (*Optical Character Recognition*): En català Reconeixement òptic de caràcters, és una tecnologia que permet convertir imatges digitals de text escrit (com ara fotos de documents escanejats, captures de pantalla o cartells) en text editable. L'**OCR** analitza la imatge identificant-hi els patrons dels caràcters individuals, per a després convertir-los en caràcters digitals que poden ser llegits i processats per un ordinador.

Onboarding: L'**onboarding** és el procés d'integració d'un usuari nou en un producte o servei. El seu objectiu és familiaritzar l'usuari amb el funcionament i les característiques del producte, i ajudar-lo a obtenir una experiència positiva des del principi. En el cas de l'aplicació es poden considerar **onboarding** tant la guia interna de l'aplicació com el manual d'ús.

Plugin: Programa o conjunt de programes que s'integren en un programari més gran per afegir-hi funcionalitats específiques sense haver de modificar el nucli del programari original.

Screen reader: Programari dissenyat per ajudar les persones amb discapacitat visual a utilitzar ordinadors i dispositius mòbils. Llegeix en veu alta el text que apareix a la pantalla, permetent als usuaris navegar per les interfícies i accedir a la informació.

SPA (Single Page Application): Tipus d'aplicació web que funciona dins d'una única pàgina web. Totes les interaccions i actualitzacions de contingut es fan dinàmicament sense necessitat de recarregar la pàgina completa, millorant així l'experiència de l'usuari i la velocitat de navegació.

Typescript: Llenguatge derivat de JavaScript que afegeix tipatge estàtic al llenguatge. Desenvolupat per Microsoft, permet detectar errors en temps de compilació i facilita la creació de codi més robust i escalable, mantenint alhora la compatibilitat amb el JavaScript tradicional.

Wrapper: És una capa d'abstracció que envolta un component existent, proporcionant una nova interfície o funcionalitat sense modificar el codi subjacent. Actua com a intermediari entre l'usuari i el component original, simplificant l'ús d'aquest últim o afegint-hi noves característiques.

6. Bibliografia

[1] **WPF**. 2022. "Informe de la ONU: las cifras de hambre mundial aumentaron hasta 828 millones en 2021 | World Food Programme." Programa Mundial de Alimentos. https://es.wfp.org/noticias/informe-de-la-onu-las-cifras-de-hambre-mundial-aumentaron-hasta-828-millones-en-2021.

[2] **Bolinches, Cristina G**. 2022. "España, tercer país de la UE que legisla el desperdicio alimentario para no tirar a la basura 1.364 millones de kilos de comida." El Diario. https://www.eldiario.es/economia/espana-tercer-pais-ue-legisla-desperdicio-alimentario-no-tirar-basura-1-364-kilos-comida_1_9058353.html.

[3] **EPE**. 2022. "Informe: La quinta parte de la producción alimentaria de la UE acaba en la basura." Diario Información. https://www.epe.es/es/medio-ambiente/20221221/informe-quinta-parte-produccion-alimentaria-80255451.

[4] **Ras, Aina**. 2023. «La Importancia del Diseño Centrado En el Usuario : El Caso de Indago». Sociedad de la Innovación, 22 de setembre de 2023. https://www.sociedaddelainnovacion.es/la-importancia-del-diseno-centrado-en-el-usuario-el-caso-de-indago/.

[5] **Estrategas Digitales**. 2023. «La Importancia del Benchmarking Como Estrategia - Estrategas Digitales». Estrategas Digitales. 17 de febrer de 2023. https://estrategasdigitales.com/blog/importancia-del-benchmarking/.

[6] **Casabona, Eugenia**. 2023. «Larga Vida A la Evaluación Heurística - Eugenia Casabona - Medium». Medium, 11 d'abril de 2023.

https://eugeniacasabona.medium.com/larga-vida-a-la-evaluaci%C3%B3n-heur%C3%ADstica-11a446a2d078.

[7] **TooGoodToGo**. 2023. «Too Good To Go | Salva Buena Comida del Desperdicio». 2023. https://www.toogoodtogo.com/es.

[8] **Encantadodecomerte**. 2024. «Encantado de Comerte - Salva Alimentos Con Hasta un 70% Dto.» Encantado de Comerte. 16 d'abril de 2024. https://www.encantadodecomerte.es/.

[9] **Phenix**. 2023. «Phenix, ¡Actuamos Contra el Desperdicio!» 13 de gener de 2023. https://www.wearephenix.com/es/.

[10] **FoodyBag**. 2024. «Foody Bag - Find Food Deals, Pick-up Free Food or Donate Food». Foody Bag. 2024. https://www.foodybag.com.au/.

[11] **Legaspi, Aaron, i Amit Jakhu**. 2023. «IxD Checklist». IxD Checklist. 2023. https://ixdchecklist.com/.

[12] **Carlos Azaustre**. 2015. «Cómo Implementar Autenticación Basada En Token Con Node.js». Carlos Azaustre. 23 de febrer de 2015. https://carlosazaustre.es/autenticacion-con-token-en-node-js.

[13] **Chris, Kolade**. 2021. «REST API Best Practices – REST Endpoint Design Examples». freeCodeCamp.Org. 16 de setembre de 2021. https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/.

[14] **Express**. 2017. «Security Best Practices For Express In Production». 2017. https://expressjs.com/en/advanced/best-practice-security.html.

[15] **Zanini, Antonello**. 2023. «Best Practices For Building A Scalable Vue.js Application». Codemotion Magazine. 28 de setembre de 2023.

https://www.codemotion.com/magazine/frontend/building-scalable-vuejs-application/.

[16] **Leaflet**. 2024. «GitHub - Leaflet/Leaflet.Markercluster: Marker Clustering Plugin For Leaflet». GitHub. 2024. https://github.com/Leaflet/Leaflet.markercluster.

[17] **Abrab**. 2019. «How Can I Integrate A Vue Component In A Leaflet Popup?» Stack Overflow. 21 de juny de 2019. https://stackoverflow.com/questions/56696928/how-can-i-integrate-a-vue-component-in-a-leaflet-popup.

[18] **Fu, Anthony**. 2024. «VueUse». 2024. https://vueuse.org/core/useTemplateRefsList/.

[19] **KlearStack**. 2024. «How To Improve OCR Accuracy - KlearStack - Medium». Medium, 28 de febrer de 2024. https://medium.com/@klear-stack/how-to-improve-ocr-accuracy-97039aaeabe1.

[20] **Nam, Ethan**. 2021. «Understanding The Levenshtein Distance Equation For Beginners». Medium, 8 de desembre de 2021. https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0.

[21] **Capacitor**. 2024. «Push Notifications - Firebase | Capacitor Documentation». Capacitor Docs. 2024. https://capacitorjs.com/docs/guides/push-notifications-firebase.

[22] **Google**. 2024. «Configurar una Aplicación Cliente JavaScript Firebase Cloud Messaging | Firebase Cloud Messaging». Firebase. 2024. https://firebase.google.com/docs/cloud-messaging/js/client?hl=es.

[23] **Google**. 2024. «Configurar una Aplicación Cliente de Firebase Cloud Messaging En Android | Firebase Cloud Messaging». Firebase. 2024. https://firebase.google.com/docs/cloud-messaging/android/client?hl=es.

[24] **Ionic**. 2024. «Ion-tabs | Ionic Documentation». 2024. https://ionicframework.com/docs/api/tabs.

[25] **Ionic**. 2024. «Animations | Ionic Documentation». 2024. https://ionicframework.com/docs/utilities/animations.

[26] **Wavezync**. 2024. «Vue3 Google Sign-in». 2024. https://vue3-google-signin.wavezync.com/guide/google-signin-button.html.

[27] **CodetrixStudio**. 2019. «GitHub - CodetrixStudio/CapacitorGoogleAuth: Capacitor Plugin For Google Auth. Lightweight & No Dependencies.» GitHub. 28 de junio de 2019. https://github.com/CodetrixStudio/CapacitorGoogleAuth.

[28] **Vladimir, Anthony Fu Ari Perkkiö, Patak, Joaquín Sánchez, Dunqing, Hiroshi Ogawa And Vitest Contributors**. 2024. «Vitest». 2024. https://vitest.dev/guide/mocking.

[29] **Vue.js**. 2024. «Testing Stores | Pinia». 2024. https://pinia.vuejs.org/cookbook/testing.html.

7. Annexos

7.1. Desenvolupament de l'anàlisi Benchmarking

Un cop establerts els criteris d'anàlisi i els productes a analitzar comencem per analitzar el primer producte de la llista **Too Good To Go**.

1. **Registre obligatori per a utilitzar l'app**: Sí, el registre és obligatori per fer ús de l'aplicació. La primera vegada que arranca l'aplicació apareix un formulari d'inici de sessió juntament amb les opcions de registrar-se, iniciar sessió amb Google i de registrar un establiment.
2. **Guia de com funciona l'App**: Sí, aquesta aplicació en la primera execució mostra una petita guia en forma d'una vista amb un *slider* i diferents diapositives sobre com funciona l'aplicació i on trobar les funcionalitats principals.
3. **Layout de l'App**. Aquesta aplicació utilitza una *layout* de l'app en forma de pestanyes. D'aquesta manera es pot accedir a diferents vistes principals mitjançant una barra en la part inferior de l'aplicació. Per altra banda, la navegació dins de l'app és intuïtiva i gràcies a les icones i els textos explicatius permet identificar fàcilment les funcionalitats de cada controlador i el significat de cada dada disponible a la vista. L'app presenta unes icones senzilles, amb un alt contrast i una proporció coherent enfront del text. El text també és fàcilment llegible gràcies a la tipografia i el contrast del color. Totes les ofertes que es mostren a través de l'app es mostren mitjançant un component de tipus targeta que permet agrupar les accions (per exemple, marcar com a preferit) i la informació corresponent a l'element que representa el component.
4. **Mapa d'ofertes i establiments**. Aquesta aplicació ofereix una eina per a cercar ofertes dels establiments mitjançant un mapa. L'aplicació també permet a l'usuari en aquesta vista a filtrar els resultats, segons el tipus de lots, l'horari de recollida, les preferències alimentàries…
5. **Llista d'ofertes i establiments**. A més a més, aquesta aplicació també ofereix llistar aquests establiments un cop hem seleccionat una àrea en el mapa. En aquest llistat també ens apareix un requadre de cerca el qual ens permetrà introduir la localització de forma manual.
6. **Procés de pagament**. Targeta de pagament, Google Pay i PayPal.
7. **Avaluar l'establiment**. Aquesta aplicació ofereix una eina per a poder avaluar un establiment de forma numèrica. Però únicament deixa avaluar un establiment si s'ha fet una comanda a l'establiment. A més a més, l'aplicació també atorga a l'establiment com un tipus de medalla per a informar els usuaris de la quantitat d'ofertes venudes a través de l'app i el temps el qual porta registrat l'establiment a la plataforma.
8. **Estadístiques**: Sí, aquesta aplicació compta amb un apartat on mostra estadístiques als usuaris i a més atorga diferents distincions als establiments segons les seves estadístiques, com els lots salvats o el temps que porta l'establiment venent lots a través de l'app.
9. **Accessibilitat**: Quan s'utilitza Google TalkBack, la transcripció és literal del text i inclús es transcriuen alguns símbols que no apareixen en el text

i que poden confondre als usuaris que necessitin algun *screen reader*, per la qual cosa s'arriba a la conclusió que aquesta aplicació no està preparada per a *screen readers* i no té especificat els atributs "ARIA". Per l'altra banda, aquesta aplicació no ofereix cap mena d'opció per a poder canviar l'esquema de colors de l'aplicació, fet que pot limitar l'ús de l'aplicació a certes persones.

10. **Verbositat**: L'aplicació mostra missatges a l'usuari cada vegada que aquest realitza una acció al sistema, quan afegeix un establiment com a preferit, quan en el mapa no es troben resultats…

Seguidament procedirem amb **Encantado de comerte**.

1. **Registre obligatori per a utilitzar l'app**. Sí, aquesta app requereix que l'usuari inici sessió perquè es pugui utilitzar.
2. **Guia de com funciona l'App**: L'app no ofereix cap mena *d'onboarding* explicant les funcionalitats ni com funciona l'aplicació.
3. **Layout de l'App**. Aquesta aplicació també compta amb una *layout* de pestanyes on l'usuari navega a través de les diferents vistes gràcies a les pestanyes. Per altra banda, la navegació a través de l'App és intuïtiva i simple, ja que no conté tantes funcionalitats com l'aplicació anterior. Finalment, les icones i la barra de pestanyes de la part inferior de la pantalla es veuen extremadament grans en proporció de la lletra. A més a més, els colors escollits per a les icones, el text i el fons, ofereixen poc contrast per a usuaris amb problemes de visió. Aquesta aplicació també representa cada una de les ofertes amb un component de tipus targeta el qual permet agrupar les accions i la informació de l'element que representa.
4. **Mapa d'ofertes i establiments**. No, l'aplicació no permet cercar les ofertes i establiments propers a través d'un mapa interactiu. Fet que és una mica incòmode, en el cas de trobar-te en una àrea extensa amb nuclis de població molt dispersos, has d'anar introduint manualment els nuclis de població un en un a la llista amb l'esperança de trobar alguna oferta.
5. **Llista d'ofertes i establiments**. La llista d'establiments ordena, en primer lloc, les ofertes disponibles primer i ordenades per temps restant; tot i això, la presentació de les targetes és poc atractiva, ja que el logotip de l'empresa es casi inapreciable i el component no aporta molta informació sobre l'oferta. Un punt fort d'aquesta aplicació amb la llista és que permet filtrar les ofertes per tipus d'aliment.
6. **Procés de pagament**. Pagament amb targeta mitjançant una passarel·la de pagament anomenada Stripe.
7. **Avaluar l'establiment**. Únicament es veu la quantitat d'usuaris que han marcat l'establiment com a preferit. A més a més, aquesta xifra es troba dins del cor que permet a l'usuari marcar l'establiment com a preferit i això no permet visualitzar de manera fàcil el nombre d'usuaris, ja que el nombre que apareix és massa petit.
8. **Estadístiques**. Aquesta aplicació no ofereix estadístiques sobre l'ús de l'aplicació, és a dir, no apareix, per exemple, el nombre de lots salvats de l'usuari ni el nombre de lots salvats dels establiments.

9. **Accessibilitat**. Aquesta aplicació funciona correctament amb Google TalkBack, dona una descripció correcta de tots els elements i controls disponibles a la pantalla. Tot i això, aquesta aplicació no ofereix la possibilitat de canviar els colors del tema de l'aplicació ni modificar la mida del text de l'aplicació i a més a més certes icones de l'aplicació no ofereixen suficient contrast per a una visualització correcta.
10. **Verbositat**. No, aquesta aplicació no informa textualment l'usuari quan ha completat una acció com quan emmagatzema a preferits algun establiment. No es pot analitzar si notifica a l'usuari quan ha realitzat una comanda, ja que no existeixen ofertes a la meva zona per a poder-ho analitzar. Tot i això, quan no es troben establiments a la zona mostra un missatge amb les accions que podem dur a terme per a buscar altres establiments.

A continuació, es procedeix a analitzar **Phenix**.

1. **Registre obligatori per a utilitzar l'app**. Sí, és necessari registrar-se i iniciar sessió per a utilitzar l'app.
2. **Guia de com funciona l'App**. Sí, la primera vegada que s'utilitza l'app es mostra un *onboarding* on s'explica quines són les funcionalitats i els objectius principals de l'app.
3. **Layout de l'App**. Aquesta aplicació també ofereix una *layout* amb pestanyes, fet que permet agrupar les diferents funcionalitats en pestanyes i ajuda que la navegació a través de l'app sigui intuïtiva, a més d'utilitzar icones i textos aclaridors en cada flux d'interacció de l'app. Aquesta app també representa cada una de les ofertes amb un component de tipus targeta que permet agrupar la informació de l'element que representen.
4. **Mapa d'ofertes i establiments**. Aquesta aplicació permet cercar ofertes i establiments en un mapa. A més a més, aquesta aplicació permet filtrar el tipus de comerç i en cada marcador en el mapa es pot veure una icona que representa el tipus d'establiment de manera fàcil i intuïtiva.
5. **Llista d'ofertes i establiments**. Si, aquesta aplicació també permet cercar ofertes i establiments a partir d'una llista i també permet en la llista filtrar les ofertes pel tipus d'establiment, a més a més, de filtrar en l'horari en el qual es pot passar a recollir l'oferta.
6. **Procés de pagament**. Pagament mitjançant una passarel·la de pagament amb targeta.
7. **Avaluar l'establiment.** Sí, permet avaluar l'establiment amb una nota numèrica únicament si l'usuari ha fet alguna comanda. D'altra banda, l'aplicació atorga un tipus de medalles que permeten informar a l'usuari de les estadístiques de l'establiment i diferenciar els millors establiments.
8. **Estadístiques**. Sí, aquesta app disposa d'un apartat on es poden veure les estadístiques dels establiments
9. **Accessibilitat.** Les accions i icones no estan preparades per a ser utilitzades amb Google TalkBack, ja que no descriuen l'acció que duen a terme. Per altra banda, l'screen reader llegeix cada un dels elements d'una carta per separat i per exemple, quan ha d'indicar la valoració de l'establiment únicament diu el nombre que representa la nota sense

comunicar que es tracta de l'avaluació de l'establiment. Per altra banda, l'aplicació no està preparada per a navegar a través d'ella amb TalkBack, ja que en el moment de navegar entre els resultats, salta alguns i no sembla seguir un ordre. Aquesta aplicació tampoc permet canviar els colors del tema de l'aplicació ni modificar la mida del text, tot i que es segueixen bones pràctiques aplicant un contrast correcte en el text i les icones.

10. **Verbositat**. Si, aquesta aplicació informa l'usuari quan es completen totes les accions com per exemple afegir un establiment a preferits, quan no es troben establiments disponibles a la zona …

I finalment s'analitzarà l'app **Foody Bag**.

1. **Registre obligatori per a utilitzar l'app**. No, no és necessari estar registrat per a poder fer servir l'aplicació.
2. **Guia de com funciona l'App**: Únicament inclou 3 presentacions abans del registre explicant les funcionalitats i els objectius principals de l'aplicació
3. **Layout de l'App**. Aquesta aplicació també ofereix una *layout* amb pestanyes. Aquesta aplicació també representa cada una de les ofertes amb un component del tipus targeta el qual permet agrupar les accions i la informació corresponent a l'oferta.
4. **Mapa d'ofertes i establiments**. Si, es permet explorar ofertes i establiments a través d'un mapa, a més a més es permet filtrar els resultats dels marcadors del mapa a través de diversos filtres com, el tipus d'establiment, el tipus d'aliment del lot, rang de preu, rang d'horari…
5. **Llista d'ofertes i establiments**. Sí, des del mapa interactiu es pot accedir a un llistat d'establiments on a més es permet filtrar el tipus d'establiment i ofertes disponibles.
6. **Procés de pagament**. No s'ha pogut avaluar aquest apartat, ja que l'aplicació no permet registrar-se, ja que demana un número de tel. d'Austràlia per a poder fer el registre.
7. **Avaluar l'establiment**: Sí, i a més a més es permet avaluar a l'establiment mitjançant una nota numèrica i un comentari amb fotografies sobre la comanda.
8. **Estadístiques**: El tipus d'estadística visible sobre els establiments és el nombre aproximat de lots venuts a través de l'aplicació, però com que tampoc s'ha pogut iniciar sessió en la plataforma, no es pot avaluar si existeixen més vistes amb estadístiques una vegada s'aconsegueix registrar-se a l'aplicació.
9. **Accessibilitat**: Mitjançant un *screen reader* com TalkBack s'ha pogut observar que l'aplicació segueix ordenadament els elements visibles en cada vista i redacta correctament els labels, els controls i els labels associats a cada control. Tot i això, aquesta aplicació tampoc permet canviar els colors del tema ni modificar la mida de les lletres.
10. **Verbositat**: En les poques accions que es poden dur a terme dins de l'app l'aplicació mostra missatges a l'usuari quan intenta guardar un establiment com a preferit i quan es produeix l'error d'inici de sessió.

7.2. Entrevistes

Entrevista als usuaris:

1. Amb quina freqüència compres menjar que està a punt de caducar a preus reduïts en supermercats?
2. Has utilitzat alguna vegada una aplicació o servei similar per a comprar aliments amb descompte?
3. Quin tipus de problemes o dificultats has experimentat en intentar de reduir el malbaratament d'aliments en la teva vida diària?
4. Quin tipus d'aliments estàs més interessat a comprar a través de la nostra aplicació (per exemple, productes frescos, forn, menjars preparats)? Hi ha algun tipus de menjar que evitaries comprar a través d'una aplicació com la nostra, fins i tot si està a un preu reduït?
5. Quins criteris utilitzes per a decidir si una oferta de menjar a punt de caducar val la pena?
6. Quin tipus d'incentius o recompenses et motivarien a usar la nostra aplicació amb regularitat?
7. Tens alguna preocupació sobre la qualitat o seguretat dels aliments que s'ofereixen en aplicacions com la proposada?
8. Quines característiques consideres més importants en una aplicació d'aquest tipus?
9. Consideraries beneficiós poder gestionar un inventari dels aliments del teu rebost a través de l'aplicació? Utilitzaries aquesta funcionalitat i portaries el rebost al dia?
10. Què t'impulsaria a recomanar l'aplicació a altres persones?

Entrevista als establiments:

1. Quin tipus d'aliments tendeixen a tenir excedents amb més freqüència en el seu establiment?
2. Han utilitzat abans solucions per a reduir el desaprofitament d'aliments? Per què o quines?
3. Com creus que una aplicació com aquesta podria ajudar a millorar la seva eficiència?
4. Com gestiones actualment l'inventari?
5. Quin tipus d'informació li agradaria rebre dels usuaris per a millorar la seva experiència amb l'aplicació?
6. Hi ha alguna preocupació específica que tinguis sobre la col·laboració amb una aplicació d'aquest tipus?

Entrevista perfil tipus usuari:

Nom: Adrià
Edat: 24 anys
Població: Tivissa (Tarragona)
Resideix a: Barcelona en un pis d'estudiants

1. Amb quina freqüència compres menjar que està a punt de caducar a preus reduïts en supermercats?

Habitualment, sempre que vaig a comprar em fixo en aquestes ofertes per estalvi tot i que també soc conscient del malbaratament d'aliments i és una acció a favor de reduir el malbaratament.

2. Has utilitzat alguna vegada una aplicació o servei similar per a comprar aliments amb descompte?

Sí, utilitzo habitualment Too Good To Go quan estic per Barcelona, per Tivissa no l'acostumo a utilitzar a causa de les poques ofertes.

3. Quin tipus de problemes o dificultats has experimentat en intentar de reduir el malbaratament d'aliments en la teva vida diària?

En un pis d'estudiants a vegades és difícil no llençar aliments a la brossa, ja que a vegades si no ens fiquem d'acord en la compra, dues persones fan la compra i ens sobren aliments al pis, majoritàriament passa amb la fruita, la verdura i el pa.

4. Quin tipus d'aliments estàs més interessat a comprar a través de la nostra aplicació (per exemple, productes frescos, forn, menjars preparats)? Hi ha algun tipus de menjar que evitaries comprar a través d'una aplicació com la nostra, fins i tot si està a un preu reduït?

Aliments precuinats i fleca

5. Quins criteris utilitzes per a decidir si una oferta de menjar a punt de caducar val la pena?

Em fixo majoritàriament en la data de caducitat.

6. Quin tipus d'incentius o recompenses et motivarien a usar la nostra aplicació amb regularitat?

M'agradaria rebre de forma periòdica cupons de descompte per aplicar-los a les meves comandes.

7. Tens alguna preocupació sobre la qualitat o seguretat dels aliments que s'ofereixen en aplicacions com la proposada?

No em preocupa, ja que es suposa que en comerços no es pot vendre cap aliment un cop passada la data de caducitat.

8. Quines característiques consideres més importants en una aplicació d'aquest tipus?

El fet de rebre notificacions quan hi ha ofertes noves a prop meu. També m'agradaria explorar les ofertes disponibles com amb Too Good To Go, a través d'un mapa per a veure diferents zones de la ciutat.

9. Consideraries beneficiós poder gestionar un inventari dels aliments del teu rebost a través de l'aplicació? Utilitzaries aquesta funcionalitat i portaries el rebost al dia?

Sí, considero que utilitzaria aquesta funcionalitat si pogués compartir el meu rebost amb els companys de pis per a poder reduir la quantitat d'aliments que llancem a la brossa.

10. Què t'impulsaria a recomanar l'aplicació a altres persones?

Recomanaria l'ús de l'aplicació si un nou usuari que es registra l'aplicació fica el meu usuari com a "referal" i pogués aconseguir descomptes en les comandes. Seria necessari també que tingués suficient varietat d'ofertes disponibles per a poder triar diferents aliments cada dia.

Entrevista perfil tipus usuari:

Nom: Jordi
Població: Móra d'Ebre (Tarragona)
Edat: 42 anys

1. Amb quina freqüència compres menjar que està a punt de caducar a preus reduïts en supermercats?

Sempre que vaig a comprar, dos cops per setmana, normalment sempre productes de carn, fruites i verdures.

2. Has utilitzat alguna vegada una aplicació o servei similar per a comprar aliments amb descompte?

No, no coneixia que existís una aplicació d'aquest tipus

3. Quin tipus de problemes o dificultats has experimentat en intentar de reduir el malbaratament d'aliments en la teva vida diària?

Tinc dues filles i entre el treball, l'escola i les seves activitats extraescolars em queda poc temps i acostumo a fer la compra després de tot el dia de treball i moltes vegades compro articles de més perquè no recordo que tinc a casa i sovint acaben a la brossa.

4. Quin tipus d'aliments estàs més interessat a comprar a través de la nostra aplicació (per exemple, productes frescos, forn, menjars preparats)? Hi ha algun tipus de menjar que evitaries comprar a través d'una aplicació com la nostra, fins i tot si està a un preu reduït?

Estaria interessat en tota mena d'aliments però majoritàriament forn i menjars preparats. No, no evitaria comprar cap classe d'aliment excepte que no tingui un bon descompte

5. Quins criteris utilitzes per a decidir si una oferta de menjar a punt de caducar val la pena?

Pel descompte.

6. Quin tipus d'incentius o recompenses et motivarien a usar la nostra aplicació amb regularitat?

Per exemple, seria atractiu un sistema que cada 5 o 10 compres obtingui un descompte

7. Tens alguna preocupació sobre la qualitat o seguretat dels aliments que s'ofereixen en aplicacions com la proposada?

M'agradaria que existís alguna manera de comparar la qualitat dels lots oferts entre diferents establiments, podent avaluar la qualitat o la quantitat dels lots d'un establiment.

8. Quines característiques consideres més importants en una aplicació d'aquest tipus?

Rebre notificacions sobre noves ofertes i establiments a prop de la meva ubicació.

9. Consideraries beneficiós poder gestionar un inventari dels aliments del teu rebost a través de l'aplicació? Utilitzaries aquesta funcionalitat i portaries el rebost al dia?

Sí, i tant. Portaria al dia el rebost de casa per a saber quins aliments hi ha i quan em caduquen per aprofitar tot el possible i reduir la quantitat d'aliments tirats a la brossa.

10. Què t'impulsaria a recomanar l'aplicació a altres persones?

Que les ofertes disponibles tinguin bons descomptes i si la possibilitat d'afegir els aliments al rebost amb la càmera funciona correctament, si no introduir les compres que faig manualment m'ocuparien massa temps.

Entrevista perfil tipus usuari:

Nom: Rebeca
Població: Reus (Tarragona)
Edat: 39 anys

1. Amb quina freqüència compres menjar que està a punt de caducar a preus reduïts en supermercats?

Sempre que vaig a comprar al supermercat i tota mena d'aliments, ja que puc revisar l'aspecte dels aliments.

2. Has utilitzat alguna vegada una aplicació o servei similar per a comprar aliments amb descompte?

Sí, utilitzo Too Good To Go almenys un cop per setmana.

3. Quin tipus de problemes o dificultats has experimentat en intentar de reduir el malbaratament d'aliments en la teva vida diària?

No tinc cap classe de dificultat. És a dir, no es malbaraten aliments a casa meva, ja que únicament compro el que és necessari.

4. Quin tipus d'aliments estàs més interessat a comprar a través de la nostra aplicació (per exemple, productes frescos, forn, menjars preparats)? Hi ha algun tipus de menjar que evitaries comprar a través d'una aplicació com la nostra, fins i tot si està a un preu reduït?

Fruites i verdures. Però evitaria comprar carn o peix, ja que no puc veure l'estat de l'aliment que compro, en canvi, les fruites i verdures es poden aprofitar força bé.

5. Quins criteris utilitzes per a decidir si una oferta de menjar a punt de caducar val la pena?

Sobretot la data de caducitat per si el puc consumir en aquesta data i l'aspecte.

6. Quin tipus d'incentius o recompenses et motivarien a usar la nostra aplicació amb regularitat?

M'agradaria rebre vals de descomptes

7. Tens alguna preocupació sobre la qualitat o seguretat dels aliments que s'ofereixen en aplicacions com la proposada?

Sí, sobretot no compro carn o peix en aplicacions com aquesta, ja que no puc veure l'aspecte de l'aliment. Tot i que si pogués saber l'opinió mitjançant comentaris d'altres usuaris m'animaria a provar de comprar un altre tipus d'aliments.

8. Quines característiques consideres més importants en una aplicació d'aquest tipus?

Rebre notificacions sobre les últimes ofertes i poder explorar els establiments mitjançant un mapa i poder aplicar-hi filtres. També estaria bé poder emmagatzemar establiments com a preferits i poder visualitzar-los des de la pàgina principal.

9. Consideraries beneficiós poder gestionar un inventari dels aliments del teu rebost a través de l'aplicació? Utilitzaries aquesta funcionalitat i portaries el rebost al dia?

No utilitzaria aquesta funcionalitat, ja que no tinc cap problema per reduir el malbaratament

10. Què t'impulsaria a recomanar l'aplicació a altres persones?

La recomanaria si per cada usuari que recomano l'aplicació i introdueix el meu correu en registrar-se rebo algun cupó de descompte. I si tinc suficients opcions per a poder-la utilitzar freqüentment.

Entrevista perfil tipus establiment

Nom: Silvia
Població: Tivissa (Tarragona)
Edat: 48 anys
Tipus d'establiment: Petit supermercat

1. Quin tipus d'aliments tendeixen a tenir excedents amb més freqüència en el seu establiment?

Sobretot les fruites i verdures, ja que a vegades els clients trien les millors peces i moltes vegades s'han d'acabar llençant a la brossa algunes peces que primerament eren totalment comestibles, però no les compraven per l'aspecte. Alguna vegada també hem de llençar algun dolç de fleca industrial i algun iogurt o postres refrigerats.

2. Han utilitzat abans solucions per a reduir el desaprofitament d'aliments? Per què o quines?

No, no he utilitzat mai cap mena d'aplicació bàsicament per desconeixement que podia existir una aplicació com aquesta.

3. Com creus que una aplicació com aquesta podria ajudar a millorar la seva eficiència?

Seria molt beneficiós per al meu negoci poder salvar aquestes peces de fruita abans que es facin malbé per un preu més reduït i poder crear ofertes quan sigui necessari salvar qualsevol classe d'article.

4. Com gestiones actualment l'inventari, tens en compte el malbaratament d'aliments?

Actualment, utilitzo un programa TPV en un ordinador preparat per a estar a la tenda juntament amb el lector, la impressora i la pantalla. En aquest programa es troben registrats els preus, el nom de l'article i el codi de barres.

5. Quin tipus d'informació li agradaria rebre dels usuaris per a millorar la seva experiència amb l'aplicació?

Sobretot saber si la qualitat dels lots que ofereixo a través de l'app és suficient per als usuaris i també saber si la quantitat d'aliments en el lot és suficient.

6. Hi ha alguna preocupació específica que tinguis sobre la col·laboració amb una aplicació d'aquest tipus?

No tinc cap preocupació excepte que seria necessari saber cada quant de temps es realitzen les transaccions dels pagaments de les comandes venudes al compte de l'empresa. M'agradaria que les transaccions fossin en el moment de la reserva de la comanda.

Entrevista tipus establiment:

Nom: Sandra
Població: Tivissa (Tarragona)
Tipus d'establiment: Forn

1. Quin tipus d'aliments tendeixen a tenir excedents amb més freqüència en el seu establiment?

Normalment, acostumem a vendre tot el pa i quan s'acaba deixem de vendre'n, però encara així alguns dies tenim excedents de pa i algun dolç que no s'aconsegueix vendre.

2. Han utilitzat abans solucions per a reduir el desaprofitament d'aliments? Per què o quines?

No, bàsicament per desconeixement

3. Com creus que una aplicació com aquesta podria ajudar a millorar la seva eficiència?

Encara que siguin pocs els aliments que es malbaraten en aquest establiment el fet de poder vendre'ls a través de l'aplicació ens faria guanyar uns diners extres molt importants.

4. Com gestiones actualment l'inventari, tens en compte el malbaratament d'aliments?

No tinc cap sistema per a controlar l'inventari del negoci, utilitzo una bàscula per a pesar els aliments que van a pes i saber el preu de venda segons el preu/kg que calculo amb anterioritat. També utilitzo la bàscula per a extreure el tiquet per als clients, els productes els fico el preu amb una etiquetadora i els afegeixo manualment el preu a la bàscula per afegir-lo al tiquet. Intento ajustar les comandes als proveïdors per haver de llençar la mínima quantitat possible per almenys es tira algun article a la brossa.

5. Quin tipus d'informació li agradaria rebre dels usuaris per a millorar la seva experiència amb l'aplicació?

M'agradaria saber la qualitat dels aliments que ofereixo són suficients per als usuaris. A més a més de saber si la quantitat és suficient.

6. Hi ha alguna preocupació específica que tinguis sobre la col·laboració amb una aplicació d'aquest tipus?

M'agradaria tenir la certesa que les transaccions es fan en el moment en el qual es fa la reserva del lot.

7.3. Inputs tests amb usuaris (Fase de disseny)

7.3.1. Inputs tests amb usuaris (Perfil client)

Tasca	Inputs positius	Inputs negatius
Registrar-se amb correu i contrasenya	No ha trobat cap mena de dificultat	
Iniciar sessió	No ha trobat cap mena de dificultat	
Cerca un establiment amb el mapa	Ha trobat ràpidament aquesta funció gràcies a la icona de la barra de pestanyes.	
Cerca un establiment amb la llista	Ha fet aquesta acció sense haver-li demanat	Tot i això, ens indica falta informació als components de tipus targeta. Com per exemple l'horari de recollida.
Filtra els resultats	Ha entrat directament als filtres sense haver-li demanat, fet que indica que es veu i s'entén correctament.	
Marca un establiment com a preferit		Li costa trobar el botó per marcar un establiment com a preferit.
Compra un article	No ha trobat cap mena de dificultat.	
Veure últimes comandes		Li costa molt trobar les últimes comandes.
Avalua l'establiment		Primer es confon i espera avaluar les comandes, tot seguit prem sobre el nom de l'establiment i en ser redirigit a la pàgina de l'establiment, troba

Tasca	Inputs positius	Inputs negatius
		correctament com avaluar l'establiment.
Crea un rebost	Afegeix correctament el rebost i a l'haver llegit *l'onboarding* no té cap mena de dificultat per a completar la tasca.	
Afegeix un article al rebost manualment	Troba fàcilment la icona	
Afegeix diversos articles al rebost amb una fotografia	Troba fàcilment la icona per a afegir un element amb una fotografia i entén correctament el procediment de corregir errors.	
Edita un article del rebost	Entén correctament la icona i no té cap problema	
Canviar informació de perfil	No ha trobat cap mena de dificultat	
Canviar contrasenya	No ha trobat cap mena de dificultat	
Canviar aparença	No ha trobat cap mena de dificultat	
Sortir de la sessió	No ha trobat cap mena de dificultat	

7.3.2. Inputs tests amb usuaris (perfil establiments)

Tasca	Inputs positius	Inputs negatius
Registrar-se	No ha trobat cap mena de dificultat	
Iniciar sessió	No ha trobat cap mena de dificultat	
Crea una oferta		Li costa trobar el botó

Tasca	Inputs positius	Inputs negatius
		flotant
Edita una oferta		Primer li costa saber diferenciar que la icona amb el llapis serveix per a editar un element
Veure les comandes	Troba correctament la pestanya gràcies a la icona.	
Crea un rebost	Un cop entesa la finalitat, ho troba força útil. Després d'haver vist com es crea una oferta, no té dificultat per a completar la tasca	L'usuari expressa que hi ha poca informació sobre la finalitat de la pestanya rebost.
Afegeix un article al rebost manualment	No ha trobat cap mena de dificultat.	
Afegeix un article al rebost amb una fotografía	Troba ràpidament com fer la fotografia gràcies a la icona.	No entén la vista següent per a rectificar els errors de l'escaneig del tiquet. Suggereix més informació a la pantalla a través d'alguna icona.
Edita un article del rebost	No ha trobat cap mena de dificultat.	Faltaria alguna opció per a indicar la quantitat restant d'un article al rebost
Canviar informació de perfil	No ha trobat cap mena de dificultat.	
Canviar aparença	No ha trobat cap mena de dificultat.	
Veure comentaris	No ha trobat cap mena de dificultat.	
Sortir de la sessió	No ha trobat cap mena de dificultat.	

7.4. Especificacions casos d'ús

Identificador	**CU-01**
Nom	Explorar establiments i Ofertes
Prioritat	Alta
Descripció	Un usuari pot utilitzar l'aplicació per a explorar establiment i les ofertes disponibles a prop seu, i també podrà veure les avaluacions dels altres usuaris fetes als establiments que estan explorant.
Actors	Client
Precondicions	- Sessió iniciada - Existència d'establiments i ofertes
Iniciat per	-
Flux	1. Cercar establiments a la pàgina principal 2. Cercar establiments amb el mapa 3. Cercar establiments amb la llista 4. Especificar filtres
Post condicions	L'usuari haurà pogut quedar-se igual com estava, comprar una oferta o marcar un establiment com a preferit
Notes	

Identificador	**CU-02**
Nom	Marcar/desmarcar establiment com a preferit
Prioritat	Normal
Descripció	Un usuari client ha de poder marcar i desmarcar un establiment com a preferit
Actors	Client
Precondicions	- Existència d'establiments
Iniciat per	CU-01
Flux	1. Clicar sobre la icona del cor damunt de l'establiment corresponent

Post condicions	- L'usuari té un establiment més a preferits o en té un menys
Notes	

Identificador	**CU-03**
Nom	Comprar una oferta
Prioritat	Alta/Normal/Baixa
Descripció	Mentre un usuari estigui explorant les ofertes ha de poder comprar una oferta
Actors	Client
Precondicions	- Existència d'ofertes
Iniciat per	CU-01
Flux	1. Seleccionar oferta
Post condicions	Condicions finals de l'entorn
Notes	

Identificador	**CU-04**
Nom	Avaluar establiment
Prioritat	Normal
Descripció	Un usuari client ha de poder avaluar els establiments on ha comprat per a poder informar a altres usuaris
Actors	Client
Precondicions	- Sessió iniciada - Haver comprat a l'establiment
Iniciat per	CU-03
Flux	1. Anar a la pàgina de l'establiment 2. Avaluar numèricament l'establiment 3. Opcionalment es pot escriure un comentari
Post condicions	L'establiment té una nova avaluació

	L'establiment rep la notificació
Notes	

Identificador	**CU-05**
Nom	Gestionar Rebosts
Prioritat	Alta
Descripció	Un usuari qualsevol ha de poder crear i gestionar els rebosts
Actors	Client, Establiment
Precondicions	- Sessió iniciada
Iniciat per	-
Flux	1. Crear rebost 2. Modificar rebost 3. Eliminar el rebost
Post condicions	-
Notes	

Identificador	**CU-06**
Nom	Gestionar elements del rebost
Prioritat	Alta
Descripció	Un usuari ha de poder afegir, modificar i eliminar sense cap mena de límit els elements presents en el seu rebost.
Actors	Client, Establiment
Precondicions	- Sessió iniciada - Rebost existent
Iniciat per	CU-05
Flux	1. Afegir element al rebost 2. Editar elements 3. Eliminar elements

Post condicions	-
Notes	

Identificador	**CU-07**
Nom	Afegir elements al rebost amb foto
Prioritat	Alta
Descripció	Un usuari ha de poder afegir múltiples elements al rebost de forma còmoda a partir de la fotografia d'un tiquet.
Actors	Client, Establiment
Precondicions	- Sessió iniciada - Rebost existent - Permisos càmera o permisos de galeria
Iniciat per	CU-06
Flux	1. Afegir elements al rebost amb fotografia. 2. Editar errors de l'escaneig 3. Eliminar possibles errors de l'escaneig
Post condicions	- Rebost amb els elements escanejats afegits
Notes	

Identificador	**CU-08**
Nom	Editar perfil
Prioritat	Normal
Descripció	Un usuari qualsevol ha de poder editar la seva informació de perfil i preferències en qualsevol moment
Actors	Client, Establiment
Precondicions	- Sessió iniciada
Iniciat per	-
Flux	1. Anar a la pestanya de configuració 2. Clicar sobre editar perfil o canviar contrasenya 3. Omplir formulari

	4. Guardar els canvis
Post condicions	- Informació de perfil actualitzada
Notes	

Identificador	**CU-09**
Nom	Canviar aparença de l'app
Prioritat	Baixa
Descripció	Qualsevol usuari ha de tenir l'opció de canviar l'esquema de colors de l'app
Actors	Client, Establiment
Precondicions	- Sessió iniciada
Iniciat per	-
Flux	1. Anar a la pestanya de configuració 2. Clicar sobre canviar aparença 3. Triar esquema de colors 4. Guardar canvis
Post condicions	- Esquema de colors actualitzat
Notes	

Identificador	**CU-10**
Nom	Veure Valoracions
Prioritat	Normal
Descripció	Un establiment ha de poder veure les valoracions i comentaris que els usuaris els fan.
Actors	Establiment
Precondicions	- Sessió iniciada
Iniciat per	
Flux	1. Anar a la pestanya de configuració 2. Clicar sobre la secció de veure avaluacions i comentaris

Post condicions	
Notes	

Identificador	**CU-11**
Nom	Gestionar ofertes
Prioritat	Alta
Descripció	Els establiments han de poder afegir, editar i eliminar les seves ofertes sense cap mena de límit a través de l'app.
Actors	Establiment
Precondicions	- Sessió iniciada
Iniciat per	-
Flux	1. Crear oferta 2. Modificar oferta 3. Eliminar oferta
Post condicions	-
Notes	

7.5. Altres recursos

- Prototip de Figma.
- Documentació en línia de l'API.
- Presentació del projecte en línia.
- Aplicació web resultat del projecte en línia.
- Descàrrega de l'aplicació Android.

Comptes demo de l'aplicació (també pots registrar-te amb un compte nou):

Tipus client

Correu: client1@gmail.com
Contrasenya: 12345678

Tipus establiment

Correu: establiment1@gmail.com
Contrasenya: 12345678