UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Aplicació web per a la gestió de comandes d'un negoci de restauració

Guillem Casassas Bertran

Director: Eloi Puertas Prats

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 27 de Juny de 2018

Resum

En aquest treball de fi de grau, es realitzarà una implementació d'una aplicació web que permetrà a negocis de restauració obrir una nova porta d'interacció amb els clients, facilitant també la realització de comandes i reserves al local.

L'aplicació web s'ha fet sabent que el restaurant que la utilitzaria seria "*l'Impronta Pizza"*, un restaurant amb potencial que no tenia cap espai web en la xarxa. El propietari del negoci, que adoptarà la figura de "*Product Owner"* en el projecte, va estar molt interessat en la proposta del web i no va dubtar en acceptar el projecte i començar a pensar en requeriments que podien facilitar i agilitzar els processos de venda i a l'hora generar més ingressos.

Aquests diferents requeriments seran analitzats i repartits en una planificació, seguint els principis de les metodologies "*Agile*" (concretament en "*Scrum*") on dividirem les tasques en "*Sprints"* per tal de fer un producte entregable funcional a la fi de cada període.

El contingut de l'aplicació es basarà en dues parts: primerament un "*backend"* conformat per una "*API Restful"* en "*Laravel 5.5",* que donarà servei a totes les peticions HTTP realitzades pels usuaris. En segon lloc, un "*frontend"* desenvolupat amb HTML i "*VueJS"* que contindrà totes les vistes i pàgines interactives de l'aplicació web.

Dins del web, també es disposarà d'un panell de gestió per als administradors anomenat "*Housekeeping"* a la documentació. En aquest espai els usuaris administradors podran analitzar els resultats del web i gestionar tots els productes, usuaris, esdeveniments... de l'empresa.

Resumen

En este trabajo de fin de grado, se realizará una implementación de una aplicación web que permitirá a negocios de restauración abrir una nueva puerta de interacción con sus clientes, facilitando también la realización de pedidos y reservas al local.

La aplicación web se ha desarrollado sabiendo que el restaurante que la utilizaría sería "*l'Impronta Pizza"*, un restaurante con potencial que no tenía ningún espacio web en la red. El propietario del negocio, que adoptará la figura de "*Product Owner"* en el proyecto, estuvo muy interesado en la propuesta del web y no dudó en aceptar el proyecto y empezar a pensar en los requerimientos que podrían facilitar y agilizar los procesos de venta y a su vez generar más ingresos.

Estos distintos requerimientos serán analizados y repartidos en una planificación, siguiendo los principios de las metodologías "*Agile*" (concretamente en "*Scrum*") donde dividiremos las tareas en "*Sprints"* con la finalidad de realizar un producto entregable funcional al final de cada periodo.

El contenido de la aplicación se basará en dos partes: primeramente, un "*backend"* formado por una "*API Restful"* en "*Laravel 5.5",* que dará servicio a todas las peticiones HTTP realizadas por los usuarios. En segundo lugar, un "*frontend"* desarrollado con HTML y "*VueJS"* que contendrá todas las vistas y páginas interactivas de la aplicación web.

Dentro de la web, también se dispondrá de un panel de gestión para los administradores llamado "*Housekeeping"* en la documentación. En este espacio los usuarios administradores podrán analizar los resultados de la web y gestionar todos los productos, usuarios, eventos… de la empresa.

Abstract

This final degree project, is about an implementation of a web application that will allow restoration businesses to open a new door of interaction with their clients, also facilitating the making of orders and reservations to the premises.

The web application was developed knowing that the restaurant that would use it would be "*l'Impronta Pizza*", a restaurant with potential that did not have any web space on the Internet. The owner of the business, who will adopt the figure of "*Product Owner*" in the project, was very interested in the proposal of the web and did not hesitate to accept the project and start thinking about the requirements that could facilitate and streamline the sales processes and generate more income.

These different requirements will be analyzed and distributed in a planning, following the principles of the "*Agile*" methodologies (specifically in "*Scrum*") where we will divide the tasks into "*Sprints*" in order to make a functional deliverable product at the end of each period.

The content of the application will be based on two parts: first, a "*backend*" formed by a "*Restful API*" in "*Laravel 5.5*", which will serve all HTTP requests made by users. Second, a "*frontend*" developed with HTML and "*VueJS*" that will contain all the views and interactive pages of the web application.

Within the web, there will also be a management panel for administrators called "*Housekeeping*" in the documentation. In this space, the administrators will be able to analyze the results of the web and manage all the products, users, events ... of the company.

Voldria agrair a la comunitat “*Rimorsoft”* online, tota l’ajuda rebuda i tots els cursos d’aprenentatge que han realitzat al llarg d’aquest projecte.

Agrair al tutor, l’Eloi Puertas Prats tot el temps dedicat i l’orientació rebuda al llarg de tota aquesta etapa.

Agrair també a familiars, amics i companys de la Universitat que també m’han ajudat en el treball i a sobrepassar les etapes difícils d’aquest camí.

Moltes gràcies a tots.

Sumari

1. Introducció

En el món en què vivim avui en dia es tendeix a digitalitzar-ho tot i a estar interconnectat amb la resta de la gent les 24 hores del dia. A poc a poc aquesta digitalització va afectant a més sectors, i un d'ells, el que tractarem en aquest projecte, en pot sortir molt beneficiat. En efecte, parlem del món de la restauració, aquest món en el qual s'ha de brindar al client la millor atenció possible i donar-li una de les millors experiències que mai hagi tingut per tal d'assegurar el seu retorn.

Avui en dia tots els negocis de restauració, i cada cop més, es digitalitzen d'una forma o altra: Ja sigui publicant-se en una pàgina web, o bé atenent als clients en el local amb un dispositiu interconnectat que envia directament les peticions dels clients a la cuina, optimitzant així el procés i un llarg etcètera…

Si tenim en compte que avui en dia aquesta digitalització passa en gran part a causa dels dispositius mòbils, que no falten en gairebé cap butxaca i que el requeriment al qual tendeix aquesta tecnologia és a fer-ho tot adaptable i funcional des del mòbil, arribem a la conclusió que una aplicació mòbil per a totes les plataformes que permeti a un negoci de restauració optimitzar els seus processos pot ser un gran invent. Hem de tenir en compte, però, que el desenvolupament d'un software per a diverses plataformes avui en dia requereix d'un alt nivell de coneixements i temps del qual en gran part no disposem. Però aquest petit inconvenient té una fàcil solució: Una aplicació web.

Amb les tecnologies web que s'han anat generant en aquests últims anys, podem perfectament simular la utilització d'una web com si fos una aplicació mitjançant la col·locació dels elements en la pantalla de forma que s'adaptin al dispositiu des d'on es visualitza. És igual el sistema operatiu del dispositiu, és igual la mesura de la pantalla d'aquest i també és igual el dispositiu en si, la visualització serà adaptada.

Amb aquest avantatge podem augmentar de forma espectacular la fidelitat del client amb l'empresa i alhora permetre a l'empresa generar una gran quantitat de dades significatives que li permetran pivotar el seu negoci d'una manera molt més eficient i productora.

1.1 Objectius

L'objectiu principal d'aquest projecte és posar en pràctica tots els coneixements adquirits al llarg de la carrera, per tal d'implementar un producte útil i eficient que serà immediatament utilitzat per a una empresa.

Com a altres objectius també es voldria assolir un alt nivell d'usabilitat, creant així una interfície agradable per a l'usuari i que a la vegada fos fàcil i entenedora per a no causar confusió als clients. Tot això controlat per un codi modularitzat i escalable per a seguir desenvolupant possibles millores i permetre un fàcil manteniment de l'aplicació.

Finalment, també es vol assolir l'objectiu d'adaptar-se al màxim a la metodologia de treball "*Agile"* per tal d'optimitzar al màxim el temps dedicat en el projecte i convergir així a un producte del qual el "*Product Owner"* se senti satisfet.

1.2 Motivacions

La motivació més important que ha empès a iniciar aquest projecte és el negoci que hi ha darrere: un restaurant amb potencial que no tenia pràcticament cap visibilitat en la xarxa, únicament en les xarxes socials "*Instagram"* i "*Facebook"*. Afegint el fet que al llarg de la carrera ens ha cridat l'atenció sobretot el software distribuït i la computació orientada al web, les ganes de crear una aplicació nova i totalment funcional per a tots els dispositius eren immenses.

L'altra gran motivació és, que al saber que l'aplicació seria utilitzada en acabar el projecte, que l'aplicació estigui al nivell desitjat per al nostre "*Product Owner"* i que aquesta redueixi la quantitat de treball als treballadors del negoci i a l'hora permeti un major grau d'interacció entre l'empresa i els clients. El més important al cap i a la fi, és que sumant tots aquests factors, s'acabi convergint en un ingrés de beneficis per al negoci.

1.3 Estructura de la memòria

A continuació s'explicarà breument el contingut de cada secció i subsecció en la que s'ha dividit la memòria.

1. Introducció: Explicació general, condicions inicials i orígens de tot el projecte.

> **1.1 Objectius:** Fites prèvies que es volen assolir en finalitzar el projecte.
>
> **1.2 Motivació:** Factors i raons per a les quals s'ha iniciat aquest projecte.
>
> **1.3 Estructura de la memòria:** Divisió amb més detall dels diferents apartats de la documentació del projecte.

2. Anàlisis: Estudi dels requeriments bàsics del projecte, agrupació d'aquests en diferents escenaris i contemplació del futur de l'aplicació.

> **2.1 Requeriments bàsics de l'aplicació web:** Necessitats imprescindibles de l'aplicació requerides pel "*Product Owner*".
>
> **2.2 Agrupació dels requeriments i anàlisi casos d'ús**: Agrupació dels diferents requeriments per tal d'analitzar les diferents tasques en conjunt.
>
> **2.3 Futur de l'aplicació:** Reflexió sobre el futur de l'aplicació relacionat amb les fases de testeig i el manteniment del sistema.

3. Disseny: Arquitectura interna del sistema.

> **3.1 Tecnologies utilitzades:** Diferents eines i "*frameworks"* utilitzats per al desenvolupament de l'aplicació.
>
> **3.2 Arquitectura de l'aplicació:** Característiques de disseny, patrons de programació i de disseny i definició d'objectes i el seu emmagatzematge.
>
> **3.3 Diagrames:** Representacions visuals representatives de l'estructura interna de l'aplicació.

4. Implementació: Anàlisis més exhaustiu dels diferents algorismes i funcionalitats principals.

> **4.1 Recomanador:** Anàlisis intern del funcionament del sistema de recomanació de productes de l'aplicació.
>
> **4.2 Sistema de cupons:** Anàlisis intern del funcionament del sistema per a generar i validar cupons de l'aplicació.
>
> **4.3 Generador de pizzes:** Anàlisis intern del funcionament de l'eina per a permetre als usuaris crear les seves pizzes.

5. Resultats: Fase de testeig dels resultats obtinguts.

5.1 Test d'usabilitat: Test realitzat a diferents usuaris de diferents característiques per avaluar la usabilitat de l'aplicació.

5.2 "*Stress Test":* Test per avaluar el funcionament de l'aplicació amb diferents conjunts de dades

6. Planificació: Organització i divisió de les tasques en el temps de desenvolupament del projecte.

6.1 Planificació inicial: Planificació i argumentació de l'organització que s'estima a l'inici del projecte.

6.2 Planificació final: Resultats reals finals comparats amb l'estimació inicial.

6.3 Costos: Avaluació del pressupost total del desenvolupament del projecte.

7. Treball futur: Diverses funcionalitats que poden ser implementades en un futur pròxim.

8. Conclusions: Reflexió del transcurs del projecte, comparant els objectius i motivacions marcats inicialment i avaluant els resultats obtinguts.

9. Bibliografia: Fonts d'informació utilitzades per a l'elaboració del projecte o la memòria.

2. Anàlisis

En aquest apartat d'anàlisis s'observaran els requisits previs demanats per al "*Product Owner*" que seria el client que desitja obtenir l'aplicació. També s'analitzaran els diferents factors externs que tenen importància per l'aplicació, com podria ser el "*target*" d'usuaris objectiu, els grups d'usuaris que utilitzaran l'aplicació, etc... Basant-nos en aquests detalls extraurem conclusions que ens seran útils per a la planificació i el disseny de les funcionalitats.

2.1 Requeriments bàsics de l'aplicació web

L'aplicació web ha de poder ser executada en qualsevol dispositiu i en qualsevol plataforma. Aquest requeriment implicarà un determinat mètode de construcció de la interfície que permeti adaptar el contingut a la pantalla del dispositiu des d'on es visualitza el web.

D'altra banda, donat que l'aplicació web està orientada a un públic contingut en un ampli rang d'edat, s'haurà d'assegurar amb alta importància que la usabilitat i l'eficiència de les diferents funcionalitats és la requerida per a poder dur a terme qualsevol acció el més ràpid possible sense fer dubtar a l'usuari que l'utilitza.

Donats els requeriments del *"Product Owner"*, que els veurem en el següent apartat, s'hauran d'implementar les funcionalitats d'una manera entenedora per a l'usuari. Per a posar a prova les interfícies realitzades, es duran a terme diferents testos per avaluar la usabilitat de les diferents interfícies i també la seva eficiència. D'aquesta manera rebrem un "*feedback*" directe dels usuaris objectius, que serà àmpliament analitzat per tal d'extreure conclusions objectives que ens ajudin a millorar la usabilitat i l'experiència de l'usuari en entrar a la pàgina web.

Un altre requeriment important per a l'aplicació, és que ha de ser segura, en el sentit d'assegurar tant a l'usuari com a l'empresa que la utilitzarà, que totes les accions que s'estan realitzant i les dades que s'estan compartint, no seran malmeses en qualsevol moment. S'hauran de codificar processos en els quals els resultats equivaldran a una quantitat determinada de diners. Sobretot en aquests processos caldrà comprovar les dades utilitzades per a assegurar que els resultats de les funcionalitats implementades són els correctes.

Al tractar-se d'una aplicació web orientada al món de la restauració i a causar bona impressió al client per a que aquest acabi comprant els productes, serà necessària una interfície clara i agradable per a l'usuari. S'hauran d'evitar confusions, textos o imatges petites o qualsevol element o procés que pugui causar incertesa a l'usuari. Com a ajuda a aquest problema, haurem de tenir en compte els 10 principis d'usabilitat de *Nielsen* i els diferents patrons de disseny d'interfícies que veurem en el disseny de l'aplicació.

A part de ser una aplicació web destinada als clients, una certa part d'aquesta també va destinada a l'equip administratiu del restaurant, perquè puguin gestionar tots els productes, comandes, reserves, esdeveniments, etc... més fàcilment. S'haurà de crear un accés personalitzat a aquest espai, i contemplar l'opció que els usuaris podran adoptar rols. Amb tres rols ens seria suficient per a cobrir els requeriments previs:

- Rol d'usuari convidat
- Rol d'usuari registrat
- Rol d'usuari administrador

2.2 Agrupació dels requeriments i anàlisis dels casos d'ús

Atès que el nostre *"Product Owner"* necessita cobrir unes certes necessitats bàsiques amb l'aplicació, podrem extreure benefici en el sentit d'organització a l'hora de desenvolupar les diferents funcionalitats que ens requereix. Aquest conjunt de requeriments els ordenarem en el nostre "*Backlog"*, els hi donarem una estimació de temps que ens ocuparà i realitzarem una planificació.

Les diferents *"user stories"* requerides per al nostre "*Product Owner"*, són petites funcionalitats, que dividirem el màxim possible fins a fer-les atòmiques, que per una raó determinada han de ser contemplades i dutes a terme. El fet de fer-les atòmiques ens ajudarà a estimar el temps que ens ocuparà i això convergirà en una millor planificació.

Aquestes funcionalitats atomitzades, les dividirem en "*Sprints*" per tal de millorar la nostra organització i poder anar mostrant els resultats obtinguts al "*Product Owner"*. D'aquesta manera obtindrem un "*feedback"* directe que ens serà totalment útil a l'hora d'implementar futures funcionalitats.

La divisió i realització de les tasques serà analitzada en l'apartat de planificació. En aquest apartat englobarem i escenificarem totes les "*user stories*" mencionades anteriorment amb la finalitat d'analitzar-les i concretar les condicions i fluxos d'execució de cada una de les funcionalitats.

Per a fer-ho realitzarem el traspàs de cada "*user story"* requerida pel nostre "*Product Owner"* a un diagrama de casos d'ús i posteriorment analitzarem les funcionalitats del diagrama una per una.

Per tal de realitzar un bon anàlisi, s'han dividit les "user stories" més importants en quatre escenaris:

- **Escenari del panell d'administració**: En aquest escenari trobarem totes les funcionalitats associades als usuaris administradors de l'aplicació.

- **Escenari de la carta:** En aquest escenari trobarem totes les funcionalitats associades als usuaris normals de l'aplicació, que podran realitzar quan aquests es trobin en la carta.

- **Escenari de l'usuari:** En aquest escenari trobarem totes les funcionalitats associades als usuaris normals de l'aplicació, que podran realitzar quan aquests es trobin en la seva pàgina de perfil.

- **Escenaris separats**: En aquest conjunt trobarem els casos d'ús que realitzen funcionalitats concretes i que no pertanyen a cap dels grups anteriors.

2.2.1 Escenari del panell d'administració

A continuació observarem les diferents històries d'usuari o "*user stories*" que són les funcionalitats requerides per als usuaris administradors per tal de gestionar tot el contingut de la web dinàmicament.

USER STORY 1.- COM A USUARI ADMINISTRADOR VULL PODER GESTIONAR ELS PRODUCTES, INGREDIENTS, ESDEVENIMENTS I USUARIS DE L'APLICACIÓ.

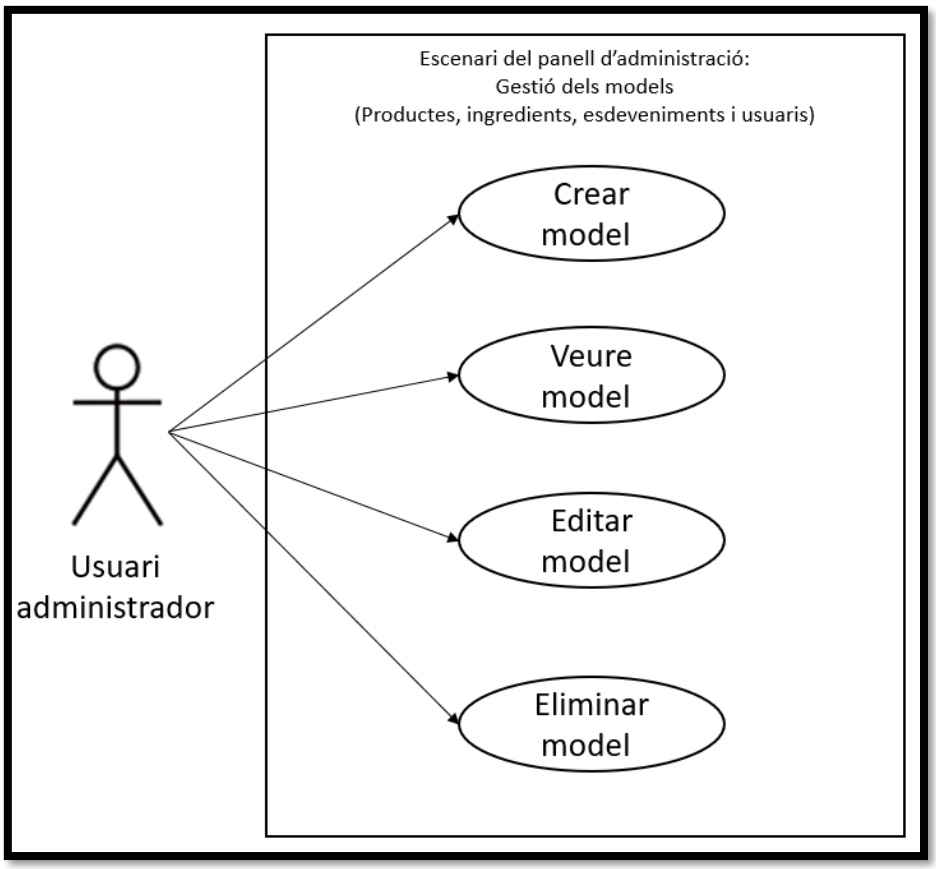


Figura 1: *Diagrama de casos d'ús per a la gestió de productes, ingredients, esdeveniments i usuaris.*

Per a no haver de crear un grup de diagrames per a cada model (producte, ingredient, esdeveniment i usuari), s'han agrupat tots en un i es fa referència amb la paraula model.

NOM	**UC1 – CREAR MODEL**
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a afegir un nou objecte d'un model al sistema.
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat del model en qüestió.
FLUX BÀSIC	1. L'aplicació mostra el formulari amb les dades necessàries per a crear un nou objecte del model. El formulari dependrà del model que s'estigui creant. 2. L'usuari omple les dades per a crear l'objecte. 3. L'usuari envia les dades al servidor. 4. El servidor comprova les dades i crea l'objecte del model.
FLUX ALTERNATIU	1.1. L'usuari introdueix les dades en un format invàlid. 1.2. El sistema atura la creació de l'objecte i mostra un missatge d'error a l'usuari.
RESULTATS	L'usuari té la sessió iniciada i se situa a la pantalla del llistat del model en el qual es trobi.

Figura 2: *Taula per analitzar el cas d'ús 1: Crear model.*

NOM	**UC2 – VEURE MODEL**
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a visualitzar la informació del conjunt d'un model del sistema.
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "housekee*eping*", a l'apartat del model en qüestió.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen tots els objectes del model i les seves característiques
FLUX ALTERNATIU	1.1 L'usuari no té connexió a internet, i no pot obtenir la informació dels models de la base de dades. 1.2 El sistema detecta l'error d'obtenció de dades i informa a l'usuari.
RESULTATS	L'usuari pot visualitzar fàcilment el conjunt de característiques de tots els models.

Figura 3: *Taula per analitzar el cas d'ús 2: Veure model.*

NOM	**UC3 – EDITAR MODEL**
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a editar la informació d'un model del sistema.
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat del model que desitja editar.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen tots els objectes del model i les seves característiques 2. L'usuari prem el botó per editar un determinat model i el sistema mostra la finestra flotant per a l'edició, omplint-lo amb les dades actuals d'aquell objecte del determinat model. 3. L'usuari edita la informació que li convé i guarda els canvis.
FLUX ALTERNATIU	1.3 L'usuari introdueix un format erroni a l'editar l'objecte del model. 1.4 El sistema mostra un missatge d'error i no realitza l'actualització de l'objecte.
RESULTATS	El sistema torna a llistar altra vegada les dades actualitzades i es mostra un missatge informant que l'acció s'ha realitzat correctament

Figura 4: *Taula per analitzar el cas d'ús 3: Editar model.*

NOM	**UC4 – ELIMINAR MODEL**
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a eliminar un objecte d'un model del sistema.
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat del model que desitja eliminar.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen tots els objectes del model i les seves característiques 2. L'usuari prem el botó per eliminar un objecte del model.
FLUX ALTERNATIU	2.1 L'usuari no té connexió a internet a l'hora d'eliminar l'objecte. 2.2 El sistema no aconseguirà enviar la petició i mostrarà un missatge d'error a l'usuari.
RESULTATS	El sistema torna a llistar altra vegada les dades actualitzades i es mostra un missatge informant que l'acció s'ha realitzat correctament

Figura 5: *Taula per analitzar el cas d'ús 4: Eliminar model.*

USER STORY 2.- COM A USUARI ADMINISTRADOR VULL PODER GESTIONAR LES COMANDES I LES RESERVES I INFORMAR A L'USUARI EN TEMPS REAL DEL SEU ESTAT.

***REQUERIMENT: VULL PODER VEURE ELS INGREDIENTS QUE ELS USUARIS HAN AFEGIT/TRET EN UN PRODUCTE D'UNA COMANDA.**

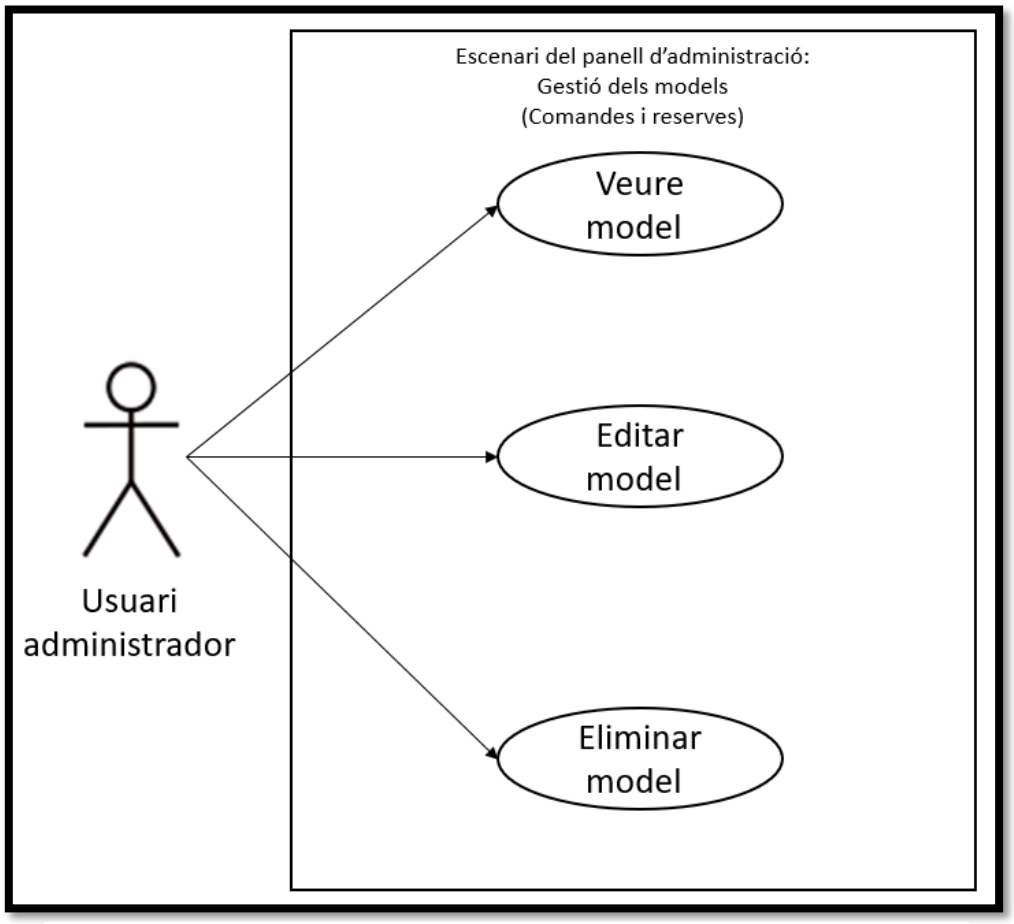


Figura 6: *Diagrama de casos d'ús per a la gestió de reserves i comandes.*

Degut al requeriment del "*Product Owner*", diferenciarem la visualització entre les reserves i les comandes.

NOM	**UC5 – VEURE RESERVA**
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a visualitzar una reserva del sistema
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat de reserves.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen totes les reserves juntament amb totes les dades necessàries.
FLUX ALTERNATIU	1.1 L'usuari no té internet per a obtenir les reserves. 1.2 El sistema mostra un missatge d'error informant que no s'han pogut obtenir les reserves.
RESULTATS	L'usuari veu clarament les diferents reserves ordenades per dies i hora de realització.

Figura 7: *Taula per analitzar el cas d'ús 5: Veure reserva.*

NOM	UC6 – VEURE COMANDA
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a visualitzar una comanda del sistema
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat de comandes.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen totes les comandes juntament amb totes les dades necessàries. 2. L'usuari prem el botó per a visualitzar una comanda en concret 3. L'aplicació mostra una finestra flotant amb tots els productes que formen la comanda. 4. L'usuari pot prémer el desplegable que conté cada producte, per a observar els ingredients que porta.
FLUX ALTERNATIU	1.1 L'usuari no té internet per a obtenir les reserves. 1.2 El sistema mostra un missatge d'error informant que no s'han pogut obtenir les reserves.
RESULTATS	L'usuari veu clarament les diferents comandes ordenades per dies i hora de realització i pot saber en qualsevol moment el contingut de cadascuna d'elles.

Figura 8: *Taula per analitzar el cas d'ús 6: Veure comanda.*

NOM	UC7 – EDITAR RESERVA O COMANDA
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a editar una reserva o una comanda del sistema
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat de reserves o comandes.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen tots els objectes del model (reserves o comandes) 2. L'usuari prem el botó per editar un determinat objecte de reserva o comanda 3. L'aplicació mostra una finestra flotant amb la informació de l'objecte escollit. 4. L'usuari realitza els canvis desitjats i guarda els resultats.
FLUX ALTERNATIU	1.1 L'usuari no té internet per a obtenir el llistat d'objectes del model. 1.2 El sistema mostra un missatge informant de l'error a l'usuari. 2.1 L'usuari edita la reserva o comanda sense necessitat d'obrir la finestra flotant i es faria un salt directe al punt 4 del flux bàsic 4.1 L'usuari modifica una dada amb un format no vàlid. 4.2 El sistema mostra un missatge informant de l'error a l'usuari.
RESULTATS	L'usuari veu el llistat d'objectes amb les dades actualitzades i es mostra un missatge informant que l'acció s'ha realitzat correctament

Figura 9: *Taula per analitzar el cas d'ús 7: Editar reserva o comanda.*

NOM	UC8 – ELIMINAR RESERVA O COMANDA
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a eliminar una reserva o una comanda del sistema
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat de reserves o comandes.
FLUX BÀSIC	1. L'aplicació mostra una taula on s'observen tots els objectes del model (reserves o comandes) 2. L'usuari prem el botó per eliminar un determinat objecte de reserva o comanda
FLUX ALTERNATIU	1.1 L'usuari no té internet per a obtenir el llistat d'objectes del model. 1.2 El sistema mostra un missatge informant de l'error a l'usuari.
RESULTATS	L'usuari veu el llistat d'objectes actualitzats i es mostra un missatge informant que l'acció s'ha realitzat correctament

Figura 10: *Taula per analitzar el cas d'ús 8: Eliminar reserva o comanda.*

USER STORY 3.- COM A USUARI ADMINISTRADOR VULL PODER VALIDAR ELS CUPONS DE PUNTS GENERATS PER ELS USUARIS.

***REQUERIMENT: VULL PODER VEURE LA INFORMACIÓ DE L'USUARI QUE BESCANVIA EL CUPÓ.**

Aquest escenari no pertany únicament a l'administració, ja que també intervé un client. S'han agrupat en un sol diagrama per a aclarir la implicació obligatòria que té l'un de l'altre.

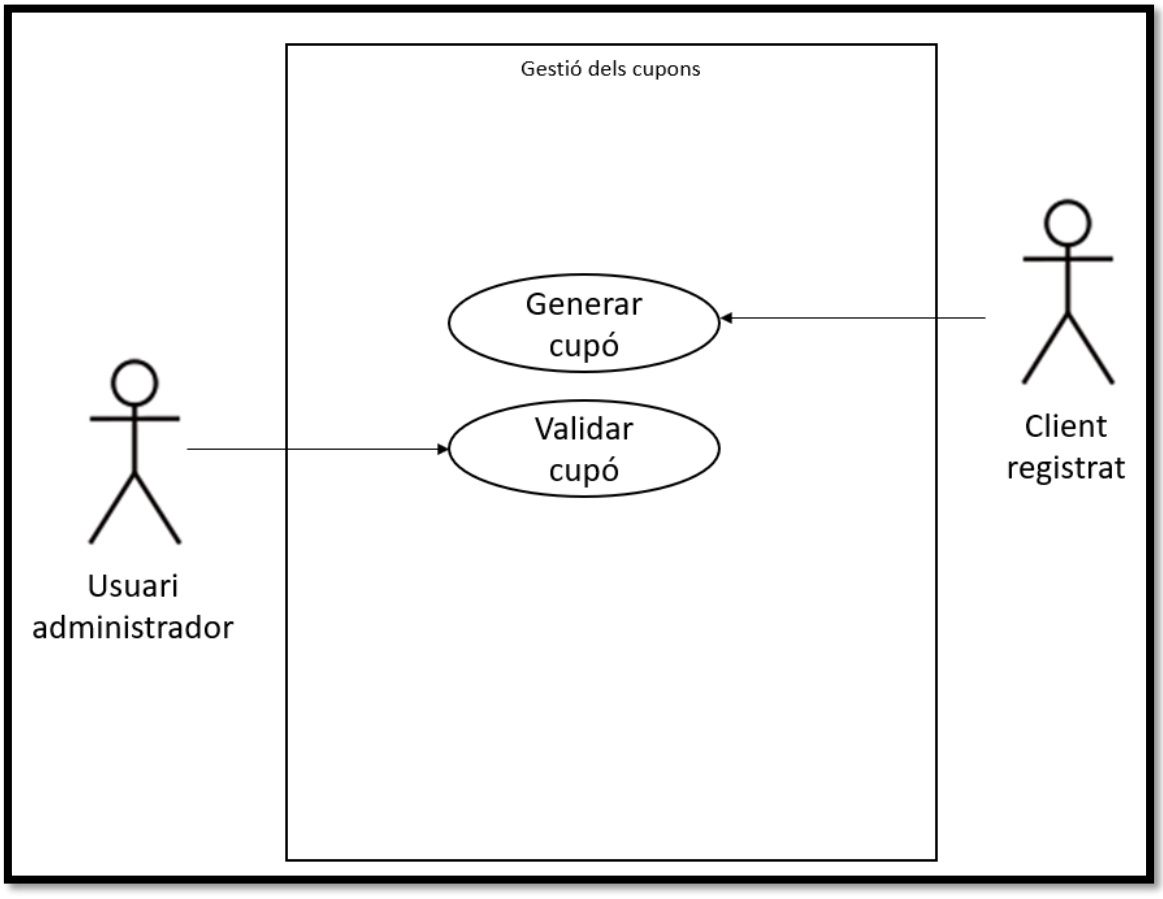


Figura 11: *Diagrama de casos d'ús per a la gestió dels cupons.*

NOM	UC9 – GENERAR CUPÓ
ACTOR	Client registrat
DESCRIPCIÓ	Procés per a generar un cupó a l'aplicació
PRE CONDICIONS	El client té un compte a l'aplicació, el qual està confirmat i té una quantitat de punts superior a 0.
FLUX BÀSIC	1. El client es dirigeix a la pàgina per a bescanviar els punts. 2. El client introdueix la quantitat desitjada de punts a bescanviar. 3. El sistema genera el cupó i li retorna el codi pertinent
FLUX ALTERNATIU	2.1 El client introdueix una quantitat invàlida de punts. 2.2 El sistema no genera el cupó i mostra un missatge d'error informant a l'usuari.
RESULTATS	El client obté el cupó, el qual podrà bescanviar simplement dirigint-se a un administrador del local i dictant-li el codi del cupó pertinent.

Figura 12: *Taula per analitzar el cas d'ús 9: Generar cupó.*

NOM	UC10 – VALIDAR CUPÓ
ACTOR	Usuari administrador
DESCRIPCIÓ	Procés per a validar un cupó generat per un usuari
PRE CONDICIONS	L'usuari és administrador i es troba al panell de control o "*housekeeping*", a l'apartat de "*Vouchers"*. El client ja ha generat un cupó per a bescanviar.
FLUX BÀSIC	1. L'aplicació mostra un "*input"* on l'usuari administrador introduirà el codi 2. L'usuari administrador introdueix el codi del cupó que li ha dit el client que desitja bescanviar els punts 3. El sistema reconeix el cupó, mostra les dades del client i del cupó per pantalla i fa visible el botó per a efectuar el bescanvi. 4. L'usuari administrador prem el botó per a efectuar el bescanvi. 5. El servidor fa les comprovacions necessàries per a assegurar la validesa de la transacció. 6. El sistema resta els punts bescanviats al client.
FLUX ALTERNATIU	1.1 L'aplicació no disposa d'internet i no pot obtenir els cupons de l'aplicació. 1.2 S'informa a l'usuari administrador per pantalla amb un missatge d'error. 3.1 El sistema no reconeix el cupó perquè no existeix, o bé perquè està caducat. 3.2 El sistema mostra un missatge informant de l'error 5.1 El sistema detecta una incongruència en la transacció. 5.2 Es mostra un missatge d'error informant a l'usuari administrador de l'error ocorregut.
RESULTATS	El client realitza el bescanvi i se li resten els punts del seu compte en l'aplicació. A canvi obté el/s producte/s bescanviats.

Figura 13: *Taula per analitzar el cas d'ús 10: Validar cupó.*

2.2.2 Escenari de la carta

A continuació observarem les diferents històries d'usuari o "*user stories"* que són les funcionalitats requerides per als usuaris de l'aplicació que es realitzaran en l'entorn de la carta.

USER STORY 4.- COM A USUARI DE L'APLICACIÓ VULL PODER VEURE TOTS ELS PRODUCTES DE LA CARTA, MARCAR-LOS COM A PREFERITS, DEIXAR COMENTARIS I PODER CREAR LA MEVA COMANDA PERSONALITZADA.

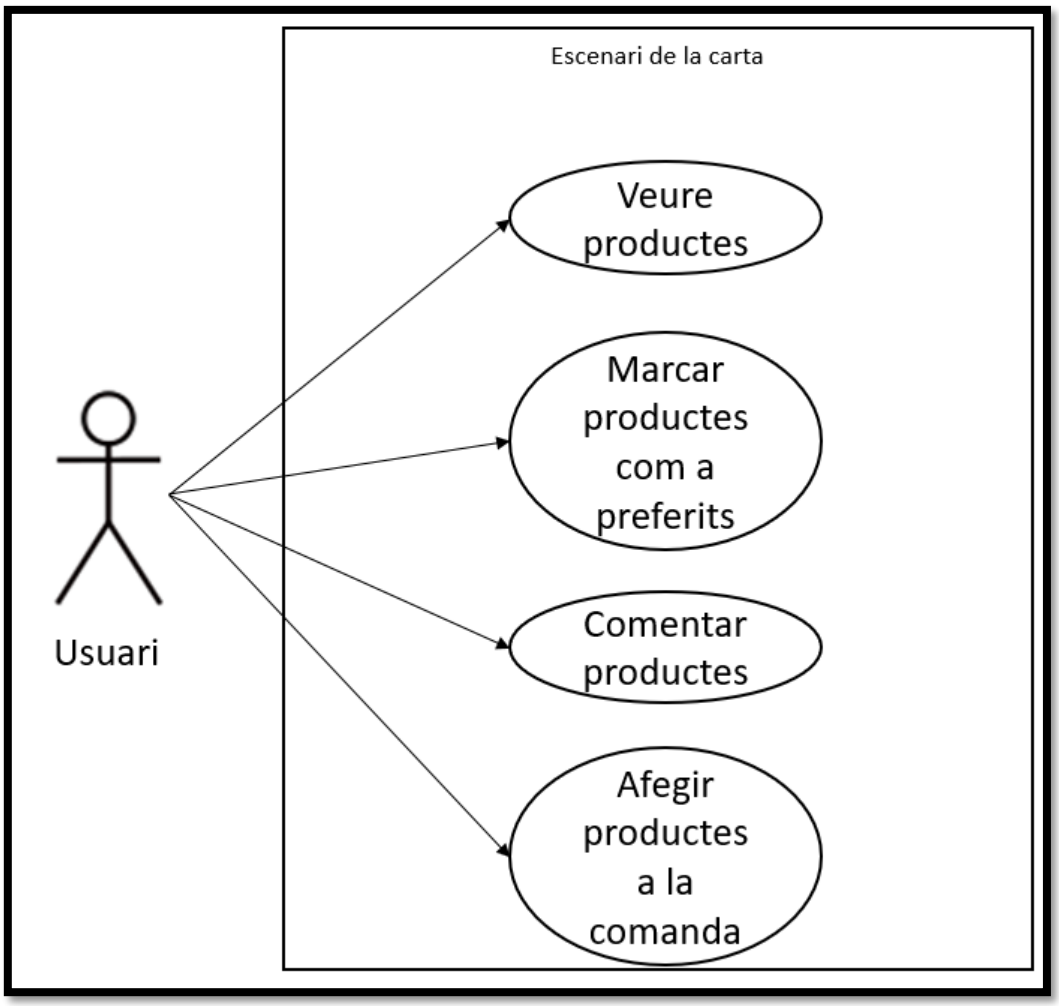


Figura 14: *Diagrama de casos d'ús per a les funcionalitats d'usuari en la carta.*

NOM	UC11 – VEURE PRODUCTES
ACTOR	Usuari
DESCRIPCIÓ	Procés per a veure tots els productes que ofereix l'empresa.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina de la carta.
FLUX BÀSIC	1. L'aplicació mostra tot el conjunt de productes dividits segons el tipus. 2. L'usuari navega per les pestanyes de la carta per anar veient els diferents productes que s'ofereixen 3. L'aplicació actualitza el contingut de la carta depenent de la pestanya en la qual es troba l'usuari
FLUX ALTERNATIU	1.1 L'aplicació no disposa d'internet per obtenir els productes. 1.2 Es mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari pot visualitzar dinàmicament i de forma agradable tots els productes que s'ofereixen

Figura 15: *Taula per analitzar el cas d'ús 11: Veure producte.*

NOM	**UC12 – MARCAR PRODUCTE COM A PREFERIT**
ACTOR	Usuari
DESCRIPCIÓ	Procés per a marcar un producte com a preferit.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina de la carta.
FLUX BÀSIC	1. L'aplicació mostra tot el conjunt de productes dividits segons el tipus. 2. L'usuari prem el botó de "m'agrada" en un producte
FLUX ALTERNATIU	1.1 L'aplicació no disposa d'internet per obtenir els productes. 1.2 Es mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari pot visualitzar en qualsevol moment els seus productes preferits d'una forma més ràpida respecte a la resta de productes.

Figura 16: *Taula per analitzar el cas d'ús 12: Marcar producte com a preferit.*

NOM	UC13 – COMENTAR PRODUCTE
ACTOR	Usuari
DESCRIPCIÓ	Procés per a deixar un comentari a un producte de la carta.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina de la carta.
FLUX BÀSIC	1. L'aplicació mostra tot el conjunt de productes dividits segons el tipus. 2. L'usuari omple el camp de text amb el comentari que desitja publicar. 3. L'usuari prem el botó per a publicar el comentari.
FLUX ALTERNATIU	1.1 L'aplicació no disposa d'internet per obtenir els productes. 1.2 Es mostra un missatge d'error informant a l'usuari. 1.1 L'usuari intenta escriure un text molt llarg que pot posar en perill la visualització de la pàgina. 1.2 El sistema limita la longitud del comentari a 120 caràcters.
RESULTATS	L'usuari veu el comentari realitzat en el producte.

Figura 17: *Taula per analitzar el cas d'ús 13: Comentar producte.*

NOM	UC14 – AFEGIR PRODUCTE A LA COMANDA
ACTOR	Usuari
DESCRIPCIÓ	Procés per a deixar un comentari a un producte de la carta.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina de la carta.
FLUX BÀSIC	1. L'aplicació mostra tot el conjunt de productes dividits segons el tipus. 2. L'usuari prem el botó d'afegir a la comanda d'un determinat producte. 3. El sistema mostra una finestra flotant preguntant a l'usuari la quantitat desitjada i en el cas de les pizzes, el tamany desitjat. 4. L'usuari introdueix les dades i afegeix el producte a la comanda.
FLUX ALTERNATIU	4.1 L'usuari introdueix dades invàlides 4.2 El sistema mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari afegeix a la comanda la quantitat del producte indicada.

Figura 18: *Taula per analitzar el cas d'ús 14: Afegir producte a la comanda.*

2.2.3 Escenari de l'usuari

A continuació observarem les diferents històries d'usuari o "*user stories*" que són les funcionalitats requerides per als usuaris de l'aplicació que es realitzaran en l'entorn de la carta.

USER STORY 5.- COM A USUARI DE L'APLICACIÓ VULL PODER CREAR LES MEVES PRÒPIES PIZZES, INDICAR QUINS INGREDIENTS NO M'AGRADEN, VEURE ELS MEUS PRODUCTES PREFERITS I GENERAR RESERVES.

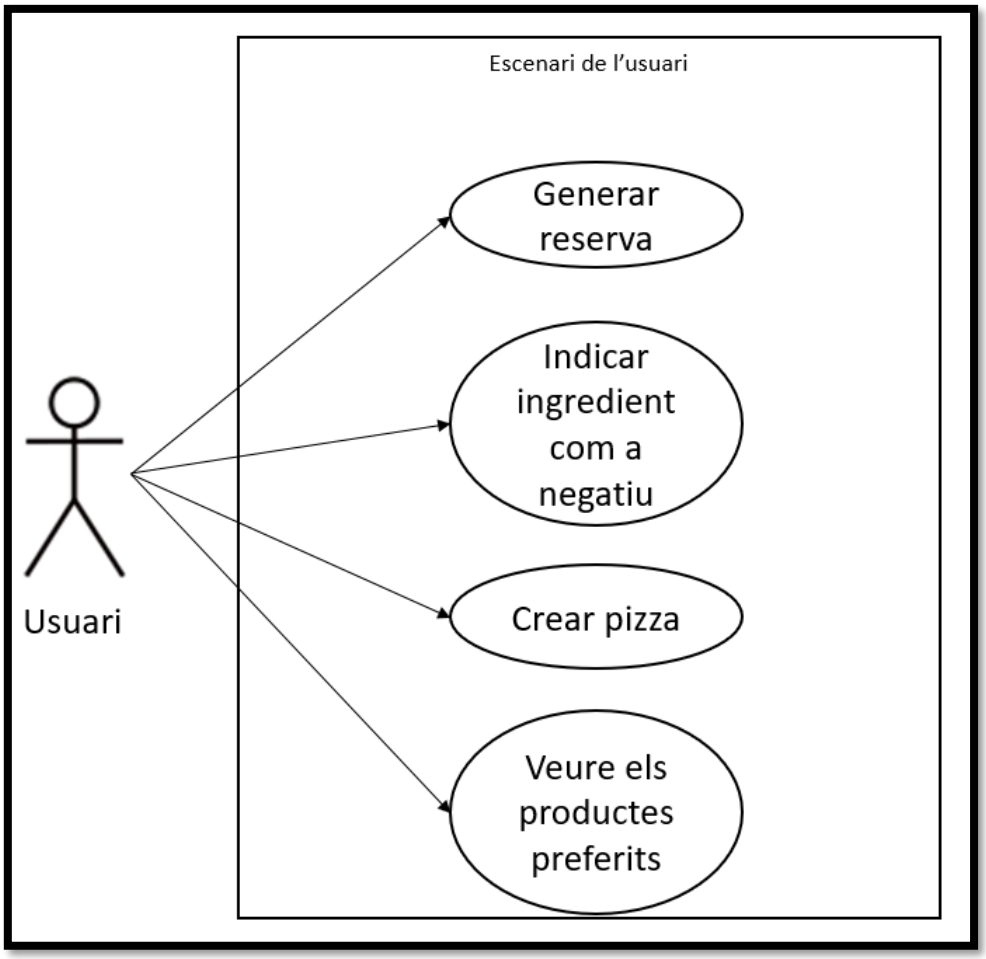


Figura 19: *Diagrama de casos d'ús per a les funcionalitats de la pàgina de perfil de l'usuari.*

NOM	UC15 – GENERAR RESERVA
ACTOR	Usuari
DESCRIPCIÓ	Procés per a generar una nova reserva en el sistema.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, a l'apartat de les seves reserves.
FLUX BÀSIC	1. L'aplicació mostra totes les seves reserves futures i un botó per a crear-ne de noves. 2. L'usuari prem el botó per crear-ne una de nova. 3. El sistema obra una finestra flotant amb les dades necessàries per generar una nova reserva. 4. L'usuari omple les dades i genera la reserva.
FLUX ALTERNATIU	4.3 L'usuari introdueix dades invàlides 4.4 El sistema mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari genera una reserva, fet que fa que als usuaris administradors els hi arribi una notificació i atenguin la reserva. L'usuari rebrà una resposta sabent si la seva reserva ha estat confirmada.

Figura 20: *Taula per analitzar el cas d'ús 15: Generar reserva.*

NOM	UC16 – INDICAR INGREDIENT COM A NEGATIU
ACTOR	Usuari
DESCRIPCIÓ	Procés per a indicar un ingredient com a negatiu.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, a l'apartat de "No m'agrada".
FLUX BÀSIC	1. L'aplicació mostra tots els ingredients actualment marcats com a negatius i un "*input*" per a introduir-ne de nous. 2. L'usuari introdueix el nom de l'ingredient que vol marcar com a negatiu. 3. El sistema detecta aquell ingredient i l'afegeix com a ingredients negatius d'aquell usuari
FLUX ALTERNATIU	2.1 El sistema completa el nom de l'ingredient amb la tecnologia d'auto completat, d'aquesta forma assegurem la validesa de la dada d'entrada.
RESULTATS	S'afegeix l'ingredient a la llista d'ingredients negatius de l'usuari. Els productes amb què continguin aquest ingredient seran visualitzats d'una forma especial per aquell usuari al navegar per la carta.

Figura 21: *Taula per analitzar el cas d'ús 16: Indicar ingredient com a negatiu.*

NOM	UC17 – CREAR PIZZA
ACTOR	Usuari
DESCRIPCIÓ	Procés per a crear una nova pizza.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, a l'apartat de "Les meves pizzes".
FLUX BÀSIC	1. L'aplicació mostra totes les pizzes actualment creades per l'usuari i un botó per a crear-ne de noves. 2. L'usuari prem el botó per crear-ne una de nova. 3. L'aplicació obra una finestra flotant amb les dades necessàries per crear una nova pizza. 4. L'usuari omple les dades necessàries i genera la pizza.
FLUX ALTERNATIU	3.1 L'usuari completa les dades de la nova pizza amb un format invàlid. 3.2 El sistema atura la creació de la pizza i mostra un missatge d'error informant a l'usuari
RESULTATS	Es crea la pizza per a l'usuari. En cas d'haver-la configurat com a pública, tota la comunitat podrà veure-la, marcar-la com a preferida i demanar-la.

Figura 22: *Taula per analitzar el cas d'ús 17: Crear una nova pizza personalitzada.*

NOM	UC18 – VEURE PRODUCTES PREFERITS
ACTOR	Usuari
DESCRIPCIÓ	Procés per a visualitzar els productes preferits de l'usuari
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, a l'apartat de "Preferits".
FLUX BÀSIC	1. L'aplicació mostra tots els productes preferits marcats per l'usuari, juntament amb els ingredients de cadascun.
FLUX ALTERNATIU	1.1 L'aplicació no té connexió a internet per a obtenir els productes preferits de l'usuari. 1.2 El sistema mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari té un ràpid accés als seus productes preferits.

Figura 23: *Taula per analitzar el cas d'ús 18: Veure productes preferits d'un usuari.*

USER STORY 6.- COM A USUARI DE L'APLICACIÓ VULL PODER CONFIGURAR LES MEVES DADES. COM EL CORREU ELECTRÒNIC, LA IMATGE DE PERFIL I LA DIRECCIÓ DE L'ENVIAMENT DE LES COMANDES PER DEFECTE.

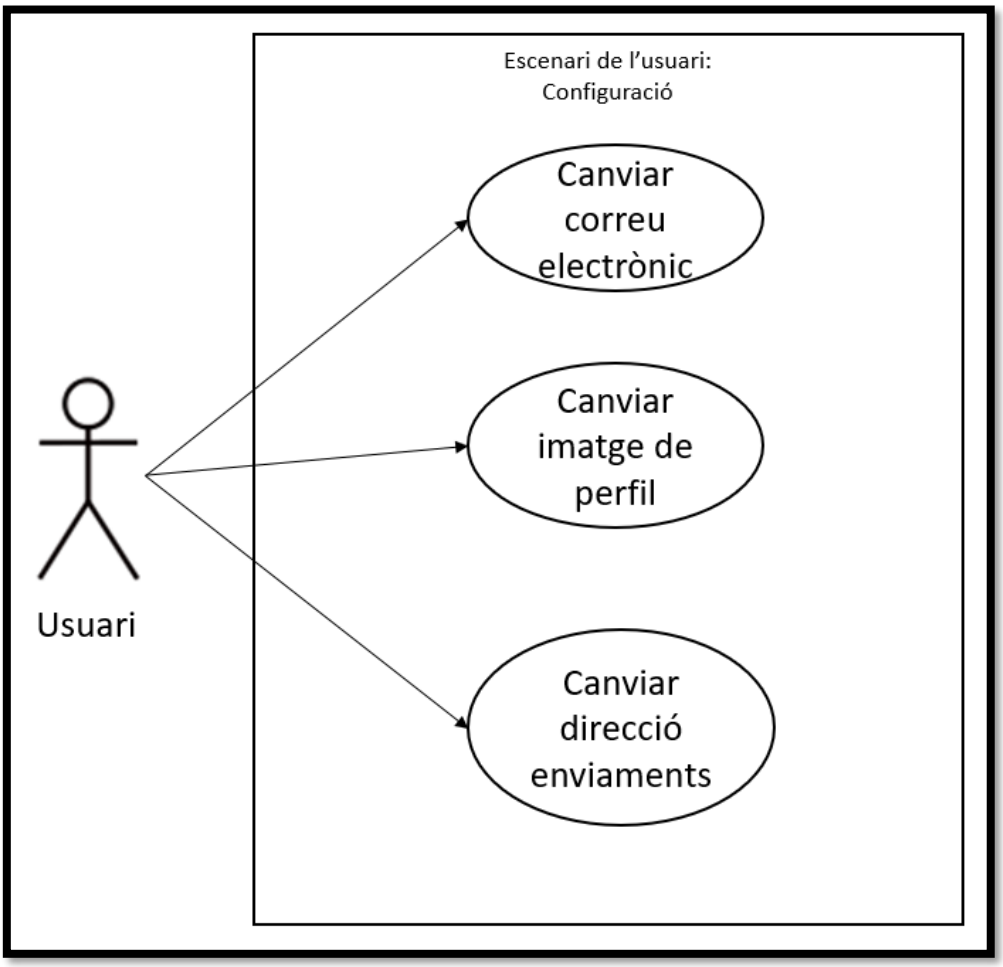


Figura 24: *Diagrama de casos d'ús per a les funcionalitats de configuració de la pàgina de perfil de l'usuari.*

NOM	**UC19 – CANVIAR CORREU ELECTRÒNIC**
ACTOR	Usuari
DESCRIPCIÓ	Procés per a canviar el correu electrònic de l'usuari
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, dins de la configuració a l'apartat de "Canviar correu electrònic".
FLUX BÀSIC	1. L'usuari introdueix la seva adreça de correu electrònic actual. 2. El sistema li envia un correu amb un enllaç a la pàgina per a actualitzar el nou correu. 3. L'usuari es dirigeix al correu, visita l'enllaç que se li proporciona i omple les dades amb el nou correu electrònic. 4. El sistema revisa les dades i realitza el canvi.
FLUX ALTERNATIU	1.1 El sistema no detecta el correu actual de l'usuari. 1.2 El sistema mostra un missatge d'error informant a l'usuari. 3.1 L'usuari introdueix les dades del nou correu electrònic amb un format invàlid. 3.2 El sistema atura el canvi de correu i mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari ha configurat un nou correu electrònic associat al compte. Tots els correus automàtics de l'aplicació i l'inici de sessió estaran associats a aquest nou compte.

Figura 25: *Taula per analitzar el cas d'ús 19: Canviar correu electrònic.*

NOM	**UC20 – CANVIAR IMATGE DE PERFIL**
ACTOR	Usuari
DESCRIPCIÓ	Procés per a canviar la imatge de perfil d'un usuari
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, dins de la configuració a l'apartat de "Canviar imatge de perfil".
FLUX BÀSIC	1. L'usuari selecciona una imatge de disc. 2. El sistema mostra el botó per a realitzar el canvi d'imatge. 3. L'usuari prem el botó per al canvi d'imatge. 4. El sistema realitza el canvi i recarrega la pàgina amb les noves dades.
FLUX ALTERNATIU	4.1 El sistema detecta una incongruència en la imatge 4.2 S'atura el canvi de imatge, es manté la que l'usuari tenia prèviament i es retorna mostrant un missatge d'error a l'usuari.
RESULTATS	S'actualitza la imatge de perfil de l'usuari.

Figura 26: *Taula per analitzar el cas d'ús 20: Canviar imatge de perfil.*

NOM	**UC21 – CANVIAR DIRECCIÓ D'ENVIAMENTS**
ACTOR	Usuari
DESCRIPCIÓ	Procés per a canviar la direcció per defecte dels enviaments de les comandes.
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina del seu perfil, dins de la configuració a l'apartat de "Canviar direcció d'enviaments".
FLUX BÀSIC	1. L'aplicació mostra la direcció actual i un "*input*" per al cas que es vulgui modificar. 2. L'usuari omple l'input amb la nova direcció i prem el botó per a realitzar el canvi 3. El sistema detecta les noves dades i actualitza la direcció de l'usuari
FLUX ALTERNATIU	2.1 L'usuari introdueix les noves dades en un format invàlid 2.2 El sistema mostra un missatge d'error a l'usuari
RESULTATS	S'actualitza la direcció d'enviament de l'usuari.

Figura 27: *Taula per analitzar el cas d'ús 21: Canviar direcció d'enviament de l'usuari.*

2.2.4 Escenaris separats

A continuació tenim les "*user stories*" que es realitzen en escenaris separats. Primerament, i una de les més importants, la generació de comandes que es realitza en una pàgina a part, ja que s'aprofitarà l'espai per a recomanar productes a l'usuari i mostrar amb molta més claredat la comanda i tota la informació associada. És per aquests motius i a part la importància que adquireix en l'aplicació que tota aquesta funcionalitat es realitza en un escenari separat.

D'altra banda tenim el canvi de contrasenya, que per temes d'usabilitat s'ubicarà a la pantalla en la que s'inicia sessió, apartada de tota la resta de configuració de l'usuari. Es realitzarà d'aquesta manera ja que l'únic moment en que l'usuari necessita la seva contrasenya és a l'iniciar sessió, i és allà on realment es preocupa en cas que no se'n recordi. Un cop iniciada la sessió i a més a més amb la tecnologia de caché per a recordar la sessió de l'usuari, aquest no pensarà en la contrasenya de l'aplicació pràcticament mai més. Per aquest motiu, s'ha decidit ubicar aquesta opció apartada, ja que es confia en què no sigui utilitzada pràcticament per a cap usuari i a part, suposarà una opció més al menú de la pàgina del client, fet que pot causar més confusió per a trobar els objectius ràpidament.

USER STORY 6.- COM A USUARI DE L'APLICACIÓ VULL PODER GENERAR COMANDES I VEURE EL SEU ESTAT EN TEMPS REAL.

Com s'ha analitzat anteriorment, en l'escenari de la carta l'usuari pot anar afegint productes a la comanda, però, com finalitza la comanda l'usuari? Per a aquest cas concret no es realitzarà diagrama de casos d'ús ja que es basa en una simple acció. S'analitzarà simplement amb la taula.

NOM	**UC22 – FINALITZAR COMANDA**
ACTOR	Usuari
DESCRIPCIÓ	Procés per a finalitzar una comanda acabada
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió iniciada i es troba a la pàgina de la seva comanda.
FLUX BÀSIC	1. L'aplicació mostra el contingut de la comanda, les recomanacions del local i les recomanacions adaptades a la comanda de l'usuari. 2. L'usuari prem el botó per a finalitzar comanda. 3. El sistema mostra un missatge per a confirmar que l'usuari ja ha finalitzat la comanda 4. L'usuari introdueix les dades de recolliment de la comanda 5. L'usuari finalitza la comanda.
FLUX ALTERNATIU	3.1 L'usuari no accepta finalitzar la comanda. 3.2 El sistema atura la finalització de la comanda i permet a l'usuari seguir editant-la. 4.1 L'usuari introdueix les dades del recolliment de la comanda amb un format invàlid. 4.2 El sistema mostra un missatge d'error informant a l'usuari
RESULTATS	Els administradors reben una notificació d'una nova comanda i l'usuari pot seguir dins del seu perfil l'estat en temps real d'aquesta comanda.

Figura 28: *Taula per analitzar el cas d'ús 22: Finalitzar una comanda.*

USER STORY 7.- COM A USUARI DE L'APLICACIÓ VULL PODER CANVIAR LA CONTRASENYA DEL COMPTE EN CAS DE QUE ME N'OBLIDI

Aquesta "*user story"* també serà analitzada amb la taula degut a que es basa en una sola acció.

NOM	**UC23 – CANVIAR CONTRASENYA**
ACTOR	Usuari
DESCRIPCIÓ	Procés per a canviar la contrasenya del compte de l'usuari
PRE CONDICIONS	L'usuari té un compte en l'aplicació, la sessió sense iniciar i es troba a la pàgina per a realitzar l'inici de sessió, a la pestanya de "Has oblidat la teva contrasenya?"
FLUX BÀSIC	1. L'usuari introdueix la seva adreça de correu electrònic actual. 2. El sistema li envia un correu amb un enllaç a la pàgina per a actualitzar la nova contrasenya. 3. L'usuari es dirigeix al correu, visita l'enllaç que se li proporciona i omple les dades amb la nova contrasenya. 4. El sistema revisa les dades i realitza el canvi.
FLUX ALTERNATIU	1.1 El sistema no detecta el correu actual de l'usuari. 1.2 El sistema mostra un missatge d'error informant a l'usuari. a. L'usuari introdueix les dades de la nova contrasenya amb un format invàlid. b. El sistema atura el canvi de contrasenya i mostra un missatge d'error informant a l'usuari.
RESULTATS	L'usuari ha configurat una nova contrasenya associada al compte. A partir d'aquest moment l'usuari haurà d'iniciar sessió amb la nova

Figura 29: *Taula per analitzar el cas d'ús 23: Canvi de contrasenya.*

2.3 Futur de l'aplicació

Un aspecte important a considerar en aquesta fase d'anàlisis, és que l'aplicació serà utilitzada per a l'empresa un cop el projecte estigui acabat. Aquest factor ens beneficia degut a que tenim la informació del grup de clients que faran ús de l'aplicació, dada que ens permet estimar termes d'escalabilitat del sistema i que ens donarà més agilitat a l'hora de realitzar el disseny de certes característiques de l'aplicació.

Segons la informació del "*Product Owner"*, sabem que el nombre de clients que poden arribar a utilitzar l'aplicació pot estar al voltant d'uns 200. Per tant, el sistema ha de poder gestionar 200 comandes alhora, mentre que també va gestionant paral·lelament totes les peticions de reserves que es van realitzant.

El testeig del funcionament de l'aplicació amb diferents conjunts de dades generades aleatòriament es realitzarà a la part de resultats, en el "*Stress Test"*.

Testeig

L'aplicació en ser finalitzada, serà provada durant un mes per a l'empresa que la demana. Durant aquest mes de testeig, el "*Product Owner"* analitzarà el funcionament del sistema i decidirà quines funcionalitats s'han d'afegir, treure o modificar de cara al primer manteniment.

Manteniment

L'aplicació en si ja estarà dissenyada de tal forma que no necessiti un manteniment excessiu, optimitzant al màxim els recursos i no generant una gran quantitat de dades on gran part d'aquestes no tinguin utilitat. Tot i així és inevitable que s'arribi a un punt on la memòria del "*hosting"* on s'allotgi la web acabi arribant als seus límits degut a que cada dia es generin un cert nombre de dades en l'aplicació.

Caldrà doncs periòdicament fer un refresc de la base de dades, on no caldrà eliminar les dades generades, simplement s'emmagatzemaran en un fitxer a part que podrà ser consultat en qualsevol moment per l'empresa, i es deixarà la base de dades neta una altra vegada.

Les dades que s'hauran de mantenir seran:

- Conjunt de comandes
- Conjunt de reserves
- Usuaris amb un cert període d'inactivitat
- Cupons caducats

No caldrà mantenir:

- Esdeveniments, ja que ja seran autogestionats pels administradors.
- Imatges de perfil d'usuaris, ja que només guardarem les que són realment utilitzades.
- Productes i ingredients, també autogestionats pels administradors.

3. Disseny

Per a l'òptim funcionament de l'aplicació s'han utilitzat diferents tecnologies, s'han implementat diferents patrons i s'ha seguit una estructura determinada. En aquest apartat de disseny observarem i analitzarem l'arquitectura interna de l'aplicació.

3.1 Tecnologies utilitzades

A continuació s'analitzarà el disseny de l'aplicació basat en les tecnologies que s'han utilitzat per a desenvolupar-la. Primerament es parlarà sobre els diferents *"frameworks"* tant de *"frontend"* com de *"backend"* i posteriorment s'argumentarà l'enfocament que se li ha donat al flux de funcionament a l'aplicació.

3.1.1 Laravel

Per a tota la gestió de dades interna de l'aplicació, en altres paraules, les funcionalitats del servidor o ***"backend"*** han sigut realitzades mitjançant l'ajuda del *"framework"* Laravel. Aquest *"framework"* té dependències de "*Symfony*" i està basat en el llenguatge de programació PHP.

"*Laravel*" ens ajudarà a optimitzar el codi i a fer-lo visualment molt més agradable, gràcies a les grans simplificacions que ens porta. Les seves característiques principals de les quals s'ha extret profit són les següents:

- **Sistema de rutes (*"Laravel Routing"*)**: Sistema que ens permet definir una sèrie de rutes en l'aplicació, i gestionar, depenent el tipus de crida a la ruta, a quin controlador i a quina funció cridar.

- **"*Blade*", motor de plantilles**: Sistema d'arxius que et permet desenvolupar tota la vista, i a la vegada incloure-hi codi PHP. Alhora també inclou un munt de dreceres per a incorporar funcionalitats de Laravel a les vistes. Per defecte, els arxius "*blade*", són emmagatzemats a caché, el que els fa extremadament ràpids!

- **Peticions "*Fluent*"**: Sistema de crides per a la gestió de la base de dades, d'una manera molt simple i clara. Gestiona totes les consultes realitzades assegurant la seguretat contra atacs d'injecció.

- **Suport per al patró MVC**: Tot i que el mateix *"framework"* et permet implementar diferents patrons de programació, està força orientat i documentat de tal manera perquè es pugui implementar el patró MVC fàcilment.

- **Laravel "*Elexir*"**: Sistema d'agrupació d'arxius que ens permet no declarar a cada pàgina l'ús de "*scripts*" i estils que es repeteixen. Es basa en un fitxer el qual se li pot indicar quines agrupacions realitzar en quins arxius, i a quin arxiu final acabaran. Per exemple, en el nostre cas concret s'agrupen tots els "*scripts*" en un fitxer que els engloba i tots els fitxers d'estils en un altre. Aquests dos fitxers finals són els que s'afegeixen a l'estructura bàsica de la vista, que veurem més endavant.

- ***"Eloquent: Relationships"***: Com indica el nom, Laravel ens dóna també un sistema eloqüent per a establir totes les relacions entre els nostres models. Mitjançant la crida de mètodes que indiquen clarament de quina relació es tracta, el "*framework*" mateix ja crea les taules necessàries per a poder emmagatzemar totes les dades necessàries per a aquella relació.

- **Sistema de sessions:** El mateix Laravel ja ens aporta un mòdul que gestiona les sessions automàticament, guardant les dades necessàries en caché, revistant la caducitat de les sessions en realitzar una petició, etc. Com a programadors, podem establir quin serà el temps de caducitat de la sessió, si les dades de sessió es guarden encriptades o no etc.

3.1.2 Vue JS

Per a tota la gestió de la interfície d'usuari i les funcionalitats associades a aquesta, o en altres paraules, el que és conegut com a "***frontend***", s'ha utilitzat el "*framework*" Vue. Aquest "*framework*" està basat en llenguatge "*JavaScript*" i té un munt de funcionalitats.

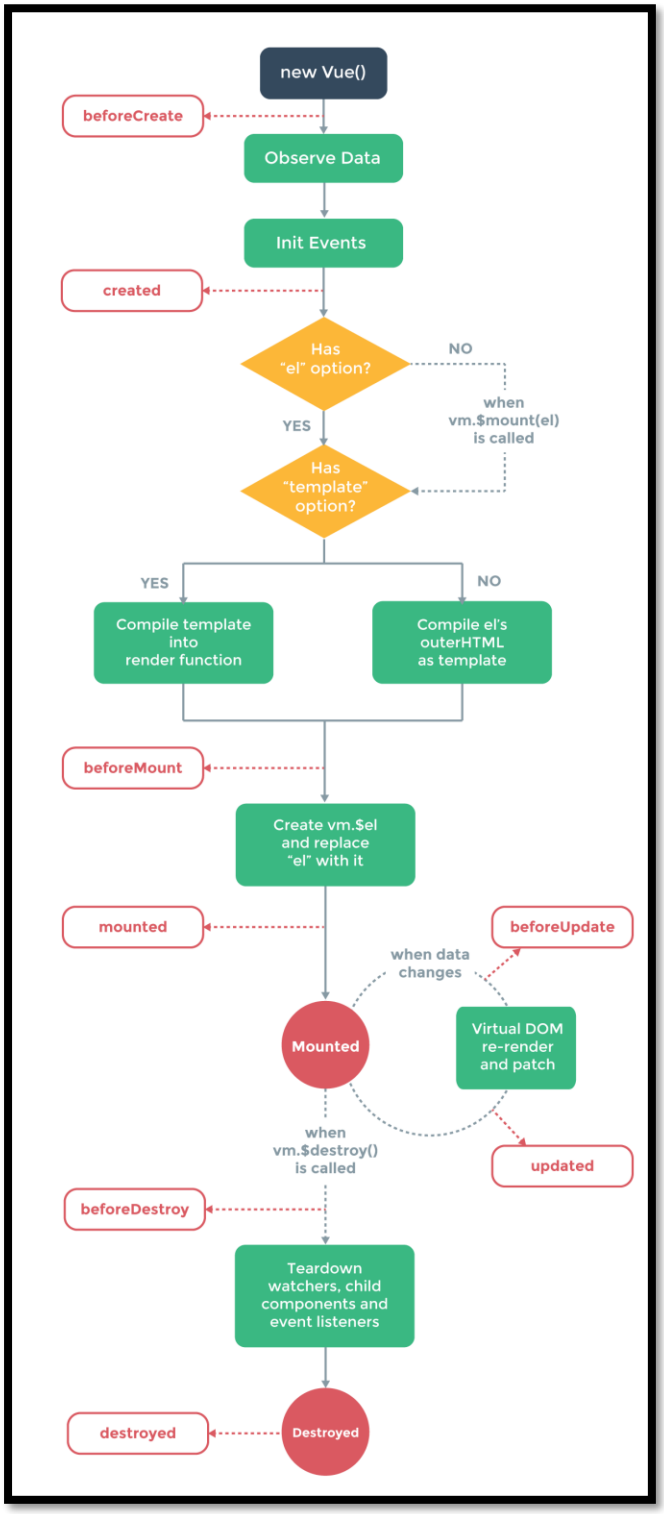
Particularment, per a la gestió de totes les dades a la banda del client, Vue ajuda i fa molt clar i entenedor tot el flux de funcionalitat d'ençà que es renderitza la pàgina fins que es destrueix.

Vue utilitza una sintaxi de plantilla basada en HTML que ens permet enllaçar de forma declarativa el DOM representat a les dades subjacents de la instància de Vue.

Combinat amb el sistema de reactivitat, Vue pot calcular el nombre mínim de components per tornar a renderitzar i aplicar la quantitat mínima de manipulacions del DOM quan l'estat de l'aplicació canvia. El sistema de reactivitat es basa en la manipulació i actualització automàtica (re-renderització) dels objectes de "*JavaScript*" en segon pla. Fet així que quan un objecte és modificat, s'actualitzi automàticament a la vista.

Vue també ens ofereix la declaració de components, que serien un conjunt de peces de codi que es van repetint al llarg de la vista. En altres paraules, aquesta característica ens permet reaprofitar un codi HTML de la vista, juntament amb els seus estils i "*scripts*" pertinents, associant-lo a una nova etiqueta HTML, que podrà ser utilitzada en qualsevol espai de la vista.

Segons la figura 30 que és una imatge oficial de la documentació de Vue caldrà seguir el següent cicle de vida d'un objecte Vue:



* Obtenció de les dades necessàries per a la vista

* Inicialització d'escoltadors d'events

* Neteja de la memòria dinàmica utilitzada.

Figura 30: *Cicle de vida d'un objecte Vue des de que s'instancia fins que és destruït.*

3.1.3 *"WebSockets"*

Una de les funcionalitats amb més rellevància dins del projecte, era que l'usuari havia d'estar informat en temps real de l'estat de la seva comanda o reserva. Per a simular aquest temps real, és necessari l'ús de les tecnologies dels "*WebSockets"*.

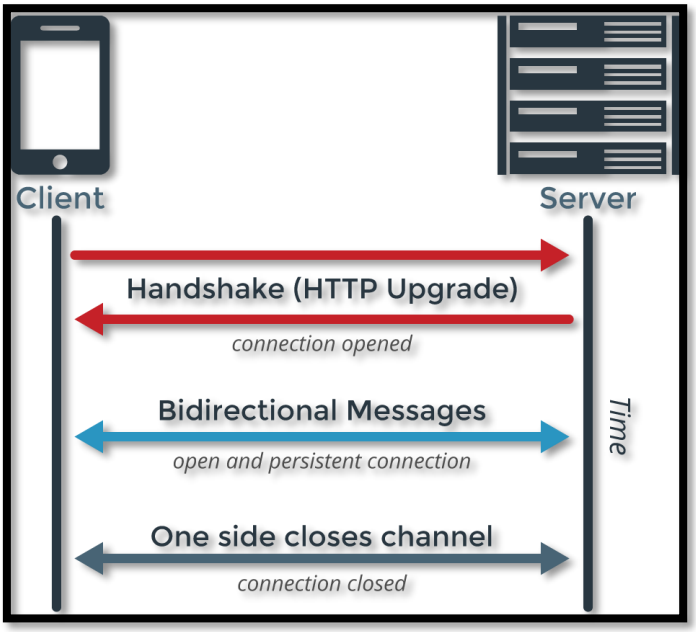


Figura 31: *Esquema d'interacció entre client i servidor amb tecnologia "WebSockets".*

La figura 31 defineix a la perfecció el funcionament dels "*WebSockets"*. A diferència de la connexió HTTP habitual, on el client obra la connexió amb el servidor fent una petició, i el servidor la tanca retornant una resposta, aquesta és força diferent.

S'inicia amb una connexió HTTP del client, fet que és mencionat com a "*Handshake"* a la imatge, i que en el nostre projecte és anomenat com a salutació o "dir hola". En aquesta salutació el client s'identifica al servidor, on aquest el registra a la seva llista d'usuaris i mantenen la connexió persistent. Mentre aquesta connexió estigui oberta, es podran anar intercanviant missatges entre ells. A diferència del protocol normal, on és sempre el client qui inicia la connexió, en aquest cas també pot fer-ho el servidor.

Aquesta connexió perdura fins que un dels dos dispositius tanca la connexió, i és llavors quan el servidor actualitza les dades dels clients que té connectats.

En termes més específics en la nostra aplicació, el sistema de missatges (una vegada s'ha establert la connexió entre client i servidor via "*WebSocket*") és el que es veu en la figura següent:

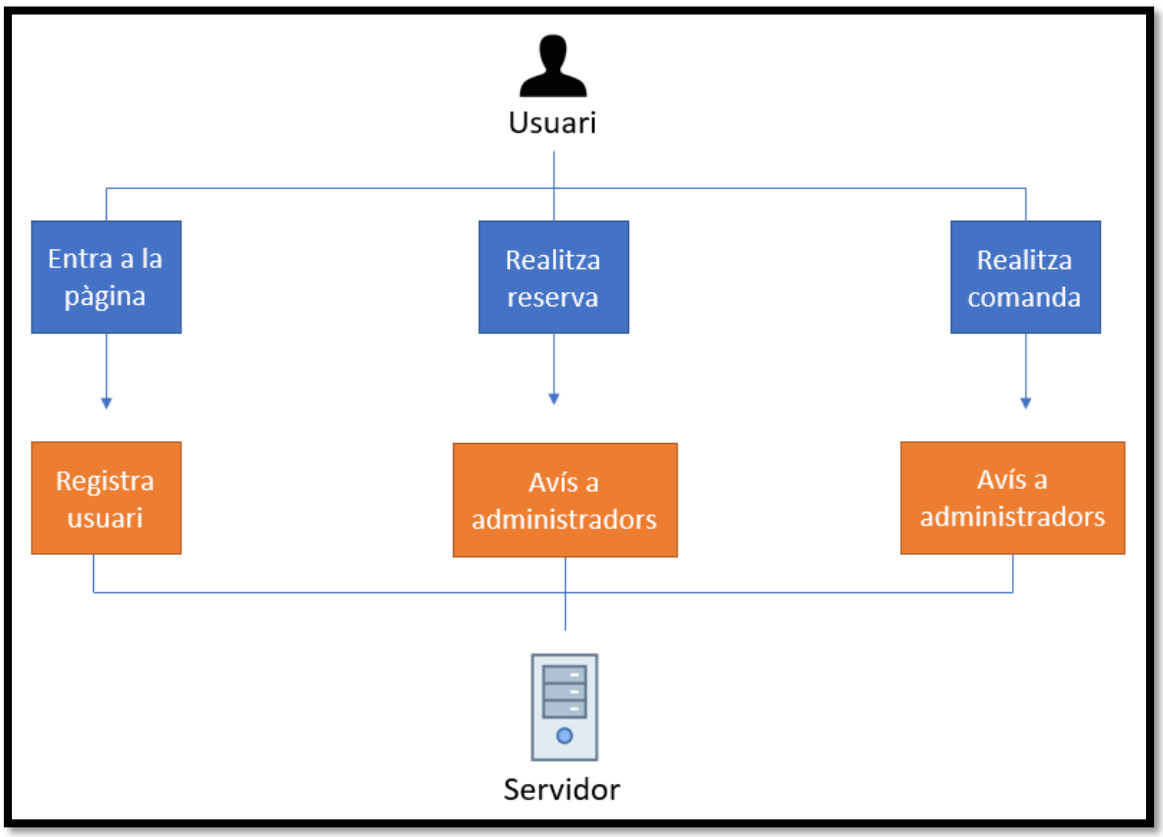


Figura 32: *Diagrama d'acció del WebSocket per part del client, relacionat amb les accions que realitzarà conseqüentment el servidor*

En termes del client, quan aquest entra a la pàgina, s'inicia la connexió a partir de la petició HTTP. L'usuari s'identifica al servidor i aquest l'emmagatzema per a poder enviar-li missatges en futures ocasions.

Mentre l'usuari vagi navegant per la pàgina, la connexió amb el servidor es mantindrà oberta i aniran intercanviant missatges. Els missatges que poden intercanviar s'adapten a les necessitats de l'aplicació, per tant:

S'enviarà un missatge al servidor quan l'usuari realitzi una reserva o bé una comanda. Aquest missatge es codifica, en format JSON, juntament amb totes les dades necessàries.

Un cop arriba al servidor, aquest descodifica el missatge i depenent del tipus de missatge que sigui realitzarà una acció o una altra. En tractar-se d'una nova reserva o comanda, aquest fa una crida a una funció que avisarà als usuaris administradors del nou esdeveniment.

L'enregistrament i gestió de la comanda a la base de dades s'ha realitzat en un altre fil d'execució, el "*WebSocket*" simplement és usat per a l'avís en temps real d'aquestes notificacions. S'ha de tenir en compte que el servidor tindrà molts "*WebSockets*" oberts alhora, un amb cada usuari connectat, i que poden estar intercanviant missatges tots a l'hora. Per tant, cal assegurar no sobrecarregar el "*WebSocket*" amb funcionalitats complexes.

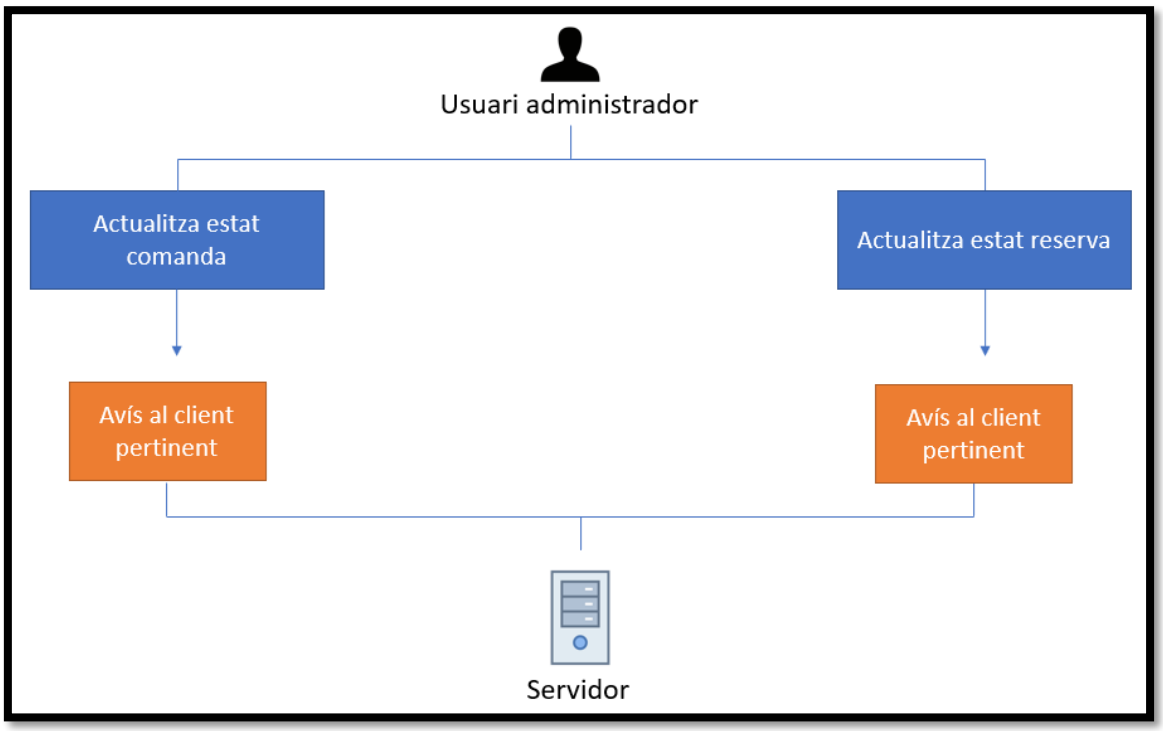


Figura 33: *Diagrama d'acció del WebSocket per part de l'usuari administrador, relacionat amb les accions que realitzarà conseqüentment el servidor*

En termes d'usuaris administradors, com es veu a l'anterior figura, quan aquests reben una notificació asíncrona que algun client ha realitzat una nova comanda o reserva es dirigeixen al panell d'administració o "*Housekeeping*".

Des d'allà, poden actualitzar l'estat de la reserva o la comanda segons ells creguin necessari. En el moment en què s'actualitza una reserva o una comanda, s'envia un nou missatge codificat del "*WebSocket*" de l'administrador cap al servidor.

El servidor, com en el cas anterior, descodifica les dades del missatge i mira segons el tipus de missatge que sigui, quina acció ha d'emprendre. En aquest cas s'adona que el missatge prové d'un administrador que ha actualitzat l'estat d'un client "X". Per tant, en aquest cas el servidor recuperarà el "*WebSocket*" de l'usuari "X" i li enviarà una notificació asíncrona informant-lo en temps real del nou estat de la seva comanda o reserva.

Quan un usuari es desconnecta de la pàgina, el servidor detecta la desconnexió i tanca el "*WebSocket*" pertinent d'aquell usuari. Quan un administrador actualitzi l'estat d'una comanda o reserva d'un usuari que actualment no es troba connectat a la pàgina i que per tant, no té un "*WebSocket*" obert, simplement no s'enviarà el missatge, ja que el servidor no trobarà on enviar-lo. Tot i així l'usuari quan torni a entrar a la pàgina per veure l'estat de la seva comanda o reserva, aquest estat ja haurà estat actualitzat i l'usuari serà informat igualment.

La implementació dels "*WebSockets*" s'ha realitzat gràcies a la llibreria "*Ratchet*" de PHP, i s'executa mitjançant una comanda al servidor que inicia la gestió de les connexions. Un cop iniciada la comanda, el mateix servidor, com s'ha explicat, ja va gestionant les connexions i desconnexions ells sol, guardant i alliberant memòria contínuament.

3.1.4 *"Gmail Mailing"*

L'aplicació porta incorporada un sistema de "*Mailing"* associat a la plataforma de Google "*Gmail*". D'aquesta manera, obtenim diversos avantatges en el conjunt de la plataforma que són força interessants:

- **Major seguretat en el sistema:**
  Donat que el correu per a registrar-se és únic, i s'ha de confirmar, ens assegurem que l'usuari no sigui fals i que realment és un usuari que vol donar un bon ús a la nostra aplicació.

- **Sistema eficaç per a contactar amb clients:**
  Sabent que els correus dels usuaris són els correctes, ens permet posteriorment poder-los informar de promocions, esdeveniments i tot tipus d'informacions que volem promulgar. Segons un estudi de eMarketer (enllaç situat a la bibliografia) i basant-nos en les dades mostrades en la figura 34, denotem que el correu electrònic és la millor eina per fer arribar informació als nostres clients.

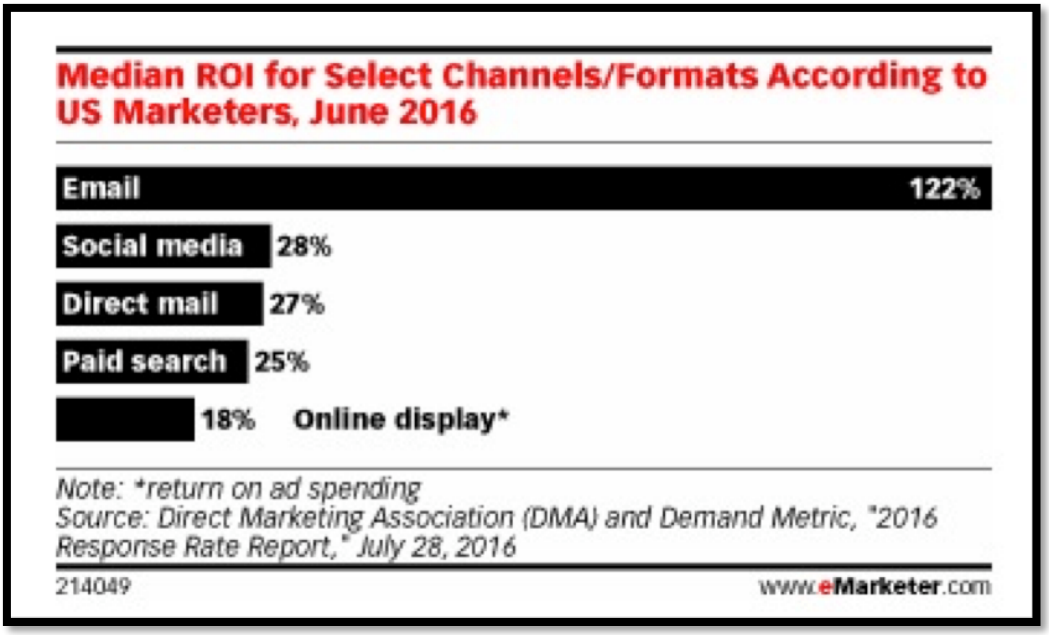


Figura 34: *Estadística publicada per els investigadors de eMarketer on es demostra l'eficàcia del correu electrònic en comparació amb altres mètodes.*

- **Sistema personalitzable:**
  Podem personalitzar els correus segons la seva finalitat. Els correus de confirmació de registre no tenen la mateixa aparença que els correus de promocions, ni tan sols fan ús de les mateixes dades. De la mateixa manera, una de les dades a l'hora d'enviar un correu a un usuari són les dades d'aquest. Per tant es pot arribar a personalitzar el correu fins a tal mesura de personalitzar-lo segons els gustos de l'usuari.

- **Sistema interactiu i proper a l'usuari:**
  Tot aquest sistema automàtic prové d'una conta de "*Gmail"* normal i corrent. Aquesta conta és la que s'associa al programa i envia correus automàticament. Tot i així aquesta conta de correu pot ser utilitzada d'igual manera com a adreça de correu electrònic normal i corrent, i els clients poden contactar amb els administradors de l'empresa amb aquest mateix correu. D'igual manera, els administradors podran contestar els missatges de forma lliure des de la pàgina web del "*Gmail".*

L'únic inconvenient que se li pot trobar al sistema de correus automàtics amb "*Gmail"* és que no és mesurable. No ens dóna cap eina d'estudi per poder analitzar l'eficàcia dels correus a l'hora de llençar promocions, esdeveniments, etc...

D'altra banda, "*Gmail"* té un detector automàtic de quan una adreça de correu pot tractar-se de "*spam"*. Aquest detector salta quan detecta que s'han estat enviant molts correus seguits en un període de temps molt curt des de la mateixa direcció. Aquesta escena podria arribar a ocórrer en la nostra aplicació si es dóna el cas que s'arriben a tenir el nombre mínim d'usuaris determinats per considerar-se com a "*spam"*.

3.2 Arquitectura de l'aplicació

L'estructuració dels fitxers del projecte ha seguit el patró MVC (model, vista i controlador). S'ha seguit aquest patró adoptant la pròpia estructura de fitxers que el mateix *"framework"* Laravel implica.

Aquest patró ens permet organitzar molt efectivament l'estructura de fitxers, a més a més s'acobla molt bé amb la mateixa estructura del projecte Laravel. Amb aquest avantatge ens apropem més a l'objectiu d'assolir un codi perfectament estructurat i modularitzat per tal de donar-nos facilitats a l'hora de realitzar implementacions o manteniments en un futur.

D'altra banda, hem seguit aquest patró degut a que encaixa perfectament amb la idea d'una aplicació web "*Restful"*: Donada una petició d'un client, aquesta es detecta i es crida al controlador pertinent per tal d'atendre aquesta petició. En aquest controlador es processen les dades necessàries associades als models lògics i a les seves relacions, que posteriorment es carregaran amb la vista i s'entregaran com a resultat a l'usuari.

3.2.1 *"Restful"* web

La web disposa d'una arquitectura "*Restful"*, que es basa en el protocol HTTP i té les següents característiques:

- **Protocol client/servidor sense estat**: Es basa en un protocol de client/servidor on no tenim estat. En cada petició HTTP que fem estem enviant tota la informació necessària per a executar-la. Aquesta característica ens permet lliurar al client i al servidor de recordar l'estat previ per tal de realitzar noves tasques.
  Ens permet també definir algunes respostes a peticions HTTP a caché, per poder retornar molt més ràpidament les mateixes respostes a les peticions idèntiques.

- **Ús dels mètodes HTTP de manera explícita**: Per a la gestió de les dades s'utilitzen diferents mètodes a diferents URIs:
  - Obtenir o llegir: GET
  - Enviar o crear: POST
  - Actualitzar: PUT
  - Eliminar: DELETE

  Per exemple si a l'aplicació volem eliminar el producte amb identificador número 1, simplement caldrà fer una crida amb el mètode HTTP pertinent a la URI concreta:

  DELETE /products/1 HTTP/1.1

  Mètode HTTP URI Protocol HTTP

- **Interfícies basades en URIs:** Per a accedir a un recurs del servidor i realitzar una determinada acció, prèviament cal definir una URI. Aquestes URIs han de ser intuïtives i ens permeten definir un procés únic per a dur a terme una acció a un determinat recurs.

- **Format JSON en la transferència de dades:** En les transferències de dades que es realitzen entre el servidor i el client, s'utilitza un únic format de dades, per tal de facilitar l'obtenció i enviament d'aquestes. El servidor també envia codis d'estat associats a les seves respostes, de tal forma en arribar la resposta al client, aquest pot respondre segons com hagi anat el procés en el servidor.

3.2.2 Vista

El conjunt de fitxers que formen les diferents vistes que conté l'aplicació, es troben dins de la carpeta "***resources"***, a dins de la carpeta "***views"***.

Dins d'aquest directori es poden trobar totes les vistes de l'aplicació, estructurades segons l'ús que se'ls hi dóna. Es poden diferenciar diferents conjunts de vistes:

"*Emails*"

Dins d'aquest conjunt de vistes trobem tots els fitxers necessaris per a donar a forma als diferents correus electrònics que s'envien automàticament des de l'aplicació. Hi ha diferents tipus de correus automàtics dins de l'aplicació, com per exemple el de confirmació a l'hora de crear la conta d'usuari, el de restablir tant el correu electrònic com la contrasenya i el de promocions. Cadascun d'aquests correus requereixen dades diferents i també s'organitzen les dades visualment d'una manera única en cadascun d'ells.

Vistes principals

Dins d'aquest conjunt de vistes es poden trobar aquells fitxers que són utilitzats habitualment en la pàgina. Com serien la capçalera i el peu de pàgina ("*header*" i "*footer*"), la pàgina principal o de benvinguda, la pàgina d'ofertes i la pàgina que dóna informació sobre l'empresa, anomenada Nosaltres.
Aquest conjunt de vistes segueix un patró d'estructuració, que facilita a termes de codificació, tota l'ordenació final del codi.

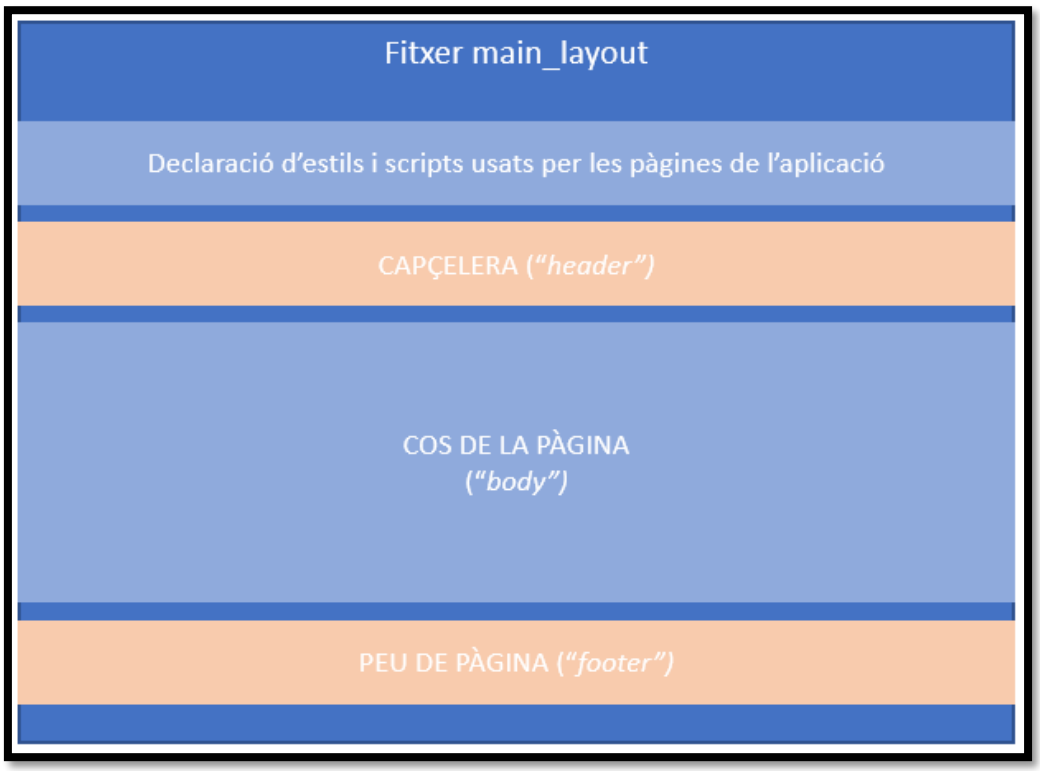


Figura 35: *Representació gràfica del fitxer "main_layout" amb totes les parts que el formen.*

El fitxer que implementa aquest patró s'anomena "*main_layout"*, i com s'aprecia a la figura 35 conté 4 apartats.

- Declaració dels fitxers d'estils i "*scripts"* que utilitzarà la pàgina i tots els seus subconjunts.
  Dins dels "*scripts"* utilitzats, trobem un fitxer anomenat com a base, que conté totes les llibreries principals necessàries per al funcionament, com podrien ser; *JQuery*, *Bootrstrap*, *Axios* etc... D'altra banda trobem el "*script"* necessari per gestionar la comanda de l'usuari en qüestió. Es globalitza ja que l'usuari pot anar modificant la seva comanda des de diverses pàgines de l'aplicació. Per últim trobem declarat també el "*script"* encarregat de gestionar el funcionament dels "*WebSockets*".
- Capçalera de l'aplicació ("*header")*
- Contingut principal de la pàgina (Dinàmic)
- Peu de pàgina de l'aplicació ("*footer")*

Utilitzant aquest patró ens estalviem haver de declarar els fitxers d'estils i "*scripts"* a cada pàgina en què s'utilitzen. D'aquesta manera simplement el que anirem canviant és el contingut de la pàgina, depenent de la petició que ens faci l'usuari. Per exemple, quan l'usuari ens faci una petició en la que vol accedir a la pàgina d'ofertes, carregarem aquesta estructura, amb el contingut de la secció d'ofertes.

Cal mencionar també que en el patró, es declaren els fitxers d'estils i "*scripts*" que es comparteixen entre totes les pàgines. Però després trobem que les pàgines individuals requeriran certs "*scripts*", que altres no en faran ús. Aquests requeriments es declararan a les mateixes pàgines en qüestió, per tal de no declarar "*scripts*" globalitzats que posteriorment no s'utilitzin en una àmplia majoria de pàgines.

"Housekeeping"

En aquest conjunt de vistes es troben tots els fitxers necessaris per a la visualització del panell de control per als administradors de l'aplicació.

Aquest conjunt de vistes també segueixen una estructura semblant al conjunt de vistes principals.

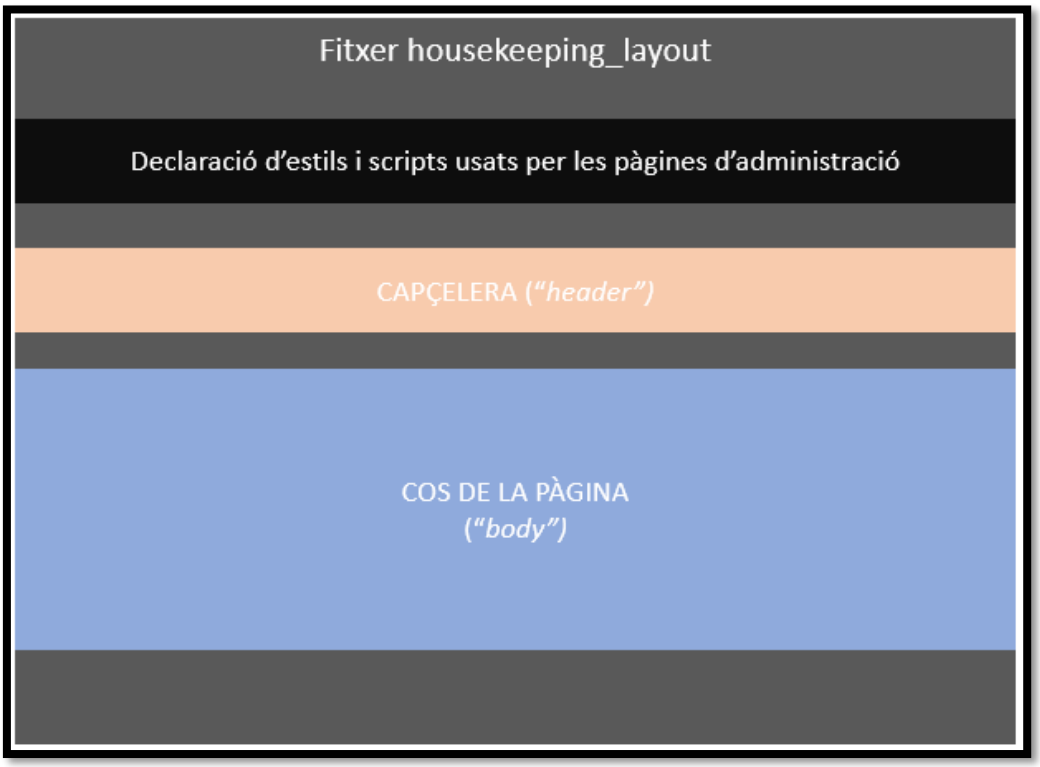


Figura 36: *Representació gràfica del fitxer "housekeeping_layout" amb totes les parts que el formen.*

El fitxer que implementa aquest patró s'anomena "*housekeeping_layout*", i com s'aprecia a la figura 36 conté 3 apartats.

- Declaració dels fitxers d'estils i "*scripts*" que utilitzaran totes les pàgines del panell d'administració
- Capçalera de l'aplicació ("*header")*
- Contingut principal de la pàgina (Dinàmic)

Els avantatges són les mateixes que prèviament, ens estalviem haver de declarar en cada pàgina quins estils i "*scripts*", seran necessaris.

No s'aprofita l'estructuració de vistes principals degut a que els *"scripts"* que s'utilitzen en aquest apartat són totalment diferents dels que s'utilitzen en la vista principal. Aquests estan destinats a la gestió (obtenció, modificació, lectura i eliminació) de tots els diferents models que tenim en l'aplicació. Per tant, en tenir funcionalitats tan diferents, s'optimitzarà l'estructura del codi separant-los en dos patrons diferents.

Dins d'aquest conjunt de vistes, trobem un subconjunt on s'han organitzat els fitxers que contenen les diferents finestres flotants o *"pop-ups"* que s'utilitzen en el panell de control. Se'n troben dues diferents per a cada model, una per a creació i l'altre per a edició.

"Partial"

Aquest conjunt de fitxers de vista és utilitzat a diferents pàgines de l'aplicació. Podem trobar aquelles finestres flotants o *"pop-ups"* que no pertanyen a cap dels grups anteriors. Un exemple podria ser la finestra flotant per afegir un producte a la comanda, que és el que s'observa a la figura 37. En aquest cas concret, aquest *"pop-up"* es crida des de la carta o també pot ser cridat des de la mateixa pàgina que resumeix la teva comanda.

També conté un fitxer de vista anomenat *"flash"*, que s'utilitza a diferents pàgines per tal de mostrar el resultat d'una petició després de ser processada pel servidor. Adopta un estil diferent en cas d'èxit i d'error i simplement imprimeix el missatge que el servidor li ha encomanat a mostrar.

Figura 37: *Finestra flotant per afegir producte a la comanda, pertanyent al grup "Partial" degut a que pot ser cridada des de diferents llocs de l'aplicació.*

Usuari

Aquest conjunt de fitxers conté totes les vistes necessàries per a la pàgina del perfil d'usuari. Aquest grup de vistes també conté un subconjunt de vistes, que contenen totes les finestres flotants utilitzades en la pàgina de l'usuari. Com podria ser a l'hora de realitzar una reserva, o bé crear una pizza o quan desitja repetir una comanda feta anteriorment.

El que seria la vista pròpiament dita de la pàgina de l'usuari, es divideix en dues parts, que es diferencien clarament en la figura 38. La primera seria el menú de navegació que permet a l'usuari moure's a través de totes les pantalles que se li ofereixen. L'altre seria el contingut, que es mostra de forma dinàmica, ja que es va amagant i mostrant segons la petició que fa l'usuari en el menú desplegable.

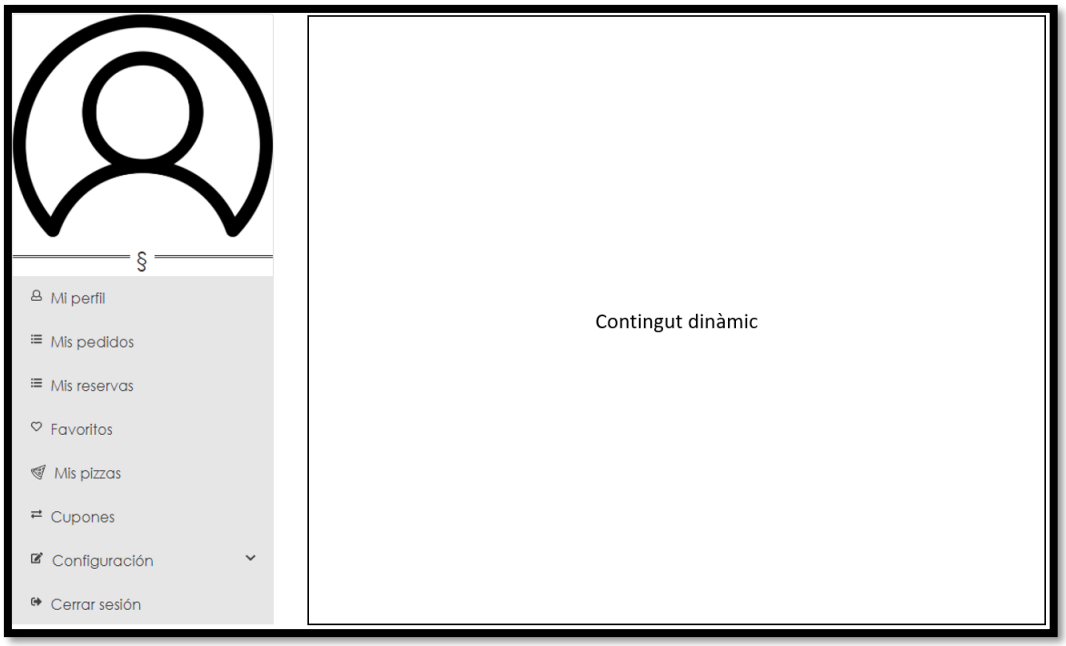


Figura 38: *Disseny de la pàgina de l'usuari, on es diferencia la part estàtica a l'esquerra referent al menú i la part dinàmica a la dreta referent al contingut.*

Patrons de disseny d'interfície

Sabent que l'aplicació és utilitzada per a un ampli grup d'usuaris objectius on les edats són força variades, ens assegurarem donar una bona impressió visualment i sobretot en termes d'usabilitat per tal d'assolir els objectius plantejats.

S'han utilitzat diversos tipus de patrons de disseny que faciliten a l'usuari la navegabilitat i la usabilitat en la web, factors que faran tenir a l'usuari una bona experiència amb la interfície. A continuació s'explicaran els diversos tipus de patrons utilitzats i les seves argumentacions.

Patrons d'introducció de dades

Aquest tipus de patrons faciliten i fan més clara la introducció de dades pels usuaris.

- *"Input Date Picker"*

  Per a la selecció de dates en l'aplicació, s'ha utilitzat aquesta funcionalitat que ens permet facilitar a l'usuari la introducció de dates i a la vegada ens ajuda en termes de validació de les dades, ja que en aquests "*Inputs*", els valors esperats no podran ser altres que els que el nostre "*Date Picker*" retorni.

  Per a la implementació d'aquest patró, s'ha fet ús de la llibreria "*JQuery UI*" i el resultat és el que es pot observar a la figura 39:

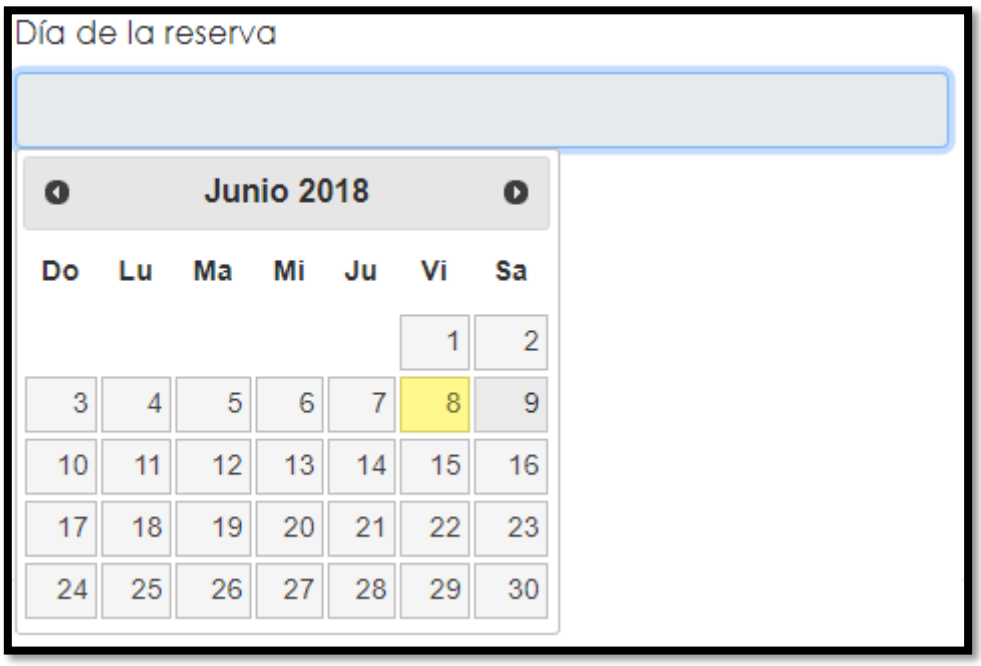


Figura 39: *Calendari que es desplega per tal d'ajudar a l'usuari al introduir dates en l'aplicació.*

- *"Input Prompt"*

  Aquest patró estableix que s'ha de donar un extra d'informació a l'usuari en els "*Inputs*" que poden generar dubtes sobre quin tipus de dades s'han d'introduir. Per a implementar aquest patró de disseny s'han utilitzat imatges per a ajudar a l'usuari a percebre de què es tracta i petits textos de suport com es pot veure a la següent figura:

Figura 40: *Introducció de dades en l'aplicació. Ajudem a l'usuari a entendre quins valors ha d'introduir mitjançant la imatge i el petit text descriptiu.*

Patrons de navegació

Aquest tipus de patrons faciliten la navegació per les diferents pàgines del web.

- *"Navigation Tabs"*
  L'usuari disposa en tot moment d'una capçalera fixa que li permet navegar fàcilment per totes les pàgines de la web.

  Dins d'aquesta capçalera s'incorpora un altre patró de navegació: *"Dropdown Menu"*, que es pot observar a la figura 41 i permet a l'usuari accedir fàcil i ràpidament a les pàgines del seu perfil personal.

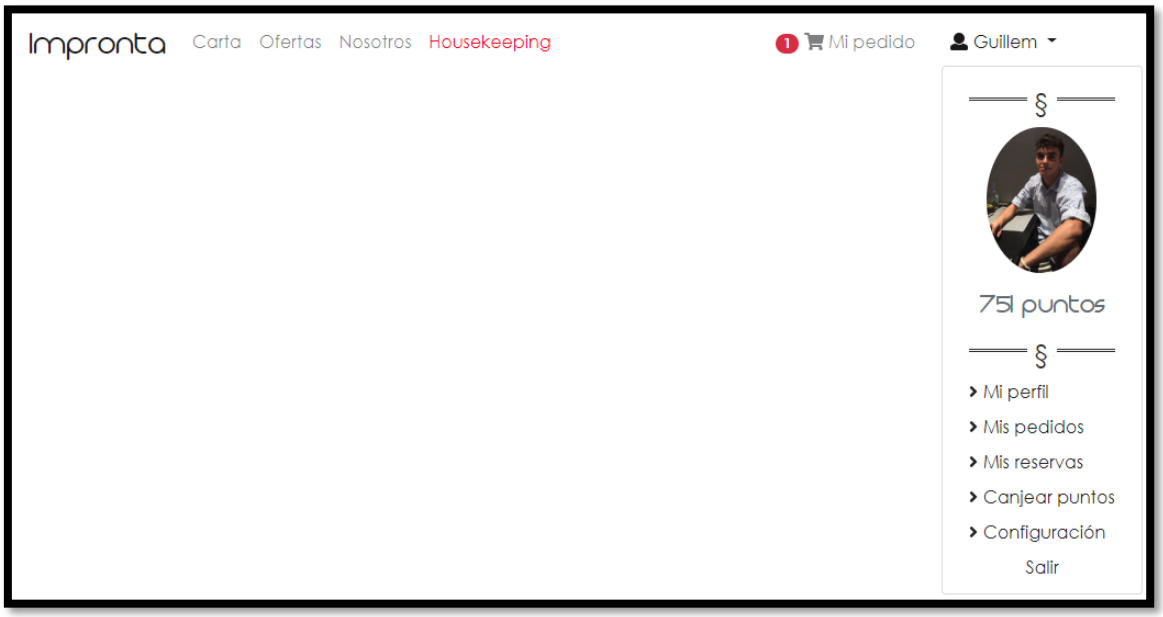


Figura 41: *Capçalera que conté el desplegable de l'usuari obert*

- *"Footer"*
  L'usuari disposa en tot moment d'un peu de pàgina que el permet dirigir-se ràpidament a les xarxes socials oficials del local. En aquesta figura 42 es pot observar el disseny del peu de pàgina.

Figura 42: *Peu de pàgina amb l'enllaç a les xarxes socials de l'empresa.*

- *"Accordion Menu"*

  Aquest patró de disseny tracta d'un menú que implementa la funcionalitat d'un acordió tal com s'observa a la figura 43. Aquest disseny permet donar més opcions a l'usuari en un espai de visualització més petit, factor que ens interessa de cara a la visualització del web des de dispositius amb pantalles petites.

  Per a optimitzar l'espai, s'ha implementat la funcionalitat que en prémer una opció de les desplegables, el menú es "plegui" sol. D'aquesta manera l'usuari pot veure el contingut de la pàgina sense necessitat d'un possible i molest "*scroll*".

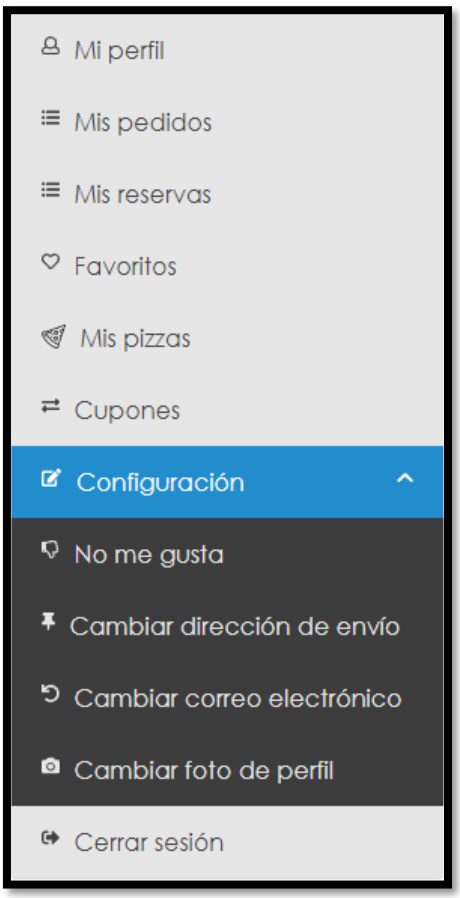


Figura 43: *Menú en acordió estès de la pàgina de l'usuari per a observar totes les funcionalitats.*

- *"Adaptable view"*

  Aquest patró de disseny permet a l'usuari adaptar la vista de la pàgina segons el seu criteri. Aquest patró s'ha implementat segons diferents interpretacions; Primerament, la pàgina de benvinguda canviarà segons els esdeveniments que hagin programat els usuaris administradors.

  L'usuari administrador també pot indicar que un producte està actualment esgotat, fet que farà variar la vista de la carta en aquell producte, indicant que està esgotat i fent desaparèixer el botó per a afegir aquell producte a la comanda.

  D'altra banda, la visualització de la carta canviarà, donats els criteris de l'usuari. En altres paraules, quan un usuari indiqui un ingredient com a negatiu, la carta adoptarà una visualització especial per a aquell usuari, informant-lo dels productes que contenen aquells ingredients.

  Per últim, el disseny de la carta s'ha implementat en desplegables, és a dir, que l'usuari en tot moment pot variar la longitud de la pàgina minimitzant o maximitzant els desplegables que contenen cada producte per a facilitar la navegació i comparació entre aquests.

- *“Favourites”*

  Aquest patró de disseny permet a l’usuari indicar certs objectes com a preferits i poder accedir d’una manera més ràpida que la resta. En el nostre cas aquest patró s’implementa amb els productes, on un usuari pot indicar que li agraden molts productes. Posteriorment, l’usuari té dues vies d’accés a aquests productes preferits; a la carta, en la secció de “Preferits” o bé en el seu perfil d’usuari, també en la secció “Preferits”. A la figura 44 s’observa, per una part, el botó per indicar com a preferit un producte i d’altra banda els diferents accessos que es proporcionen per a veure o seleccionar els diferents productes preferits.

Figura 44: *Botons de l’aplicació per a la interacció entre usuari i productes preferits. S’observa el botó per a afegir-ne un de nou (esquerra), el botó per veure’ls des de la carta (mig) i per veure’ls des del menú de l’usuari (dreta).*

- *“Home Link”*

  Aquest patró de disseny permet a l’usuari retornar a la pàgina d’inici del web fàcilment. Com s’aprecia a la figura 45, s’ha implementat en la capçalera de l’aplicació, i a distància d'un sol “*click*” a qualsevol apartat de la pàgina web, l’usuari pot dirigir-se a la pàgina d’inici.

Figura 45: *Capçalera de l’aplicació on es remarca a la part esquerre l’enllaç per a retornar en tot moment a la pàgina principal del web.*

- *“Notifications”*

  Aquest patró de disseny està implementat a l’aplicació en dues formes. Primerament les notificacions **estàtiques**, vistes en la figura 46, que podrien ser les que informen d’un possible error a l’usuari, o el nombre d’elements que contenim a la comanda.

  D’altra banda, tenim les notificacions **asíncrones**, vistes en la figura 47, que són les que s’executen via “*WebSocket*”. Aquestes notificacions varien segons si el usuari és administrador o no. Per exemple, quan es realitza una comanda (o reserva), els usuaris administradors reben una notificació de que s’ha realitzat una nova comanda (o reserva). Quan un d’aquests usuaris administradors actualitza l’estat d’aquesta comanda (o reserva), l’usuari que ha realitzat la comanda (o reserva) rebrà una notificació asíncrona informant del nou estat.

Figura 46: *Notificacions estàtiques de l’aplicació. Contingut de la comanda de l’usuari (esquerre) i missatge d’error (dreta).*

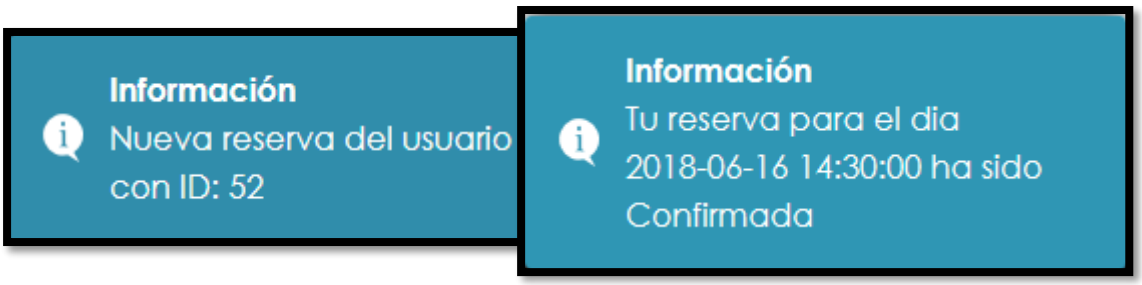


Figura 47: *Notificacions asíncrones de l’aplicació. Notificació per als usuaris administradors d’una nova reserva (esquerre) i notificació per al client que la reserva ha sigut confirmada (dreta)*

Patrons d'informació

Aquest tipus de patrons aclareixen i ajuden a l'usuari a visualitzar una determinada informació.

- *"Alternating Row Colors"*
  Aquest patró simplement es basa a variar el color entre files d'una taula, per a permetre a l'usuari una major interpretabilitat de les dades. S'ha implementat aquest patró en les taules del panell d'administració i es pot veure el resultat en la següent figura:

Usuario	Num. Personas	Fecha reserva
Carlos Larrayaga	5	11/06/2018 a las 20:30
Carlos Larrayaga	2	06/06/2018 a las 05:04
Carlos Larrayaga	1	09/06/2018 a las 10:40
Guillem Casassas	6	09/06/2018 a las 13:30
Carlos Larrayaga	7	16/06/2018 a las 20:30
Carlos Larrayaga	7	16/06/2018 a las 14:30

Figura 48: *Part de la taula que mostra les reserves al panell d'administració.*

- *"Carousel"*
  Aquest patró simplement es basa en un mètode interactiu per a mostrar certa informació, on l'usuari pot anar desplaçant-se per veure les diferents diapositives que estan contingudes dins d'aquest *"Carousel"*. S'ha implementat aquest patró per a mostrar els diferents esdeveniments del local de forma interactiva i agradable per a l'usuari. Es pot observar el resultat d'aquest patró a l'explicació del model dels esdeveniments, concretament a la figura 57.

- *“Autocomplete”*

  Aquest patró ajuda a l’usuari en la selecció d'informació, auto completant les possibles paraules que pot introduir. Aquest patró s’ha d’implementar en casos on sabem segur quin tipus de dades introduirà l’usuari. En el nostre cas, com s’observa a la figura 49, s’ha implementat a l’hora d’afegir ingredients a un producte. Aquesta funcionalitat serveix tant per l’usuari administrador en associar ingredients amb un producte, com per l’usuari normal de l’aplicació, a l’hora d’indicar quins ingredients estan continguts en la pizza que ha creat.

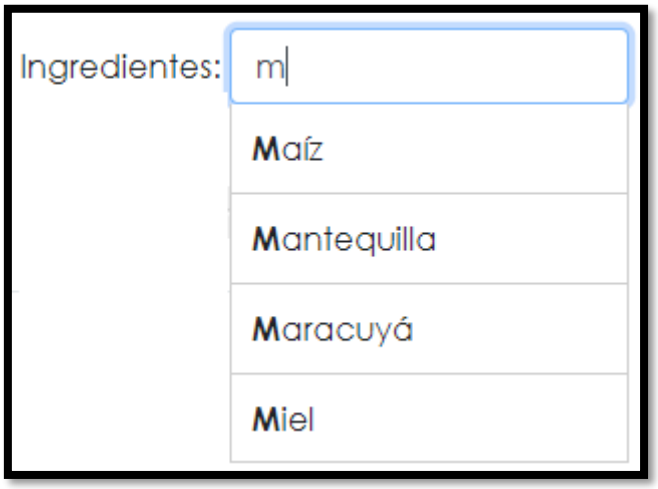


Figura 49: *“Input” de l’aplicació per a afegir ingredients a la pizza que l’usuari està creant.*

- *“Dashboard”*

Aquest patró permet a l’usuari contemplar visualment les dades que s’han generat en el seu entorn de l’aplicació. Aquest patró ha estat dissenyat, com s’observa a la figura 50, (com a implementació futura) en el panell d’administració i permet a l’usuari administrador veure gràficament les dades més rellevants que s’han generat mitjançant l’aplicació.

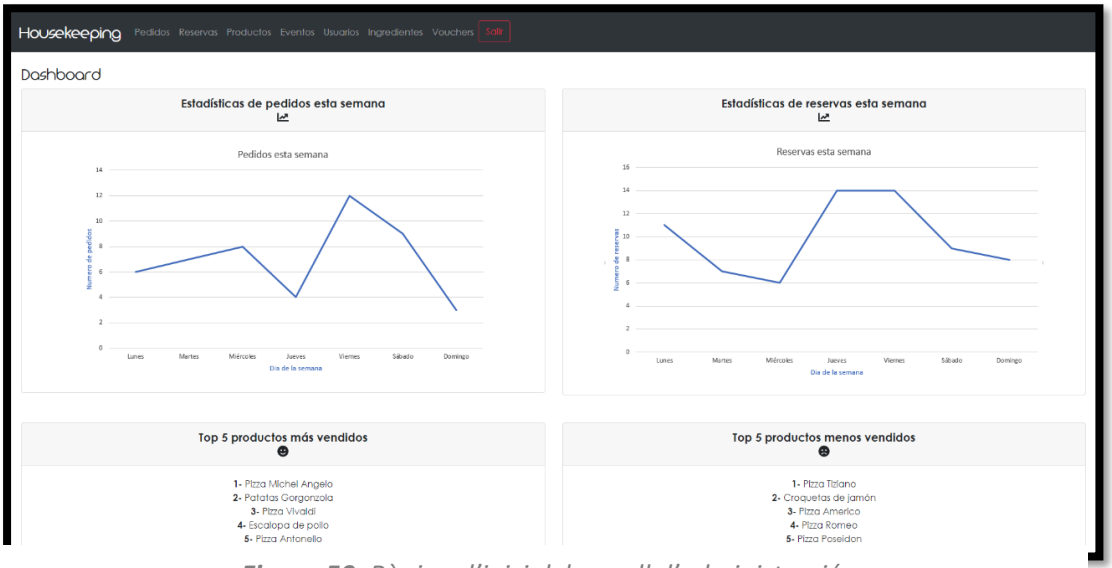


Figura 50: *Pàgina d’inici del panell d’administració.*

Patrons socials

Aquest tipus de patrons permeten la socialització entre els usuaris integrants de l'aplicació.

- *"Testimonials"*
  Aquest patró es basa a poder donar veu a l'usuari, perquè deixi opinions del web en general o bé d'objectes concrets. En el nostre cas s'ha implementat el sistema de comentaris als productes, on els usuaris poden comentar què els hi ha semblat cada producte i proposar millores per a ajudar a l'empresa.

Com a patrons socials no oficials afegits, es podria considerar el sistema de creació de pizzes i el sistema de "*Likes*", ja que permeten una certa interacció entre els usuaris, sempre que l'usuari configuri la pizza creada com a pública. La figura 51 mostra la interacció creada per un usuari que comenta una pizza creada per un altre usuari.

Figura 51: *Comentari d'un usuari a un producte de la carta creat per un altre usuari.*

Patrons de miscel·lània

Aquest tipus de patrons faciliten la interacció de l'usuari amb les funcionalitats principals de l'aplicació

- *"Shopping Cart"*
  Aquest patró permet tenir coneixement en tot moment de l'estat de la comanda realitzada per l'usuari, com s'observa a la figura 52. S'ha implementat aquest patró per a facilitar la interacció entre els productes de la carta i els productes de la comanda de l'usuari. A més a més, com aquest sistema funciona per **Sessió**, l'usuari si tanca el navegador i torna al cap d'una estona, seguirà mantenint la mateixa comanda que anteriorment.

Figura 52: *Part dreta de la capçalera de l'aplicació on podem veure la quantitat de productes de la comanda en tot moment.*

Vistes adaptables

Avui en dia l'accés a les pàgines web ja no és únicament des d'un ordinador, sinó que també s'accedeix des de tauletes, mòbils i altres. Cal assegurar que la visualització de la web sigui apta per a qualsevol pantalla des d'on es visualitzi, per tant s'ha hagut de dissenyar el codi d'una determinada manera.

Mitjançant la llibreria "*Bootstrap*" i la programació orientada a files i columnes, s'ha pogut adaptar el contingut a qualsevol pantalla, fent-la així totalment usable des de qualsevol dispositiu.

Aquesta programació orientada a files i columnes es basa en una matriu de 12 espais horitzontals i files il·limitades. La declaració d'elements gràfics dins d'aquestes columnes permet que donada una pantalla d'un dispositiu el contingut s'adapti segons les mides d'aquestes. Com s'observa en la figura 53, la matriu pot ser dividida al gust de l'usuari segons conveniència.

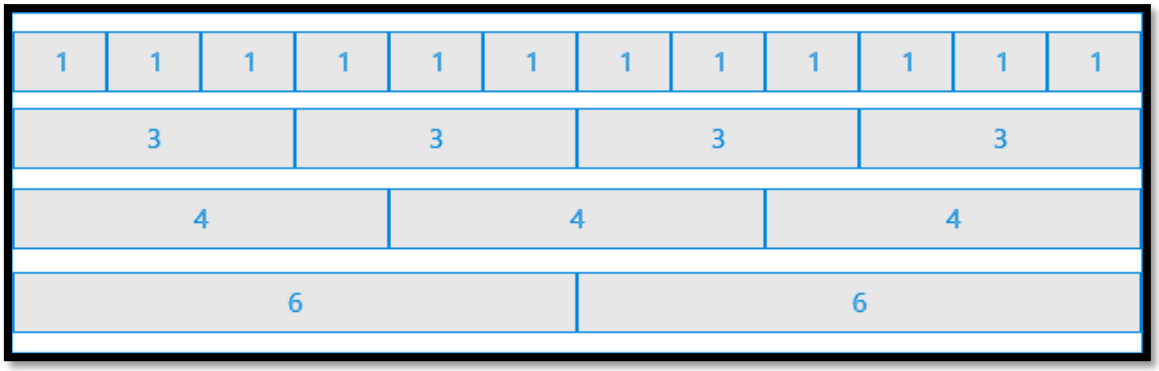


Figura 53: *Matriu de files i columnes establerta per "Bootstrap".*

3.2.3 Model

El conjunt de fitxers del model, representen els diferents objectes abstractes que generem i treballem a l'aplicació. Dins del projecte, els podrem trobar dins la carpeta de l'aplicació anomenada "***app***".

Els diferents objectes que trobem dins de l'aplicació són els següents:

"Book"

Objecte abstracte que representa una **reserva** en el local. Aquesta reserva la genera l'usuari dins de l'aplicació, i el conjunt de reserves es poden gestionar des del panell d'administració. Per a definir una reserva, fa falta un **usuari** a qui se li associa, un **nombre de persones** les quals realitzaran la visita, una **data** formada per un dia i una hora en la qual es realitza la reserva i un **estat**. L'estat serà utilitzat per informar a l'usuari en temps real si la reserva que ha demanat fer mitjançant l'aplicació, pot ser efectuada o no.

Aquest model adopta dos tipus de visualitzacions en l'aplicació; una per als administradors i una altra per als usuaris. Com s'aprecia en la figura 54, que representa la visualització per als administradors, es procura mostrar el màxim d'informació de la reserva en el mínim espai possible. En canvi a la visualització de la reserva per a un usuari, es procura adoptar un aspecte més agradable i informatiu, com es pot apreciar a la figura 55.

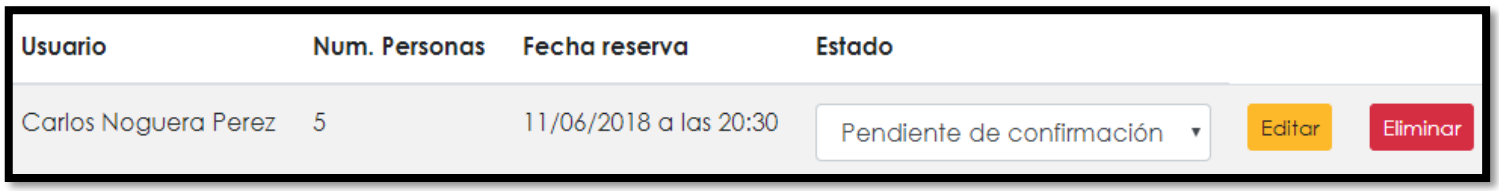


Figura 54: *Visualització d'una reserva des del panell d'administració.*

Figura 55: *Visualització d'una reserva des de la pàgina personal d'un usuari.*

"Comment"

Objecte abstracte que representa un **comentari** dins de la carta de l'aplicació. Els comentaris són realitzats pels usuaris i s'associen a un producte de la carta. Per a definir un comentari per tant, necessitarem un **usuari** que realitza el comentari i un **producte** al qual se li associa aquest comentari. Per últim i com és lògic, aquest objecte abstracte també tindrà el **text del comentari**. A la figura 56 es pot observar la representació gràfica de les dades d'un comentari a l'aplicació.

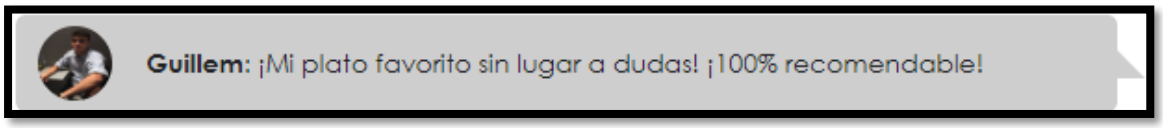


Figura 56: *Visualització d'un comentari d'un producte de la carta.*

"Event"

Objecte abstracte que representa un **esdeveniment** del local. Aquests esdeveniments es poden generar únicament des del panell d'administració per un usuari administrador i seran mostrats a la pantalla d'inici de l'aplicació. Per a definir un esdeveniment en el sistema, l'administrador haurà d'introduir un **títol** i una **descripció** que detallin l'esdeveniment, de forma clara, concisa i entenedora per a l'usuari. D'altra banda també se li haurà d'associar una **imatge** relacionada amb l'esdeveniment i una **data d'inici** i una **data de finalització**.

Com s'ha mencionat anteriorment, per a la visualització dels esdeveniments es fa servir un patró de disseny anomenat "*Carousel*".

Com s'observa a la figura 57 cada diapositiva del "*Carousel*" està continguda per un esdeveniment i es mostra la informació de cadascun juntament amb una imatge de fons escollida per a l'administrador per a representar millor el tòpic de l'esdeveniment.

Figura 57: *"Carousel" d'esdeveniments de la pàgina principal de l'aplicació on actualment només hi ha un esdeveniment programat.*

Ingredient

Objecte abstracte que fa referència a un ingredient del local. Els ingredients són generats per l'administrador des del panell d'administració, i s'associen a diferents productes. Per a la definició d'un ingredient cal indicar el nom d'aquest ingredient, i un preu. Aquest preu s'ha contemplat a causa de la funcionalitat de creació de pizzes per a l'usuari. S'afegeixen ingredients a la pizza, el preu total d'aquesta ha d'anar incrementant segons l'ingredient que s'hagi afegit. A la figura 58, es pot observar la representació gràfica dels ingredients en l'aplicació.

Figura 58: *Visualització dels ingredients d'un producte.*

"Order"

Objecte abstracte que representa una **comanda** dins l'aplicació. Aquesta comanda és generada per l'usuari i gestionada per l'administrador des del panell d'administració. L'**usuari** per a generar una comanda, simplement ha d'anar navegant per la carta i afegint a la comanda els **productes** que trobi del seu interès. Un cop completada la comanda, l'usuari ha d'indicar a quina **direcció** vol que sigui enviat el contingut d'aquesta comanda, o bé si és ell mateix que ho passarà a recollir en el local. L'usuari veu que la comanda té un **preu total**, que ve donat del contingut i d'aquest preu total s'extreu el total de **punts de recompensa** que dóna l'aplicació a l'usuari un cop es paga la comanda. A aquesta comanda, també se li associa un **estat**, que d'igual manera que per a les reserves, ens serveix per indicar a l'usuari en temps real quin és l'estat de la seva comanda.

Una comanda també conté una variable interna (desconeguda per a l'usuari), que indica si la comanda ha sigut **pagada** o no. Des de l'administració, es pot observar aquest estat de la variable, i un cop l'usuari administrador marca una comanda com a pagada, se li entreguen els punts de recompensa a l'usuari automàticament.

Finalment, també s'associa una **data** a aquesta comanda per tal de tenir més informació sobre aquesta a l'hora de registrar-la i mostrar-la a l'administració. Aquest atribut no tindrà utilitzat directe en l'aplicació, però sí que tindrà una funcionalitat important de cara al flux de treball de l'empresa. Totes les comandes que es van rebent al panell d'administració, s'ordenen segons aquesta data que emmagatzemem. D'aquesta manera s'atendran les comandes en l'ordre real en el qual es generen, assegurant així que els usuaris que utilitzin l'aplicació per a realitzar una comanda seran atesos en el seu torn.

Les comandes tenen dos tipus de visualitzacions dins l'aplicació. Una per a l'usuari que la realitza i una altra per als usuaris administradors. A la visualització per als administradors, com s'aprecia a la figura 59, es vol detallar la màxima informació de la comanda en una fila de la taula per a facilitar la lectura als administradors. En canvi a la visualització per usuari, com s'observa a la figura 60 s'assoleix un aspecte més atractiu on l'usuari pot veure el contingut, les dades de la comanda, repetir-la o bé eliminar-la.

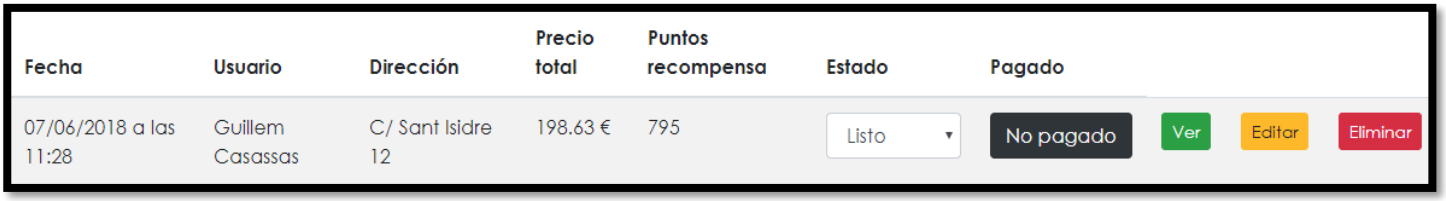


Figura 59: *Visualització d'una comanda des del panell d'administració.*

Figura 60: *Visualització d'una comanda realitzada i entregada des de la pàgina personal d'un usuari.*

"Product"

Objecte abstracte associat a un **producte** de la carta. Els productes són la base de l'aplicació. Els usuaris els veuen, opinen, creen de nous (només pizzes) i poden afegir-los a la seva comanda lliurement, mentre que els administradors poden crear-ne de nous, modificar els que ja estan creats i eliminar-los. Un producte té associat diferents **ingredients**, que seran marcats per l'usuari administrador en crear el producte. Per a definir un producte a l'aplicació es necessiten diverses dades; Primerament un **nom**, una **descripció** i una **imatge** que ensenyin i defineixin com és el producte al client. Cal indicar també de quin **tipus de producte** es tracta, ja que posteriorment serà classificat d'una manera o una altra en la carta, tenint en compte que, el tipus de producte "pizza", té **3 preus diferents**; un per a cada mida, mentre que la resta només en té **un d'únic**.

Internament un producte té més atributs que l'administrador pot gestionar. En primer lloc, l'administrador pot indicar si un producte està actualment **esgotat** o no. En cas d'estar esgotat, aquest producte mai podrà ser afegit a cap comanda de cap usuari. També té associat un nombre de "***Likes***" o "*m'agrada*", que s'indiquen a la carta i depenen totalment de la interacció dels usuaris. D'altra banda també s'associa un **usuari** al producte, sabent que els productes de la carta pertanyen a l'usuari administrador. Aquest fet es deu a que un usuari pot crear una pizza, i tots els productes que no són creats per l'administrador seran productes creats per usuaris. D'aquesta manera obtenim tots els productes de la comunitat. Relacionat amb aquest tema de creacions de pizza, ens apareix un últim atribut del producte, per saber si és **públic** o no. Lògicament tots els productes creats des de el panell d'administració seran públics i s'inicialitzaran com a disponibles (no esgotats), però es contempla aquest atribut degut a que, quan usuari crea una pizza, té la opció d'indicar si vol compartir-la amb la resta de la comunitat, o bé vol simplement tenir-la guardada per a ell.

La figura 61 és la representació visual d'un producte dins l'aplicació, on es mostren tots els atributs mencionats anteriorment que són rellevants i necessaris per a l'usuari, com els preus, els ingredients, els "m'agrada" i els comentaris.

Figura 61: *Visualització d'un producte de la carta*

"OrderProduct"

Objecte abstracte associat a un **producte que ha sigut afegit a una comanda**. Quan un usuari afegeix un producte a la comanda, ha d'indicar a l'aplicació més dades, com poden ser la **quantitat** desitjada d'aquell producte i en el cas de les pizzes, quina **mida** és el desitjat: petita, mitjana, gran o "*brusquetta"*.

Internament, sense que l'usuari se n'adoni, l'aplicació quan gestiona les comandes no treballa amb els objectes de productes, sinó que treballa amb aquests objectes. Com es veu a la figura 62, l'usuari visualitzaria aquest resultat en la seva comanda, i tindria la possibilitat d'editar o eliminar els productes continguts a la sema comanda.

Com s'ha mencionat anteriorment, al no treballar sobre els productes aquests no es modificarien. Es treballa sobre aquests nous objectes que a mida que l'usuari els va afegint a la comanda es van emmagatzemant al "*Local Storage"* del navegador.

Figura 62: *Visualització d'un producte afegit a la comanda. On s'aprecia la quantitat desitjada d'aquell producte i en el cas de les pizzes, el seu tamany.*

"User"

Objecte abstracte associat a un usuari de l'aplicació. A aquest objecte, a part de les dades bàsiques d'un usuari com poden ser el **nom d'usuari** i la **contrasenya**, l'hi afegirem els atributs que ens convinguin per tal d'assolir totes les funcionalitats requerides pel "*Product Owner".*

Els usuaris es generen únicament quan una persona es registra mitjançant el formulari de registre de l'aplicació. En aquest formulari, la persona en qüestió ens està donant diverses dades seves, com per exemple; El **nom**, **cognoms**, **adreça electrònica** i **telèfon**. També donem l'opció a l'usuari de si **vol rebre correus** de l'aplicació informant-lo de promocions i esdeveniments que poden generar els administradors en qualsevol moment.

Aquestes serien les dades que la persona que es vol registrar ens facilita. Internament, un objecte abstracte usuari té altres atributs, com poden ser els **punts**. Inicialment, un usuari que s'acaba de registrar comença amb 0 punts. Un altre atribut extra que té un usuari és un indicador de si és o no **administrador**. Aquest indicador permetrà a l'usuari accedir o no al panell d'administració de l'aplicació.

A l'usuari també se li associa una **foto de perfil** per defecte, que posteriorment podrà canviar per a qualsevol que ell/a desitgi. També conté un atribut on es defineix la seva **direcció** on realitzar els enviaments de les comandes, que pot ser introduït o no per l'usuari dins l'aplicació per a facilitar el procés de realitzar una comanda.

L'usuari en tot moment pot gestionar les seves configuracions, a través del seu perfil. L'accés al perfil de l'usuari és molt ràpid en qualsevol pàgina de l'aplicació, ja que té una connexió directa des de la capçalera, que s'ha vist prèviament en la figura 41.

"Voucher"

Objecte abstracte associat a un cupó de l'aplicació. Totes les transferències i bescanvis de punts, es realitzen mitjançant la generació de cupons per part de l'usuari i posteriorment, la validació d'aquest anterior cupó per un usuari administrador. Per a crear un cupó, necessàriament l'usuari ha de tenir una quantitat de punts superior a 0, d'altra manera el cupó no podrà ser generat. Els atributs que defineixen el cupó són; Primerament un codi únic de 6 dígits, generat en el servidor, que s'entrega a l'usuari. Aquest codi és el que posteriorment s'entregarà a l'usuari administrador per a realitzar la transacció. També té associat una quantitat de punts, que és indicada per l'usuari a l'hora de generar el cupó.

Internament es generen uns altres atributs en el servidor, que bàsicament es creen per a qüestions de seguretat. Primerament tindrem una data de caducitat, que es genera a partir de la data actual i els hi atribueix 10 dies de vida. L'altre atribut generat internament és un indicador de si ja ha sigut utilitzat o no. Ja que els cupons són únics i només poden ser utilitzats una vegada.

Com s'observa en la figura 63, el cupó indica la seva quantitat en punts i en euros per a facilitar la conversió a l'usuari. S'indica la data de caducitat i el temps restant en dies ressaltat de forma que l'usuari sàpiga en tot moment quan deixarà de ser vàlid el cupó.

Figura 63: *Visualització d'un cupó des del perfil d'un usuari.*

Relacions

Per a la codificació i per al disseny de la base de dades, han sigut necessàries establir unes relacions entre els models lògics. Amb l'ajuda del "*framework"* Laravel utilitzat, que ja ens proporciona els mètodes per a establir les relacions entre aquestes classes lògiques, s'ha hagut de definir una estructura que pot ser visualitzada més fàcilment amb el diagrama del disseny de la base de dades, que es troba més endavant de la memòria del projecte.

A continuació s'explicaran les relacions realitzades i l'argumentació d'aquestes.

Relació 1. Usuari – Producte

Aquesta relació es podria considerar doble, ja que d'ella obtenim dos tipus d'informació.

- Si ens referim a qui és el creador del producte, aquesta informació pot ser trobada dins del mateix objecte del producte, ja que estem parlant d'una relació "*One-To-One": Un producte pertany a un usuari.*

- L'altra informació que obtenim gràcies a aquesta relació és si a l'usuari li ha agradat un o molts productes de la carta. En aquest cas estem parlant d'una relació "*Many-To-Many*"*: Molts usuaris els hi poden agradar molts productes*, fet que ens implica crear dues taules a la base de dades per emmagatzemar la informació dels usuaris i els productes, afegint una altra taula pivot per a relacionar quins productes han sigut agradats per quins usuaris.

Relació 2. Usuari – Comanda

Aquesta relació és necessària per a relacionar les comandes generades a l'aplicació, amb l'usuari que les ha generat. En aquest cas estem parlant d'una relació "*One-To-Many": Un usuari pot tenir moltes comandes*. No seria una relació de "*Many-To-Many"*, ja que una comanda sempre té un únic usuari com a propietari. Per tant, amb una simple taula en la base de dades per emmagatzemar les comandes seria suficient, sempre i quant a cada comanda ens guardem a quin usuari pertany.

Relació 3. Usuari – Reserva

Aquesta relació és necessària per a relacionar les reserves generades a l'aplicació, amb l'usuari que les ha generat. Aquest cas, idèntic al cas anterior, es tracta d'una relació "*One-To-Many": Un usuari pot tenir moltes reserves*. Com s'ha vist anteriorment, seria erroni pensar que aquesta és una relació "*Many-To-Many",* ja que una reserva sempre pertany a un sol usuari. D'aquesta forma ens estalviem crear la taula pivot característica en les relacions molts a molts. Com en el cas anterior, amb una simple taula on emmagatzemar les reserves i una columna per indicar l'usuari al qui pertany seria suficient.

Relació 4. Comentari – Usuari i Producte

En aquest cas estem parlant d'una relació ternària, ja que per a l'existència d'un comentari, és imprescindible la implicació d'un producte i un usuari. Per a l'emmagatzematge d'aquesta informació crearem una taula on guardarem els comentaris amb dues columnes per a saber a quin producte i a quin usuari pertany el comentari.

Relació 5. "*Voucher*" – Usuari

Entenem per "*Voucher"* un cupó de l'aplicació. Com s'ha mencionat anteriorment, un cupó sempre tindrà un usuari al qui pertany. Tot i així un usuari pot tenir molts cupons, per tant estem parlant d'una relació "*One-To-Many": Un usuari pot tenir molts cupons*. Per tant, com s'ha vist anteriorment en aquests tipus de relacions, es crearà una taula per emmagatzemar tots els cupons de tots els usuaris, i es guardarà en un camp de cadascun d'ells l'usuari al qual pertanyen.

Relació 6. Producte – Ingredient

Aquesta relació és necessària per a relacionar els productes de l'aplicació, amb els seus respectius ingredients. En aquest cas estem parlant d'una relació "*Many-To-Many": Molts productes contenen molts ingredients*. Seria un error confondre-la amb una "*One-To-Many"*, ja que realment un ingredient pot pertànyer a molts productes. Per tant, com hem vist també anteriorment amb la relació d'usuari i producte, caldrà crear una taula per a cada model, i una altra afegida com a pivot per a relacionar quins ingredients pertanyen a quins productes.

Relació 7. Usuari – Ingredient

Aquesta relació és poc intuïtiva però necessària per a assolir una de les funcionalitats de l'aplicació. Per tal de permetre a l'usuari indicar quins ingredients no li agraden, per així posteriorment informar-lo a l'hora de veure la carta i avisar a l'administració a l'hora d'aquest usuari realitzar una comanda, ens fa falta declarar aquesta relació en l'aplicació. En aquest cas també estem parlant d'una relació "*Many-To-Many": Molts usuaris els hi poden no agradar molts ingredients.* Com hem vist en casos anteriors, haurem de crear ambdues taules per als models més una taula pivot per a relacionar quins usuaris han marcat com a negatiu els ingredients.

3.2.4 Controladors

Respectant el patró MVC, tindrem un conjunt de classes que efectuaran com a controladors de l'aplicació. La funció d'aquestes classes, és recollir una sèrie de dades relacionades amb els models lògics definits que hem explicat prèviament i realitzar una sèrie de funcionalitats amb aquestes dades. Ja poden ser comprovacions, validacions, càlculs, eliminacions, refrescos, etc...

Aquests càlculs dels controladors es realitzaran al servidor i s'enviaran els resultats en formats determinats cap a la vista perquè es processin les dades que es mostraran per pantalla a l'usuari. El format en que s'empaqueten les dades un cop han estat processades en els controladors del servidor és JSON.

El diagrama de flux on intervenen els controladors s'ha representat en la següent figura:

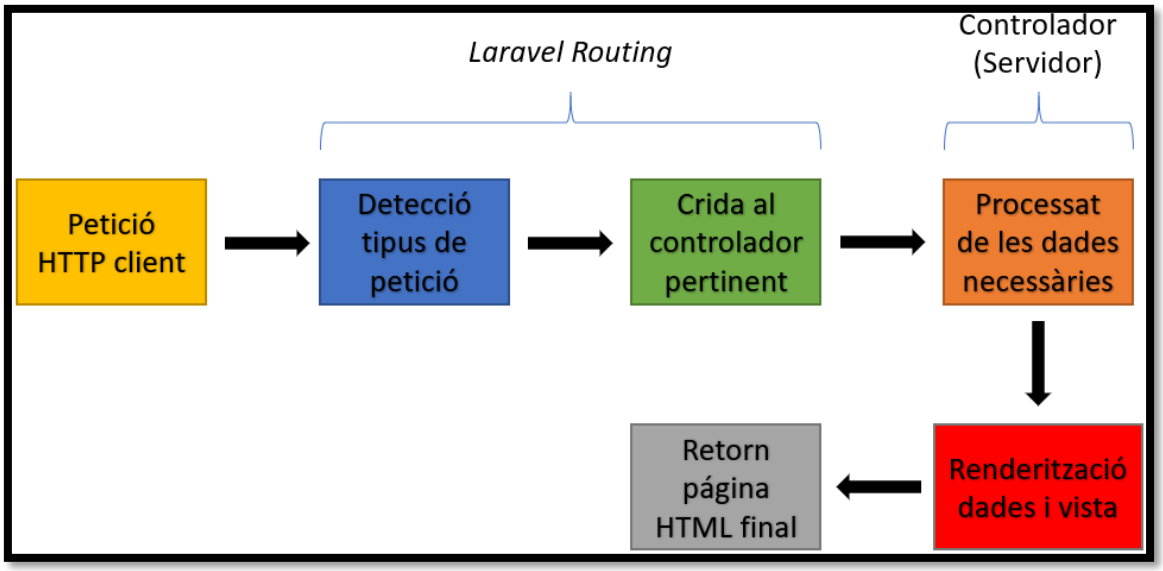


Figura 64: *Diagrama de flux de funcionament de l'aplicació al rebre una petició d'un client.*

Controladors d'autenticació

Per a gestionar totes les funcionalitats relacionades amb les sessions, ja pot ser inici de sessió com registre d'un nou usuari, fins poder reiniciar una contrasenya o l'adreça electrònica, són necessaris un grup de controladors que hem anomenat controladors d'autenticació.

El mateix "*framework*" Laravel ja incorpora aquests controladors de manera automàtica, però si volem implementar totes les funcionalitats, cal realitzar una sèrie de modificacions a aquests controladors per adaptar tots els continguts desitjats.

A continuació es comentaran els diferents controladors d'autenticació i les modificacions que s'han realitzat amb la seva argumentació:

Controlador d'inici de sessió

L'inici de sessió, tal com ve prèviament definit a Laravel, es realitza mitjançant el correu electrònic de l'usuari i la contrasenya amb la qual s'ha registrat. Aquest controlador ja s'adapta perfectament als nostres objectius. Tot i així, cal fer una petita modificació per tal d'adaptar l'inici de sessió a la nostra metodologia.

A part d'indicar correctament l'adreça electrònica i la contrasenya, l'usuari per iniciar sessió també ha hagut d'haver confirmat la seva adreça electrònica. És per això que en aquest controlador d'inici de sessió, farem una petita modificació comprovant si l'indicador de si l'adreça ha estat confirmada o no.

Controlador de registre

Donat que Laravel per defecte simplement registra l'adreça electrònica de l'usuari i la contrasenya que aquest introdueixi, en aquest controlador sí que s'hauran de realitzar modificacions. Donat que en el nostre cas volem emmagatzemar més dades relacionades amb l'usuari, com poden ser el nom, els cognoms, el número de telèfon, etc... haurem de realitzar modificacions en el controlador:

- Validació de les dades rebudes del formulari de registre de l'usuari.
  Primerament haurem d'afegir els camps que esperem rebre posteriorment al registre de l'usuari, i el format esperat d'aquests, per tal d'assegurar la validesa de les dades.

- Registre de l'usuari a l'aplicació.
  Incloem les noves dades que hem volgut associar a l'usuari, que prèviament han estat validades pel mètode de validació explicat anteriorment. En aquest punt també revisem si l'usuari desitja o no rebre correus de l'aplicació informant d'esdeveniments i promocions.

  Amb tota aquesta informació de l'usuari, l'enregistrem a la base de dades però amb l'indicador de confirmació de l'adreça electrònica com a "no confirmada". Posteriorment a la creació de l'usuari, dins d'aquest mateix mètode, cridarem a la funció que farà enviar un correu de confirmació a l'usuari, on podrà confirmar la seva adreça de correu electrònic i per tant, iniciar sessió. Aquest correu electrònic fa servir una de les vistes del conjunt de "*Emails"* mencionat anteriorment.

- Confirmació de l'adreça electrònica de l'usuari.
  Quan el sistema envia el correu de confirmació a l'adreça que l'usuari ha introduït prèviament en el formulari de registre, aquest correu incorpora un "*link"* que conté el "*token"* inicial de l'usuari en l'adreça. Sabent que aquest no podrà canviar, ja que fins que no inicii sessió aquest no es refresca, aquest controlador serà cridat quan l'usuari premi sobre aquest "*link"*, i rebrà com a paràmetre el "*token"*, contingut en l'adreça.

Treballant en local, un exemple de "*link"* de validació podria ser el següent:

http://localhost/impronta/public/register/confirm/**9Gn6blK85NGEcNB4DqvJkmAu7vBbCeXqJpFBi8eh**

El que està marcat en negreta és el "*token"* de l'usuari, que el rebrà el mètode de confirmació per paràmetre.

Un cop dins d'aquest mètode, buscarem l'usuari amb aquest "*token"* i el marcarem com l'usuari ja ha confirmat la seva adreça de correu electrònic i el redirigirem a la pàgina d'inici de sessió. En cas que no es detecti el "*token*", farem la mateixa redirecció però informant a l'usuari amb un missatge d'error el que ha sortit malament.

Controlador de canvi de contrasenya

En aquest cas, Laravel ja incorpora la funcionalitat de canvi de contrasenya, però no s'adapta del tot a les nostres necessitats, per tant aquí també haurem de realitzar una sèrie de modificacions.

- Enviament de nova contrasenya.
  Rebem les dades del formulari on es recull la informació necessària per a enviar un "*link"* a l'usuari on podrà establir de nou la seva contrasenya. Per a qüestions de seguretat, l'usuari ha d'introduir el seu correu electrònic (que és l'única dada que es demana en aquest formulari) i des d'allà és on podrà accedir a la pantalla de canvi de contrasenya.

  Validem també l'adreça que ha introduït l'usuari en el formulari i en cas de ser correcta, envia un correu amb un "*link"* que permetrà a l'usuari canviar la seva contrasenya. En cas de ser incorrecte, s'informarà a l'usuari amb un missatge d'error.

- Validació de "*token"* de l'usuari.
  La validació es realitza quan l'usuari prem el "*link"* que se li ha enviat per correu electrònic. Rep el "*token"* de l'usuari per paràmetre i valida que existeixi un usuari amb la cadena rebuda. En cas correcte envia a l'usuari a la pàgina on podrà indicar la seva nova contrasenya i en cas incorrecte s'ensenyarà un missatge d'error informant a l'usuari.

- Canvi de contrasenya en base de dades.
  Es realitza l'actualització de la contrasenya de l'usuari a la base de dades. Rep les noves contrasenyes que ha indicat l'usuari en el formulari (en rep dues, per a reforçar la seguretat). Simplement es comprova que les dues contrasenyes siguin les mateixes, i en cas correcte es realitza l'actualització a la base de dades, s'informa a l'usuari i se'l redirigeix a la pàgina per accedir. En cas incorrecte es mostrarà un missatge d'error informant a l'usuari.

Controladors del model

Els següents tipus de controladors, són els que seran cridats mitjançant peticions a determinades rutes per tal de gestionar les dades necessàries per a assolir certes funcionalitats.

Com bé indica el nom d'aquest conjunt de controladors, trobarem un per a cada objecte del model que tinguem a l'aplicació, és a dir:

- Controlador de reserves
- Controlador de comandes
- Controlador d'esdeveniments
- Controlador d'ingredients
- Controlador de productes
- Controlador d'usuaris
- Controlador de comentaris
- Controlador de "*vouchers"* o cupons

Dins de tots aquests controladors, trobarem una sèrie de funcions que es repeteixen:

- **Obtenció de tot el conjunt de dades del model (*GET)*:**
  Aquest mètode o funció serà cridat quan desitgem obtenir tot el conjunt de dades d'aquell model de la base de dades. Per exemple, obtenir tots els productes.
- **Emmagatzematge d'un nou objecte del model (*POST)*:**
  Aquest mètode o funció ens servirà per guardar noves dades d'aquell model a la base de dades. Fa una comprovació de la validesa de totes les dades necessàries per crear aquell objecte prèviament a emmagatzemar-lo.
- **Actualització d'un objecte existent del model (*PUT*):**
  Aquest mètode o funció ens permetrà editar els atributs d'un objecte d'aquell model determinat a la base de dades. Com en el cas anterior, es realitzarà una comprovació de la validesa de les noves dades a modificar de l'objecte prèviament a editar-lo.
- **Eliminació d'un objecte del model (*DELETE)*:**
  Donat l'atribut identificador de l'objecte del model, aquest serà buscat a la base de dades i eliminat.

D'altra banda, cada controlador tindrà altres mètodes o funcions requerides per tal d'assolir totes les funcionalitats plantejades a l'inici del projecte.

Per exemple, el controlador de productes tindrà totes les funcions mencionades anteriorment, però també tindrà altres mètodes com per exemple per obtenir les pizzes que han sigut creades per un determinat usuari. Com d'igual manera el controlador de cupons tindrà funcions per validar i revisar les transaccions amb els cupons de l'aplicació.

En aquests controladors també s'assegura la robustesa i seguretat de l'aplicació, ja que és aquí on es fan les comprovacions de les dades i si aquestes són vàlides en ser rebudes i/o retornades. Per exemple, al rebre una comanda realitzada per un usuari, en el controlador pertinent es re calcula el preu d'aquesta comanda per assegurar que totes les dades provinents del client no han estat malmeses.

"Middlewares"

Per a donar més robustesa a l'aplicació, s'ha fet ús dels *"Middlewares"* que per defecte venen incorporats amb el *"framework"* Laravel. Aquests ens serveixen per assegurar que l'usuari que està fent una petició determinada al servidor, tingui els privilegis suficients per a fer-la. Com s'observa a la figura 65, la petició haurà de ser avaluada pels diferents *"Middlewares"* que contingui l'aplicació prèviament a ser atesa pel servidor i enviar la resposta.

Per a assolir totes les funcionalitats, s'han creat *"Middlewares"* personalitzats, que veurem a continuació.

Aquests *"Middlewares"* s'associen a les URIs declarades en el *"Laravel Router"*, per tal de crear una jerarquia d'accessos als diferents recursos i pàgines de l'aplicació.

- **"*Middl*eware" *per a convidats*:**
  En aquest *"Middleware"* es comprova que l'usuari actual que fa la petició no tingui una sessió activa ni es relacioni amb cap conta d'usuari.

- **"*Middl*eware" *per a usuaris autenticats*:**
  En aquest *"Middleware"* es comprova que l'usuari que està fent la petició sigui un usuari registrat i amb una sessió activa en la nostra aplicació. Aquest *"Middleware"* s'aplica a totes les pàgines i crides que es realitzen a la pàgina del perfil de l'usuari.

- **"*Middl*eware" *per a usuaris administradors*:**
  En aquest *"Middleware"* es comprova que l'usuari que està fent la petició sigui un usuari administrador i amb una sessió activa en la nostra aplicació. Aquest *"Middleware"* s'aplica a totes les pàgines i crides que es realitzen al panell d'administració.

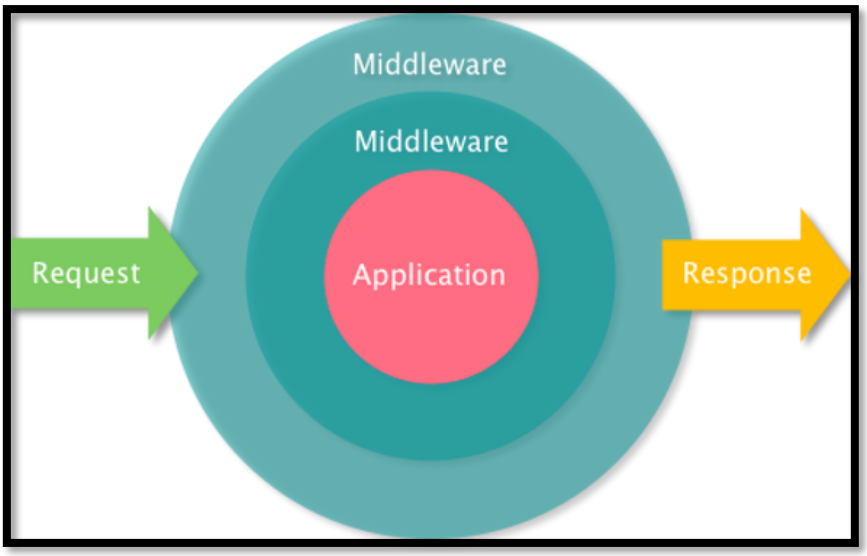


Figura 65: *Diagrama que representa les capes de seguretat que ha de recórrer una petició per a ser avaluada i contestada.*

Controlador de pàgines principal

Aquest controlador és el més simple de tots, i és el que s'encarrega de redirigir a totes les pàgines de l'aplicació, incloent-hi el panell d'administració. En alguns casos, si és necessari, envia dades compactes amb la vista, perquè posteriorment en el procés de renderització es facin els càlculs pertinents.

3.3 Diagrames

A continuació es resumirà tot el contingut argumentat anteriorment del disseny de l'aplicació mitjançant diagrames que permeten entendre millor el disseny de l'aplicació. Per una part podrem veure gràficament l'arquitectura de la base de dades amb les seves relacions. També podrem observar les diferents interaccions i rols que contemplem en l'aplicació en el model de domini. Per últim podrem analitzar les diferents vistes i els diferents enllaços interns que ens permeten navegar d'una pàgina a una altra gràcies al diagrama web.

3.3.1 Diagrama de la base de dades

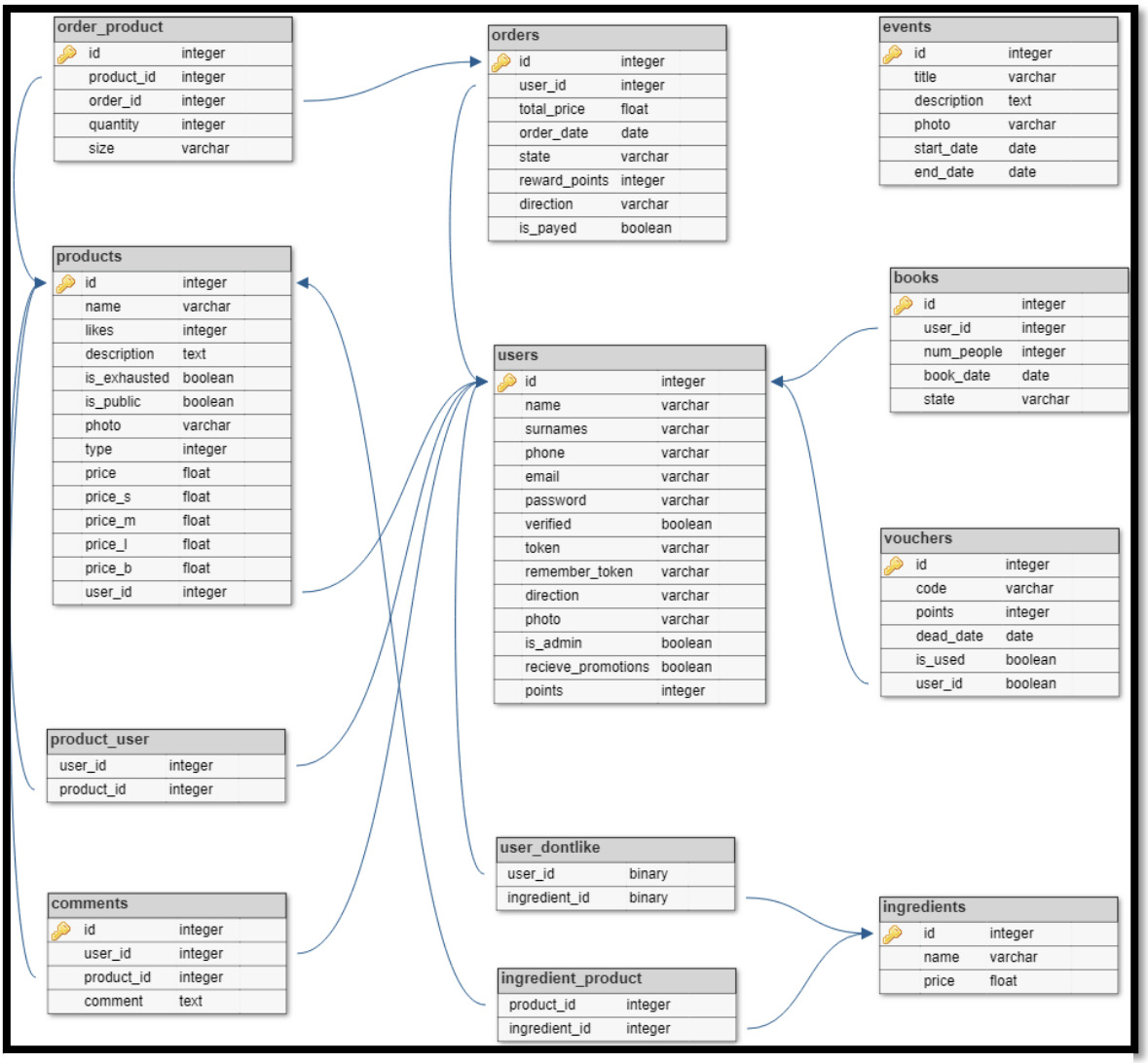


Figura 66: *Diagrama de la base de dades de l'aplicació*

3.3.2 Model de domini

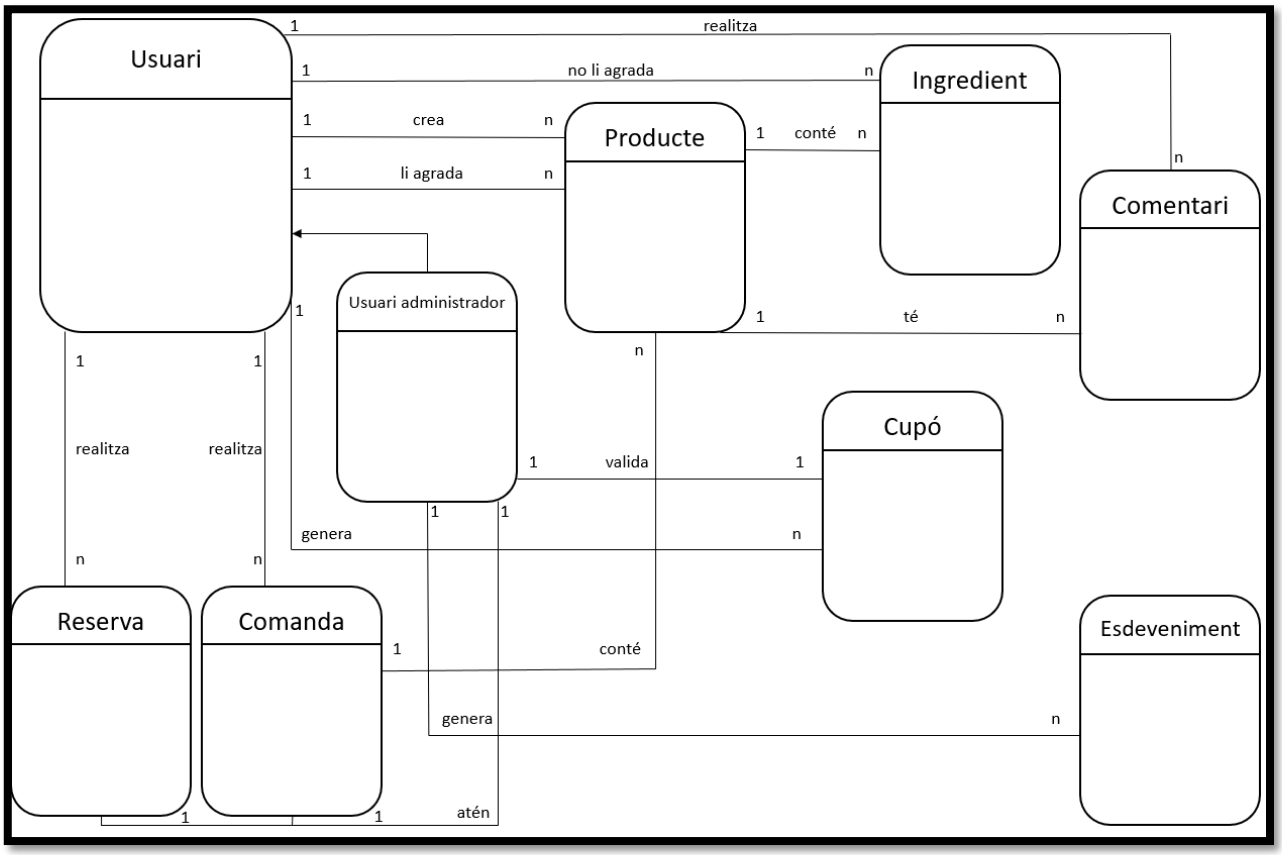


Figura 67: *Diagrama del model de domini de l'aplicació*

3.3.3 Diagrama web

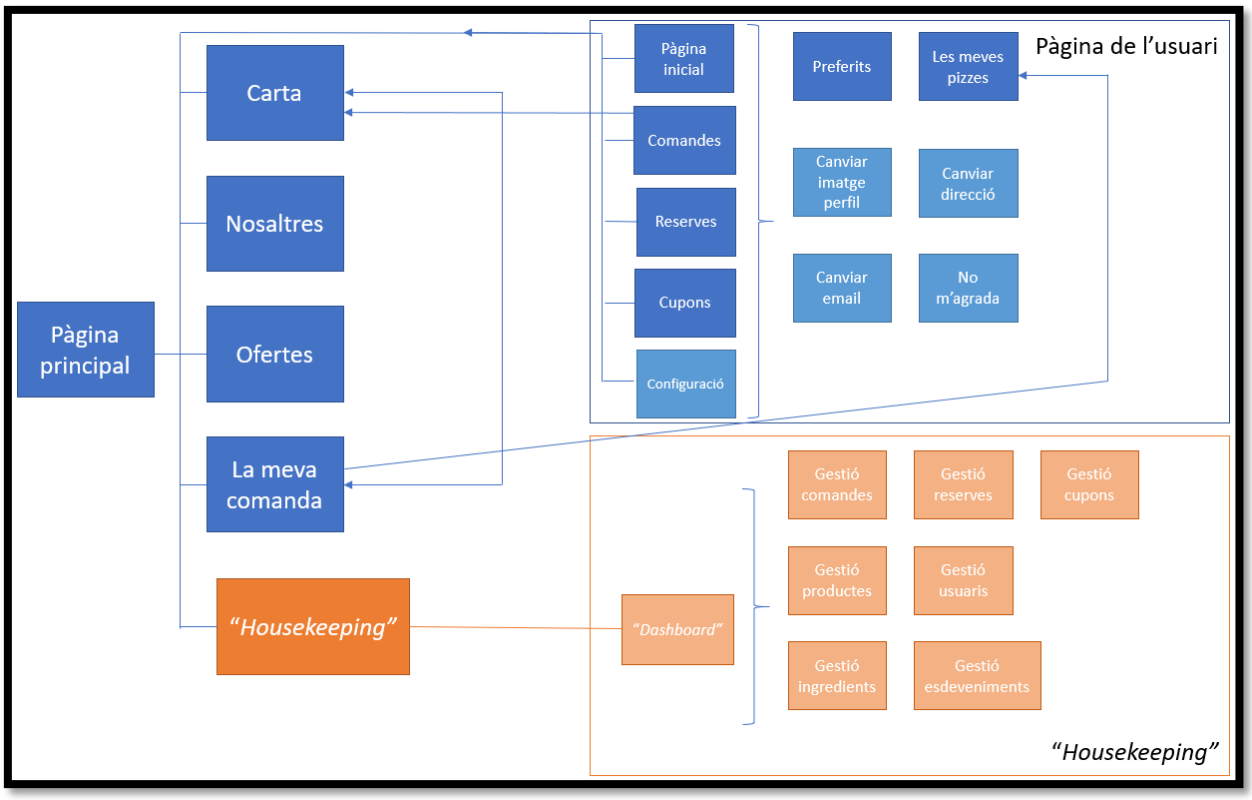


Figura 68: *Diagrama web de l'aplicació*

4. Implementació

En aquest apartat de la documentació s'analitzarà la implementació de les funcionalitats principals de l'aplicació.

4.1 Recomanador de productes

A continuació analitzarem el recomanador de productes implementat a l'aplicació, concretament com es pot veure a la figura 69, a l'apartat on l'usuari pot veure la seva comanda. Aquest apartat de suggeriments consta de dos tipus de recomanacions: les recomanacions del local, on el "*xef*» o administradors del local escullen tres recomanacions fixes que creuen que són els productes potencials del negoci i poden agradar a la majoria. A la part superior es trobaria la recomanació personalitzada per a l'usuari, que s'explicarà en detall a continuació.

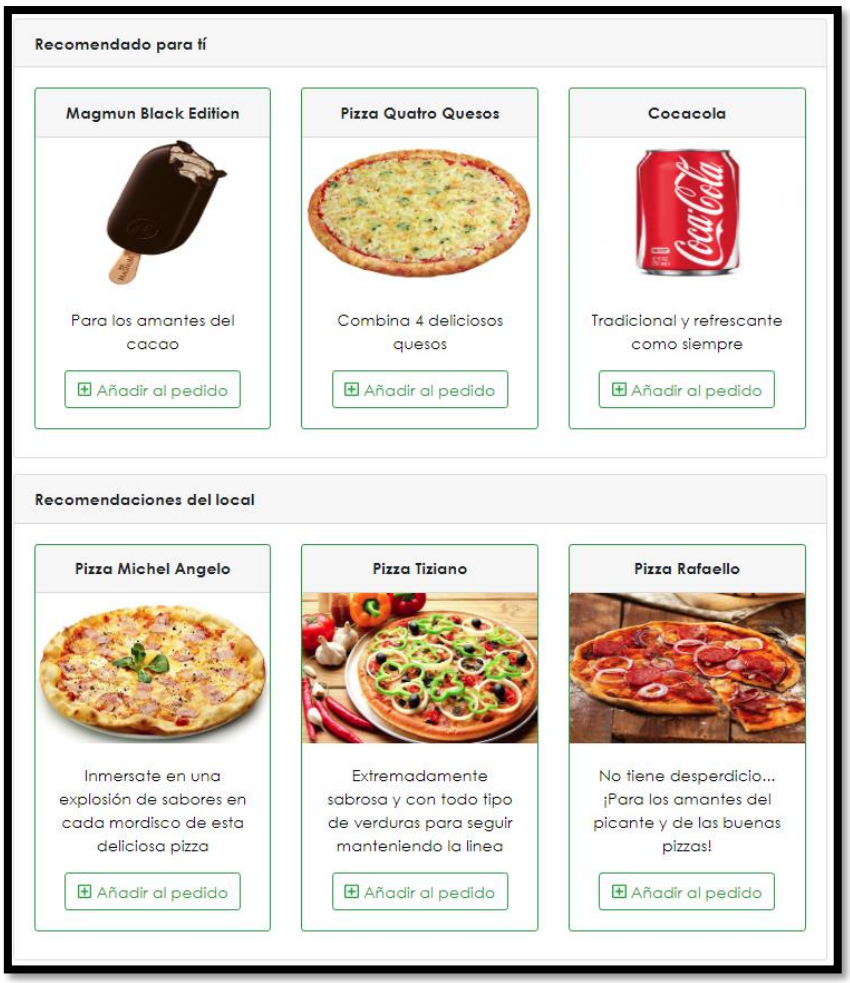


Figura 69: *Apartat de recomanacions en la comanda de l'usuari.*

El procés que es realitza internament, d'ençà que el client realitza la petició per veure la seva comanda fins que s'acaba retornant la pàgina amb les recomanacions personalitzades, s'ha dividit en 4 passes, com es pot observar a la figura 70:

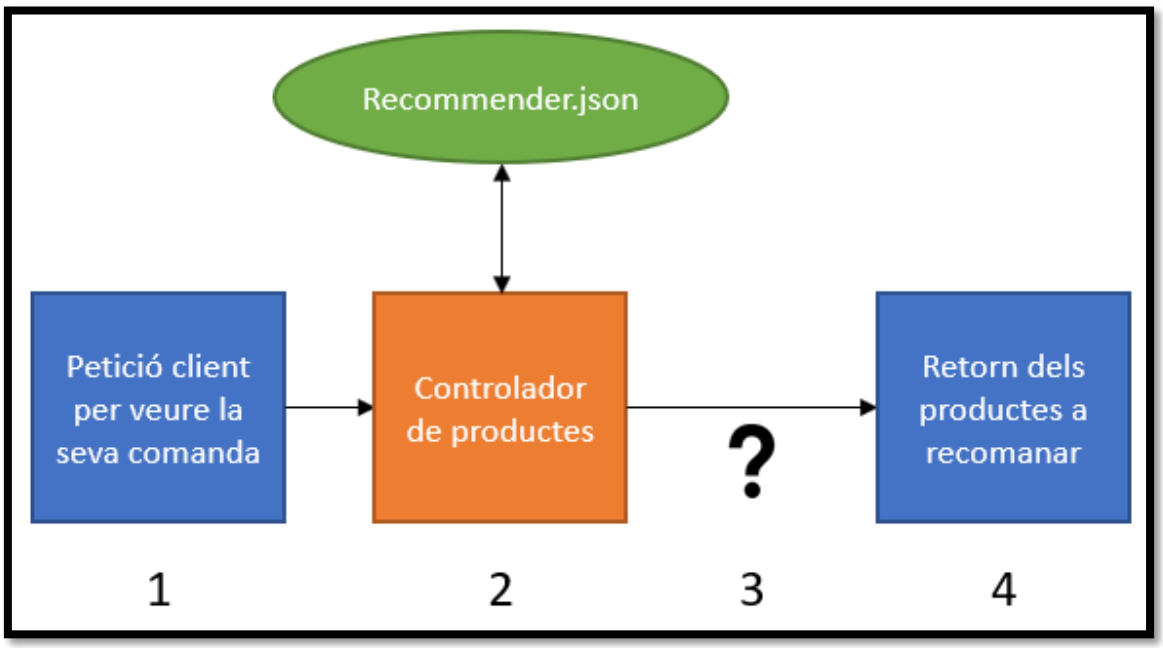


Figura 70: *Diagrama de flux de funcionament d'una petició en la que recomanem productes.*

Com s'ha comentat, el primer pas el realitza el client en visualitzar la seva comanda. Quan es detecta una crida d'aquest tipus, mitjançant "*Laravel Routing*" cridem al controlador de productes, al mètode que gestionarà els productes a recomanar donat el client que realitza la petició.

La recomanació es basa en els històrics de comandes dels altres usuaris majoritàriament i en una petita part pels gustos del client en qüestió. Com es veu a la figura 70 en el pas 2, el controlador de productes té accés a la lectura del fitxer on s'emmagatzemen les dades del recomanador. Totes les decisions per a realitzar la recomanació es troben en aquest pas 2, és a dir, entre el fitxer JSON que conté les dades de les recomanacions anteriors d'altres usuaris i el pròxim pas 3 que també es realitza al controlador de productes. Amb això el que es vol arribar a assolir és rapidesa i eficiència, evitant així haver de fer consultes a altres controladors ni a la base de dades. Per tant en el pas 2, el controlador recull els productes que s'han triat juntament amb els altres productes que coincideixen en la comanda actual de l'usuari en qüestió. Per exemple, si diversos usuaris han realitzat una comanda amb la pizza "*Michel Angello*" juntament amb una "*Cocacola",* i l'usuari actual té a la comanda una pizza "*Michel Angello*", el controlador de productes obtindrà com a resposta de recomanació la "*Cocacola".*

Tot i així, aquest producte (que podrien ser més d'un, si les comandes no fossin únicament de 2 productes) no és directament recomanat a l'usuari, s'haurà de fer un altre pas de comprovació abans de realitzar la recomanació final, com es pot veure en el pas 3 de la figura 70.

Com a últim pas abans de donar la recomanació final a l'usuari, per tal d'acorar la recomanació amb els gustos de l'usuari, farem un petit procés de comprovació que es veu a la figura 70 en l'apartat 3. Simplement mirarem els ingredients que l'usuari en qüestió ha indicat com a negatius, i compararem aquests ingredients amb els productes que anàvem a recomanar. Tots els productes que anaven a ser recomanats i que contenen ingredients que han sigut marcats com a negatius no seran recomanats.

Passant aquest últim filtre, ja obtindríem tots els productes finals que si poden ser recomanats. Aquests s'obtindran mitjançant "*Vue JS*" en carregar la pàgina i es renderitzaran juntament amb la resta de la vista, per tal que l'usuari pugui afegir-los a la seva comanada còmodament.

Finalment quan es realitza una comanda, s'actualitza el fitxer JSON afegint les noves parelles de productes que han estat escollides (sense repeticions), per a actualitzar l'històric i tenir més informació emmagatzemada per a futures recomanacions.

4.2 Generador i validador de cupons

Dins l'aplicació és necessària la funcionalitat per a bescanviar els punts per euros reals, per donar sentit al fet d'acumular punts utilitzant l'aplicació web. Aquesta funcionalitat és pràcticament idèntica a una transacció, per tant cal assegurar el seu funcionament tant per a l'usuari administrador com pel client.

Com s'ha vist anteriorment, l'usuari pot generar cupons fàcilment des del seu perfil i mantenir-los fins que caduquin. En el procés de generació del cupó, tant com a la vista com en el servidor, es comprova que l'usuari estigui generant un cupó amb una quantia de punts superior a la real. En la vista ho revisarem mitjançant el "*Seekbar*" que és la barra horitzontal mòbil que utilitza l'usuari per indicar la quantitat de punts que tindrà el cupó i que es pot veure a la figura 71. Aquesta ens permet donar-li un valor màxim de selecció, que nosaltres farem que sigui igual al total de punts del client. D'aquesta manera l'usuari no podrà introduir un valor major al qual posseeix, però de totes maneres, en tractar-se d'una transacció que implicarà costos per al negoci que hi ha darrere, en el servidor abans de generar el cupó es torna a revisar si la quantitat de punts que s'estan donant al cupó és correcte segons l'usuari.

Com es veu a la figura 71, l'usuari també pot introduir la quantitat de punts a mà. D'igual manera abans d'enviar la petició al servidor comprovarem que la quantitat de punts sigui vàlida.

Figura 71: *Part de la interfície de la pàgina d'usuari per a generar un nou cupó, on s'observa el "Seekbar" i el input com a eines per a la selecció de punts.*

Per a la part dels administradors, la validació es realitzarà en el panell d'administració en la pestanya "*Vouchers*". Com es veu a la figura 72, aquesta pantalla consta d'un sol input el qual espera un codi d'un cupó. En carregar aquesta pantalla, el servidor envia juntament tots els cupons que actualment són vàlids, és a dir, que no estan caducats o bé ja han sigut utilitzats.

A l'introduir un caràcter en l'input del codi, l'aplicació comprova si el conjunt del codi introduït actualment concorda amb algun dels codis dels cupons que ha obtingut al principi. En cas de no haver-hi coincidències mostra un missatge informant l'usuari administrador. En el moment en el qual troba una coincidència de codis, mostra tota la informació del cupó i de l'usuari que l'ha generat.

Figura 72: *Validació de cupons en el panell d'administració, on el codi introduït ha sigut reconegut pel sistema i es mostra la informació del cupó i de l'usuari que l'ha generat.*

Mostrar la informació del cupó i de l'usuari que l'ha generat, va ser un requeriment especificat pel "*Product Owner*", d'aquesta manera podia assegurar-se que la lògica era correcte (tot i les nostres diverses comprovacions) veient que el nombre de punts del cupó era menor o igual al total de punts de l'usuari. També es mostra la fotografia de l'usuari per tal que l'usuari administrador que el bescanvia s'asseguri amb una mínima validesa que la compta que genera el cupó pertany a la persona que està realitzat la transacció.

És en el moment en quant l'usuari administrador prem el botó de bescanviar el cupó, quan es resten els punts del cupó a l'usuari en qüestió, sempre tornant a fer les comprovacions pertinents per tal de donar robustesa i seguretat a l'aplicació. Si tot ha anat correctament es mostraria un missatge per pantalla que informaria a l'usuari administrador que el bescanvi ha sigut correcte i en qualsevol altre cas s'informaria a l'usuari administrador de quin ha sigut el problema a l'hora de bescanviar el cupó.

4.3 Generador de pizzes

Un factor que dóna molta sociabilitat i molt dinamisme a l'aplicació, és el fet que els usuaris puguin crear les seves pizzes, compartir-les i demanar-les d'igual manera que totes les altres. Aquesta característica parteix sempre del nostre creador de pizzes, que mitjançant un formulari, permet a qualsevol usuari crear una pizza amb els ingredients que més desitgi (i que siguin disponibles en el negoci).

Mitjançant aquest algorisme, donem la possibilitat a tots els usuaris a crear les seves pizzes i a l'hora se'ls hi dóna un preu automàtic relacionat al total d'ingredients que s'afegeixen. Les pizzes es creen mitjançant el perfil de l'usuari, omplint el formulari que es pot veure a la figura 73.

En obrir-se aquest formulari, s'obtenen tots els ingredients actuals que té el negoci registrats al sistema. Un cop l'usuari dóna un nom i una descripció a la seva pizza, s'ajuda a aquest a escollir els ingredients que vol afegir mitjançant la funció d'auto completat amb tots els ingredients que prèviament hem obtingut.

Els preus parteixen d'un preu base per a cada mida de pizza. Els ingredients que es van afegint, tenen un preu associat que s'adjudica quan s'afegeixen al sistema pels usuaris administradors. D'aquesta manera podem diferenciar un ingredient car d'un barat quan un usuari es crea la seva pizza. A mesura que l'usuari va afegint ingredients, els seus preus es van afegint o traient del preu total de cada tamany de la pizza, d'aquesta manera l'usuari pot anar veient a mida que construeix la seva pizza si aquesta és assequible o no (fet també important de cara a mostrar-ho a la resta de la comunitat).

Finalment i com ja s'ha mencionat en la part de disseny, l'usuari indica si aquesta pizza serà pública de tal manera que tota la comunitat pugui veure-la, comentar-la i demanar-la en les seves comandes.

Figura 73: *Formulari per a la creació d'una nova pizza creada per l'usuari.*

5. Resultats

Posteriorment al desenvolupament del codi, s'ha realitzat una etapa d'anàlisis dels resultats on es vol observar si la pàgina web és eficient en termes de rendiment i a l'hora usable i entenedora per a l'usuari. Per tal d'analitzar amb profunditat aquests aspectes, s'han realitzat dos tests que veurem a continuació.

5.1 Test d'usabilitat

Per tal d'avaluar la usabilitat de la interfície de l'aplicació web, s'ha fet un test el qual serà completat per diferents persones de diferents edats i diferents coneixements tecnològics, per tal d'observar si la interfície de l'aplicació és usable.

El test s'ha realitzat a un total de 23 persones, com s'ha dit, intentant variar edats, gèneres i coneixements per tal de treure conclusions molt significatives de cara al disseny de l'aplicació. En aquests testos, primerament s'informava a l'usuari i es demanava el seu consentiment per tal d'utilitzar les seves dades i informació per a l'anàlisi del projecte. Un cop donat el consentiment, l'usuari donava les seves dades i iniciava el test, que consistia de 10 accions diferents a realitzar dins de l'aplicació. Mentre l'usuari anava realitzant les proves, s'anava observant els procediments que realitzava per tal d'arribar a l'objectiu de l'acció en concret i s'anaven extraient observacions.

A continuació es veurà el test realitzat als diferents usuaris i posteriorment l'anàlisi i observacions extretes.

Test d'usabilitat de l'aplicació web: Impronta Pizza

Informació per a l'usuari:

Completant el següent test, acceptes donar les teves dades i respostes com a informació per a analitzar els resultats de l'aplicació web, realitzada com a treball de final de grau per a la Universitat de Barcelona. Les teves dades no seran publicades, però les conclusions extretes gràcies a aquestes si seran publicades juntament amb el treball global al Dipòsit Digital de la Universitat de Barcelona (http://diposit.ub.edu).

0. Dades de l'usuari

- **Nom i cognoms:**
- **Edat:**
- **Nivell tecnològic:**
- **Dispositiu des d'on es realitza el test:**

1. Observa els diferents esdeveniments que hi ha publicats al web i esmenta'ls.

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

2. Busca l'amanida Cèsar de la carta i deixa-hi un comentari.

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

3. Marca 3 productes aleatoris com a preferits, i afegeix-los a la teva comanda.

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

4. Finalitza la teva comanda i fes que l'enviïn al "Carrer Muntaner 15"

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

5. Canvia't la imatge de perfil per una altra qualsevol.

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

6. Genera un cupó de 150 punts.

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

7. Marca com a que no t'agraden els següents ingredients: *"Atún" i "Cebolla"*

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

8. Realitza una petició de reserva per al diumenge 22 de juliol a les 20:30 per a 7 persones

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

9. Quines són les zones de repartiment el restaurant?

- **L'usuari ha assolit l'objectiu?**
- **Temps en assolir l'objectiu:**
- **Observacions:**

10. Opinió personal de la pàgina web i valoració global (funcionalitats + visualització) del 1 al 10.

Anàlisi i observacions del test d'usabilitat

Pregunta 1: Observa els diferents esdeveniments que hi ha publicats al web i esmenta'ls.

Generalment s'ha assolit l'objectiu de la pregunta. Un petit percentatge d'usuaris creuen que seria bo afegir una secció d'esdeveniments a la capçalera per tal d'accedir-hi més ràpid.

Pregunta 2: Busca l'amanida Cèsar a la carta i deixa-hi un comentari.

Generalment s'ha assolit l'objectiu i amb una gran velocitat. La majoria d'usuaris no han utilitzat el buscador de productes sinó que han entrat directament a la pestanya *"ensalades"* de la carta.

Pregunta 3. Marca 3 productes aleatoris com a preferits i afegeix-los a la teva comanda.

S'ha assolit l'objectiu de la pregunta tot i que en alguns casos ha requerit força temps. A alguns usuaris els hi costa relacionar el botó *"m'agrada"* amb què aquell serà producte preferit.

Pregunta 4. Finalitza la teva comanda i fes que l'enviïn al "Carrer Muntaner 15".

S'ha assolit l'objectiu en tots els casos i s'ha observat que en molt pocs usuaris s'ha generat confusió a l'hora de realitzar aquesta acció. La majoria han realitzat el nombre de *"clicks"* mínim fet que indica que és força usable.

Pregunta 5. Canvia't la imatge de perfil per una altra qualsevol.

Generalment s'ha assolit l'objectiu tot i que en alguns casos ha costat trobar l'opció. De totes maneres no es creu necessari haver de realitzar un canvi en la interfície per aquest problema.

Pregunta 6. Genera un cupó de 150 punts.

S'ha assolit l'objectiu en tots els casos. La majoria d'usuaris accedien mitjançant el desplegable de la capçalera, fet que ens indica que hem encertat posant l'accés ràpid als cupons en lloc d'un altra funcionalitat no tan important.

Pregunta 7. Marca com que no t'agraden els següents ingredients: "*Atún*" i "*Cebolla*".

Alguns usuaris s'han arribat a frustrar per no trobar l'opció on realitzar l'acció. Un cop trobada la pantalla els usuaris ho realitzen amb molta facilitat, però el problema és trobar aquesta pantalla que ha generat confusió a un percentatge força elevat d'usuaris. En aquest cas sí que es creu convenient haver de modificar la interfície per tal de facilitar l'accés a aquesta pantalla.

Pregunta 8. Realitza una petició de reserva pel diumenge 22 de juliol a les 20:30 per a 7 persones.

Generalment s'ha assolit l'objectiu i amb força rapidesa. La gran majoria d'usuaris han sabut interpretar el funcionament de la interfície i han realitzat l'objectiu ràpidament.

Pregunta 9. Quines són les zones de repartiment del restaurant?

La gran majoria d'usuaris han realitzat aquesta tasca ràpidament i han trobat la informació amb facilitat. Un petit percentatge d'usuaris (majoritàriament d'edat avançada o inexperts en tecnologies) els hi ha costat una mica més trobar els resultats, però han acabat trobant-los.

Pregunta 10. Opinió personal i valoració.

Generalment les opinions són bones i aconsellen sobre aspectes relacionats amb el disseny. Cap usuari presenta millores en aspecte de funcionament ni de rendiment. Diversos usuaris opinen que es podria reduir l'espai en blanc de la carta i fer que els productes fossin més vistosos, en el sentit de no haver de fer *"scroll-down"* per a poder veure un sol producte.

Alguns altres usuaris opinen el que s'ha observat a la pregunta 7, que l'opció de no m'agrada és molt útil i funcional i hauria d'estar més a la vista i amb un accés més ràpid.

Gran part dels usuaris opinen que la navegabilitat és molt eficient, i el fet de poder editar la teva comanda en qualsevol moment és molt útil.

La puntuació mitja de l'1 al 10 per part dels 23 usuaris que han realitzat el test és de: **8,3**

5.2 *"Stress Test"*

Per tal d'avaluar el rendiment de l'aplicació web amb diferents conjunts de dades, mitjançant el framework "*Laravel*", gràcies al mòdul "*Laravel Seeder"* podem generar conjunts de dades que simulen les dades reals de l'aplicació. Mitjançant aquest recurs omplirem la base de dades i analitzarem els temps en l'obtenció d'aquestes dades i en la renderització de totes aquestes en la vista.

Per a una anàlisi més objectiu, es realitzarà aquest test des de dos dispositius diferents.

Primerament, en un ordinador de torre, que anomenarem **Dispositiu 1**, amb les següents característiques:

- 8 GB RAM, 128 GB SSD, Intel Core i5 i connexió per fibra òptica.

D'altra banda, farem les proves en un portàtil senzill, que anomenarem **Dispositiu 2**, de les següents característiques:

- 4 GB RAM, Intel Core i5 i connexió per "*Wi-fi"*.

*(El tipus de connexió a internet no és rellevant, ja que les proves es realitzen en local).

Sobre les dades, s'han fet 4 proves diferents dels models:

Usuaris, productes, ingredients, relació de productes amb ingredients, comentaris, esdeveniments, cupons, comandes, relació comandes amb productes i reserves.

Finalment també s'ha generat un històric de comandes aleatori per al recomanador.

En cada una d'aquestes proves es genera un nombre determinat de cadascun dels models anteriors: 50, 200, 500 i 1000 respectivament.

A cada una d'aquestes proves, es mesura el temps d'obtenció de les dades i renderització de la vista en les funcionalitats més importants de l'aplicació, que són les següents:

- Pàgina principal
- Carta
- Pàgina de l'usuari
- "*Housekeeping"* mitja dels models
- "*Housekeeping"* apartat cupons
- Crear cupó
- Validar cupó
- Recomanador

En cada una d'aquestes proves, es farà una curta observació dels resultats obtinguts a part de l'anàlisi dels temps mesurats en cada una de les accions en cada un dels dispositius.

5.2.1 *"Stress Test"* en el dispositiu 1:

Prova 1: 50 unitats de cada un dels models.

- Pàgina principal: 0,027s
- Carta: 0,068s
- Pàgina de l'usuari: 0,072s
- "*Housekeeping"* mitja dels models: 0,0019s
- "*Housekeeping"* apartat cupons: 0,0108s
- Crear cupó: 0,0083s
- Validar cupó: 0,0140s
- Recomanador: 0,0238s

Observacions: Temps d'obtenció de dades molt curts i renderització molt ràpida. No es contempla cap retràs en carregar les pàgines.

Prova 2: 200 unitats de cada un dels models.

- Pàgina principal: 0,0196s
- Carta: 0,099s
- Pàgina de l'usuari: 0,102s
- "*Housekeeping"* mitja dels models: 0,0033s
- "*Housekeeping"* apartat cupons: 0,0109s
- Crear cupó: 0,0085s
- Validar cupó: 0,0147s
- Recomanador: 0,0825s

Observacions: Temps d'obtenció de dades lleugerament més grans. Tot i així la renderització molt ràpida. No es contempla cap retràs en carregar les pàgines.

Prova 3: 500 unitats de cada un dels models.

- Pàgina principal: 0,028s
- Carta: 0,143s
- Pàgina de l'usuari: 0,163s
- "*Housekeeping"* mitja dels models: 0,0064s
- "*Housekeeping"* apartat cupons: 0,00203s
- Crear cupó: 0,0076s
- Validar cupó: 0,0164s
- Recomanador: 0,0842s

Observacions: Els temps d'obtenció de les dades són lleugerament més alts però continuen sent ràpids. Es contempla aproximadament 1 segon de congelació en aquelles pantalles on es realitza una renderització de tot un llarg llistat (carta i "*Housekeeping"* dels models).

Prova 4: 1000 unitats de cada un dels models.

- Pàgina principal: 0,0275s
- Carta: 0,285s
- Pàgina de l'usuari: 0,304s
- "*Housekeeping"* mitja dels models: 0,646s
- "*Housekeeping"* apartat cupons: 0,013s
- Crear cupó: 0,0213s
- Validar cupó: 0,019s
- Recomanador: 0,1607s

Observacions: Els temps d'obtenció de les dades són lleugerament més alts però continuen sent ràpids. Es contemplen aproximadament 5 segons de congelació en aquelles pantalles on es realitza una renderització de tot un llarg llistat (carta i "*Housekeeping"* dels models).

5.2.2 *"Stress Test"* en el dispositiu 2:

Prova 1: 50 unitats de cada un dels models.

- Pàgina principal: 0,015s
- Carta: 0,073s
- Pàgina de l'usuari: 0,095s
- "*Housekeeping"* mitja dels models: 0,0026s
- "*Housekeeping"* apartat cupons: 0,0021s
- Crear cupó: 0,0352s
- Validar cupó: 0,0304s
- Recomanador: 0,042s

Observacions: Temps d'obtenció de dades molt curts i renderització molt ràpida. No es contempla cap retràs en carregar les pàgines.

Prova 2: 200 unitats de cada un dels models.

- Pàgina principal: 0,212s
- Carta: 0,9421s
- Pàgina de l'usuari: 0,203s
- "*Housekeeping"* mitja dels models: 0,0140s
- "*Housekeeping"* apartat cupons: 0,0024s
- Crear cupó: 0,0421s
- Validar cupó: 0,0276s
- Recomanador: 0,0992s

Observacions: Temps d'obtenció de dades lleugerament més grans. La renderització comença a ser costosa notant aproximadament 1 segon de congelació en renderització de grans llistats (carta i "*Housekeeping"* dels models).

Prova 3: 500 unitats de cada un dels models.

- Pàgina principal: 0,0275s
- Carta: 0,3246s
- Pàgina de l'usuari: 0,3091s
- "*Housekeeping"* mitja dels models: 0,0145s
- "*Housekeeping"* apartat cupons: 0,00392s
- Crear cupó: 0,0502s
- Validar cupó: 0,0331s
- Recomanador: 0,1325s

Observacions: Els temps d'obtenció de les dades són lleugerament més alts però continuen sent ràpids. Es contemplen aproximadament 3 segons de congelació en aquelles pantalles on es realitza una renderització de tot un llarg llistat (carta i "*Housekeeping"* dels models).

Prova 4: 1000 unitats de cada un dels models.

- Pàgina principal: 0,0282s
- Carta: 0,429s
- Pàgina de l'usuari: 0,5022s
- "*Housekeeping"* mitja dels models: 0,712s
- "*Housekeeping"* apartat cupons: 0,0852s
- Crear cupó: 0,0642s
- Validar cupó: 0,032s
- Recomanador: 0,3702s

Observacions: Els temps d'obtenció de les dades són lleugerament més alts però continuen sent ràpids. La renderització és tan costosa que el dispositiu es manté carregant aquestes dades aproximadament uns 8 segons. Aquesta congelació es produeix en pantalles on es realitza una renderització de tot un llarg llistat (carta i "*Housekeeping"* dels models).

5.2.3 Comparacions i anàlisi dels resultats obtinguts en el "*Stress Test*"

Prova 1: 50 unitats de cada un dels models.

	Dispositiu 1	Dispositiu 2
Pàgina principal	0,027s	0,015s
Carta	0,068s	0,073s
Pàgina de l'usuari	0,072s	0,095s
"*Housekeeping*" mitja dels models	0,0019s	0,0026s
"*Housekeeping*" apartat cupons	0,0108s	0,0026s
Crear cupó	0,0083s	0,0352s
Validar cupó	0,0140s	0,0304s
Recomanador	0,0238s	0,042s

Figura 74: *Taula per comparar la prova 1 (50 unitats per model) en ambdós dispositius.*

Prova 2: 200 unitats de cada un dels models.

	Dispositiu 1	Dispositiu 2
Pàgina principal	0,0196s	0,212s
Carta	0,099s	0,9421s
Pàgina de l'usuari	0,102	0,0143s
"*Housekeeping*" mitja dels models	0,0033s	0,0140s
"*Housekeeping*" apartat cupons	0,0109s	0,0024s
Crear cupó	0,0085s	0,0421s
Validar cupó	0,0147s	0,0276s
Recomanador	0,0825s	0,0992s

Figura 75: *Taula per comparar la prova 2 (200 unitats per model) en ambdós dispositius.*

Prova 3: 500 unitats de cada un dels models.

	Dispositiu 1	Dispositiu 2
Pàgina principal	0,0288s	0,0275s
Carta	0,143s	0,274s
Pàgina de l'usuari	0,163s	0,309
"*Housekeeping*" mitja dels models	0,0064s	0,0145s
"*Housekeeping*" apartat cupons	0,002s	0,0034s
Crear cupó	0,0076s	0,0502s
Validar cupó	0,0164s	0,0331s
Recomanador	0,0842s	0,1325s

Figura 76: *Taula per comparar la prova 3 (500 unitats per model) en ambdós dispositius.*

	Dispositiu 1	Dispositiu 2
Pàgina principal	0,275s	0,0282s
Carta	0,285s	0,429s
Pàgina de l'usuari	0,304s	0,5022s
"*Housekeeping*" mitja dels models	0,646s	0,712s
"*Housekeeping*" apartat cupons	0,013s	0,0852s
Crear cupó	0,0213s	0,0642s
Validar cupó	0,0196s	0,032s
Recomanador	0,1607s	0,3702s

Figura 77: *Taula per comparar la prova 4 (1000 unitats per model) en ambdós dispositius.*

Com es pot observar els temps en obtenir les dades augmenten en el dispositiu 2, i la congelació a causa de la renderització apareix abans amb aquest mateix dispositiu. Tot i així no comença a ser molèstia fins quan tenim aproximadament 500 unitats de cada model, fet que no s'aproxima gaire a la realitat posterior de l'aplicació.

La part costosa és renderitzar tots els productes a la carta, i llistar tots els diferents models que tenim en el panell d'administració. En la realitat, probablement el cas que més s'hi assembli en un pròxim futur sigui en el cas o prova 1, on tindrem aproximadament 50 unitats de cada model (probablement amb un major nombre d'usuaris).

Veient que el que realment ens fa perdre temps en carregar les pàgines és la renderització, cal tenir en compte també, que tots els fitxers de vista són carregats en extensions "*blade.php*", que com s'ha mencionat a l'inici de l'apartat de disseny, són fitxers que es desen automàticament a caché fent així la pròxima càrrega d'aquella pàgina molt més ràpida.

6. Planificació

Seguint els principis de la metodologia de desenvolupament de software "*Agile"*, analitzarem les diferents èpiques d'usuari requerides pel "*Product Owner"* i les subdividirem en tasques, les quals seran agrupades en "*Sprints"* per prioritats.

6.1 Planificació inicial

Per tal de dividir les diferents tasques i poder estimar aproximadament el temps a dedicar al projecte, s'ha fet una planificació inicial prèvia al projecte. Dins d'aquesta planificació inicial analitzarem i argumentarem les decisions preses respecte a l'organització per tal d'assolir el funcionament esperat en l'aplicació amb la màxima eficiència.

Primerament, dividirem les èpiques d'usuari (requeriments del nostre "*Product Owner")* en històries d'usuari o "*user stories"* per tal de facilitar la divisió de les tasques i l'agrupament en "*Sprints"*. Depenent de la importància de les tasques, es realitzaran en "*Sprints"* propers o bé s'implementaran més endavant.

S'ha decidit realitzar "*Sprints"* de tres setmanes per als següents motius:

- Algunes implementacions poden arribar a ser costoses si s'ajunten amb tota la realització del disseny de la interfície de la pantalla en què el conté. S'opta per a que sobri més temps a no pas a que en falti.
- Com a cada final de *"Sprint"* es presenta un producte funcional al "*Product Owner"*, es realitzarà una petita fase de testeig del mòdul implementat. El fet de realitzar "*Sprints"* de tres setmanes en comptes de dues, ens donarà més temps per a aquesta fase de testeig. En cas que en una de les tres setmanes que formen el "*Sprint"* sobri temps, es dedicarà per fer un testeig més exhaustiu o bé recuperar temps que pot ser perdut en altres etapes del projecte.
- Com a comoditat per al "*Product Owner"* i com s'ha acordat amb ell, és preferible fer reunions cada 3 setmanes per tal d'observar l'evolució del projecte.
- El fet de tenir més temps a cada "*Sprint"* ens dóna més flexibilitat a l'hora de realitzar la planificació inicial, ja que és més probable que sobri temps a final de cada "*Sprint"* i aquest sigui aprofitat per a realitzar feina enrederida o bé comprovar el funcionament de l'aplicació fins al moment.

Prèviament a fer la planificació inicial, es van dedicar sis setmanes per a dur a terme l'aprenentatge necessari amb els *"frameworks"* i els llenguatges de programació utilitzats. D'aquesta manera, després d'una primera presa de contacte amb les eines de treball seria més fàcil fer les estimacions de temps que comportaran les diferents tasques.

La planificació inicial s'expressa gràficament mitjançant un diagrama de Gantt en la figura 78:

Nº Sprint	*"User Story"*	Tasca	Estimació inici	Estimació duració	Períodes (setmanes)
	APRENENTATGE	Aprenentatge	1	6	0%
SPRINT 1	LOGIN I REGISTRE	Disseny	7	2	0%
		Lògica	9	1	0%
		Correu de confirmació	9	1	0%
		Recuperació contrasenya	9	1	0%
	NAVEGACIÓ	Disseny pàgina benvinguda	10	2	0%
		Header	12	1	0%
		Footer	12	1	0%
SPRINT 2	PANELL D'ADMINISTRACIÓ	Disseny	13	1	0%
		CRUD Models	13	2	0%
		Validació cupons	15	1	0%
		Dashboard	15	1	0%
SPRINT 3	CARTA	Disseny	16	2	0%
		Obtenció dades	17	1	0%
		Comentaris	17	1	0%
		Sistema "likes"	18	1	0%
		Productes especials	18	1	0%
SPRINT 4	PÀGINA USUARI	Disseny	19	2	0%
		Comandes i reserves	19	1	0%
		Creador pizzes	20	1	0%
		Cupons	20	1	0%
		Preferits	20	1	0%
		No m'agrada	21	1	0%
		Configuracions	21	1	0%
S 5	INTERACCIÓ	WebSockets	22	2	0%
		Recomanador	24	1	0%
	Documentació	Documentació	25	2	0%

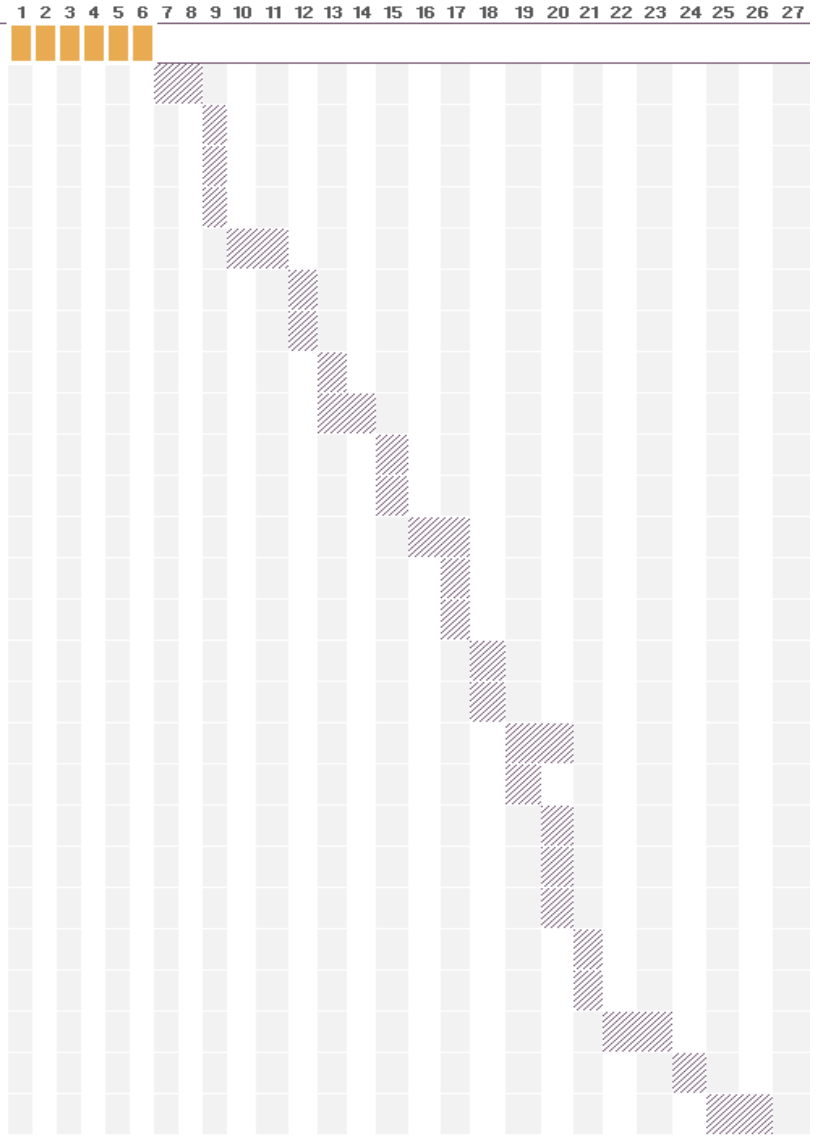


Figura 78: *Diagrama de Gantt per a representar la planificació inicial del projecte*

6.2 Planificació final

A mesura que s'acabaven les tasques, s'anava actualitzant el diagrama de Gantt per tal d'observar si la planificació inicial havia sigut acurada o no. D'aquesta manera, a mida que anem observant si la planificació inicial s'aproximava a les hores reals de dedicació, podem predir amb més precisió si ens faltarà o bé sobrarà temps de cara a la implementació de totes les funcionalitats.

Com es pot observar en la figura 80 que representa la planificació final, s'ha trigat una setmana més del previst segons la planificació inicial a realitzar el projecte. Com s'aprecia al diagrama, a la setmana 20 que era quan s'implementava el creador de pizzes va succeir un imprevist que va fer desajustar la planificació inicial a totes les tasques restants. Aquest imprevist es deu al fet que el "*Product Owner*" va canviar de criteri a l'hora del disseny del creador, i la nova implementació va ser més costosa del previst, fet que va fer necessari una setmana més d'implicació per acabar d'implementar la funcionalitat i testejar-la per a assegurar l'òptim funcionament.

Anteriorment tots els desajustos temporals s'han pogut anar corregint, gràcies al petit marge de temps que ens donava la tercera setmana del "*Sprint*". Tot i així no va ser necessari recuperar una gran quantitat de feina acumulada degut a que la planificació inicial es feia estimant el temps a l'alça, per tant ens trobem que certes tasques es comencen amb una setmana d'antelació.

Finalment, la dedicació de temps en el desenvolupament de tot el projecte es pot representar gràficament en una gràfica que es pot trobar a la figura 79:

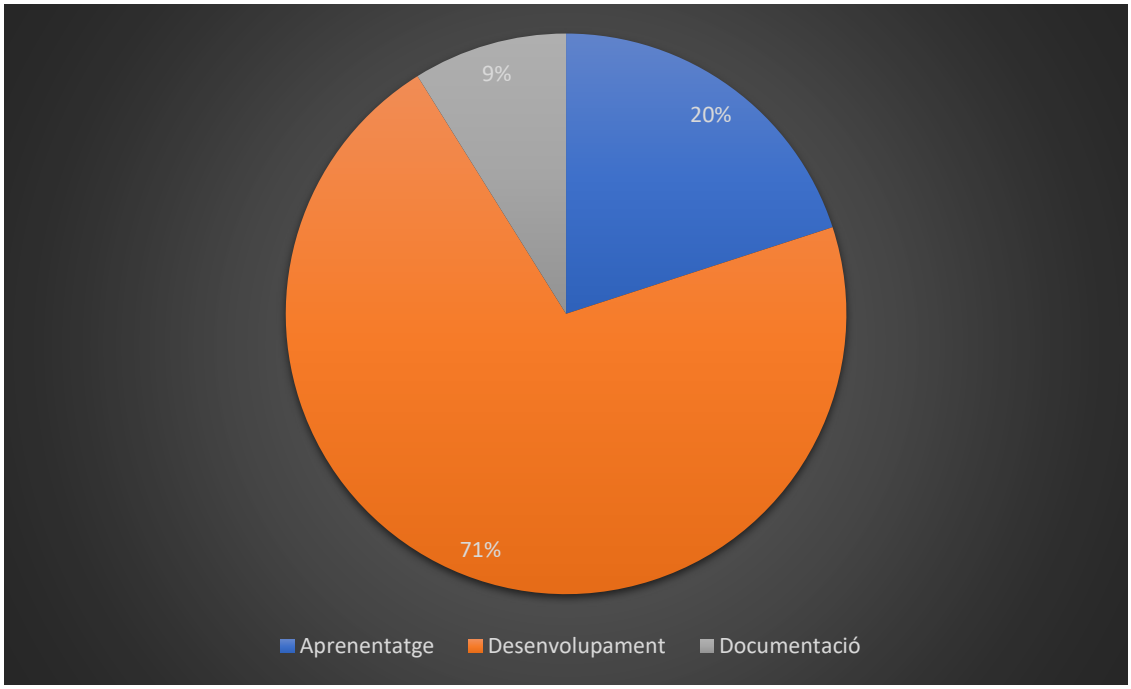


Figura 79: *Gràfica representativa de les hores dedicades al conjunt del projecte*

Per a concloure es mostrarà la planificació final mitjançant el diagrama de Gantt utilitzat per a la planificació inicial, però complet amb els temps reals:

Nº	"*User Story*"	Tasca	Estimació inici	Estimació duració	Inici real	Duració real	Percentatge completat
	APRENENTATGE	Aprenentatge	1	6	1	6	100%
SPRINT 1	LOGIN I REGISTRE	Disseny	7	2	7	1	100%
		Lògica	9	1	8	1	100%
		Correu de confirmació	9	1	9	1	100%
		Recuperació contrasenya	9	1	9	1	100%
	NAVEGACIÓ	Disseny pàgina benvinguda	10	2	10	2	100%
		Header	12	1	12	1	100%
		Footer	12	1	12	1	100%
SPRINT 2	PANELL D'ADMINISTRACIÓ	Disseny	13	1	13	1	100%
		CRUD Models	13	2	13	2	100%
		Validació cupons	15	1	15	1	100%
		Dashboard	15	1	15	1	50%
SPRINT 3	CARTA	Disseny	16	2	16	2	100%
		Obtenció dades	17	2	16	1	100%
		Comentaris	17	1	17	2	100%
		Sistema "likes"	18	1	17	1	100%
		Productes especials	18	1	18	1	100%
SPRINT 4	PÀGINA USUARI	Disseny	19	2	19	1	100%
		Comandes i reserves	19	1	19	1	100%
		Creador pizzes	20	1	20	2	100%
		Cupons	20	1	21	1	100%
		Preferits	20	1	21	1	100%
		No m'agrada	21	1	21	2	100%
		Configuracions	21	1	22	1	100%
S 5	INTERACCIÓ	WebSockets	22	2	23	2	100%
		Recomanador	24	1	25	1	100%
	Documentació	Documentació	25	2	26	2	100%

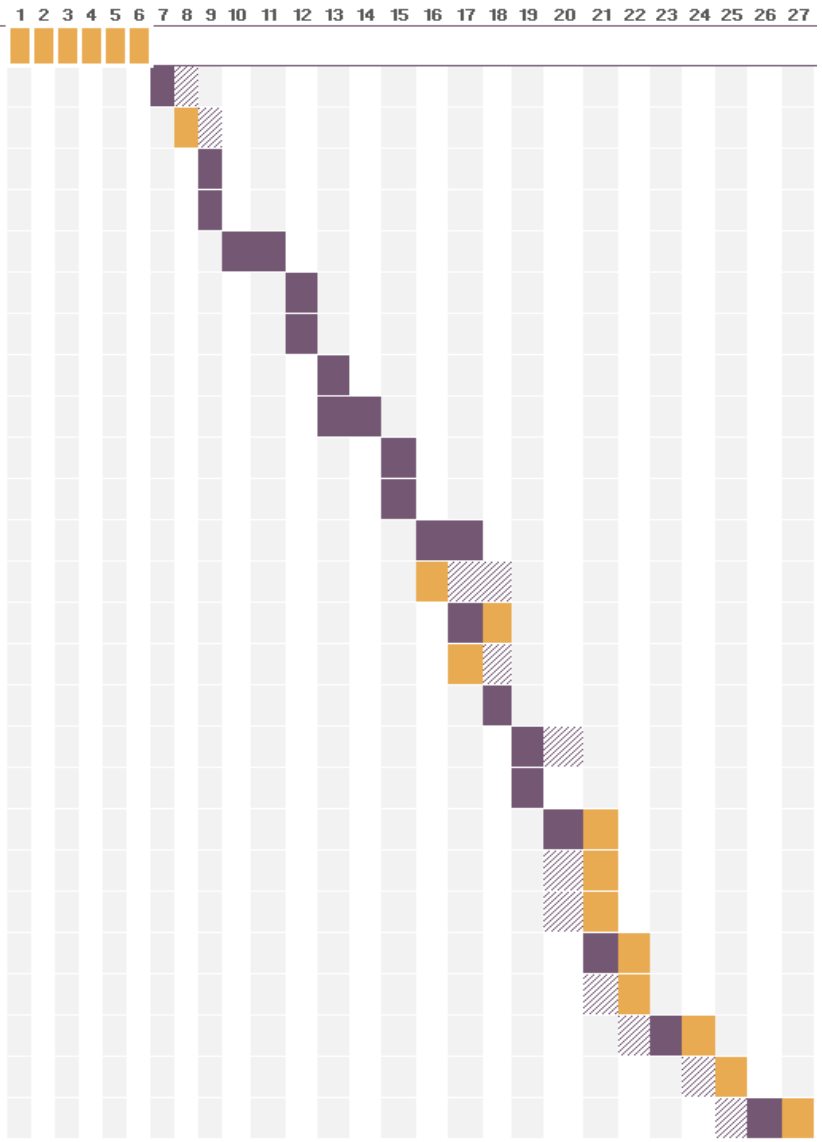


Figura 80: *Diagrama de Gantt per a representar la planificació final del projecte*

6.3 Costos

Per tal de fer una estimació del cost real de l'aplicació, es tindran en compte diversos factors necessaris per al desenvolupament i possible funcionalitat de l'aplicació web.

Primerament, tindrem en compte les hores dedicades en tota la implementació general del projecte. Com es pot observar en la taula de la figura 81, l'aprenentatge es contempla però no incrementa el preu del projecte degut a que va ser un període previ a tot l'inici del projecte.

	Setmanes	Hores/setmana	Hores totals	Preu/Hora	Preu
Aprenentatge	6	15	90	0,00 €	0,00 €
Sprint 1	3	20	60	7,00 €	420,00 €
Sprint 2	3	20	60	7,00 €	420,00 €
Sprint 3	3	20	60	7,00 €	420,00 €
Sprint 4	3	20	60	7,00 €	420,00 €
Sprint 5	4	20	80	7,00 €	560,00 €
Documentació	2	20	40	5,00 €	200,00 €
TOTAL	**24**		**450**		**2.440,00 €**

Figura 81: *Taula representativa de les hores invertides per al projecte.*

D'altra banda, també haurem de tenir en compte que el servei web necessita estar ubicat a un servidor en línia, el qual ha d'estar 24h operatiu per tal de donar servei als clients en qualsevol moment del dia. Analitzant el pes en disc de tot el conjunt del projecte i tenint en compte la base de dades i l'increment de pes en disc que pot suposar la generació de dades en l'aplicació, arribem a la conclusió que amb un servidor bàsic (mitja de 15 € al mes) ja serviria.

També cal tenir en compte el domini que se li donarà a la web, per tal de facilitar l'accés dels usuaris. En cas que es vulgui registrar el web en un sol domini el preu variaria entorn els 10 € a l'any.

Per últim caldria tenir en compte també tot el material necessari que s'ha requerit per tal de desenvolupar l'aplicació. La connexió a Internet, un ordinador decent per a executar els diferents programes de desenvolupament i un espai de treball amb llum per a realitzar les hores de treball serien les tres bases que també s'han de tenir en compte en el pressupost. Aproximadament i englobant aquests factors mencionats anteriorment podríem estimar que s'haurien d'afegir uns 150 € al mes durant el transcurs del desenvolupament del projecte.

7. Treball futur

- **Integració xarxes socials**

En una aplicació on es dóna tanta importància a la interacció entre usuaris, on permets al client crear i compartir amb la resta, seria vital permetre l'extensió de poder compartir totes aquestes dades amb les diferents xarxes socials potencials del moment com "*Facebook"*, "*Instagram*" o "*Twitter"*. A més a més, permetre el registre i inici de sessió mitjançant les dades d'aquestes altres xarxes per a facilitar l'accés i millorar molt més l'experiència d'usuari. El fet que els usuaris fossin capaços de compartir les pizzes que creen o les pizzes d'altres a les altres xarxes, també donaria molta més visualització de l'aplicació fent-la arribar a més usuaris.

- **Dashboard informatiu**

Actualment el Dashboard del panell d'administració es troba amb unes dades falses i gràfiques estàtiques. Seria d'una gran importància que el sistema pogués generar aquestes gràfiques i analitzar aquestes dades per tal de mostrar-les als usuaris administradors. D'aquesta manera tindrien un "*feedback"* molt directe de l'ús de l'aplicació dels clients amb els seus productes, fet que els pot fer corregir factors que poden acabar derivant en una millora de processos de negoci i d'ingressos.

- **Sistema Mailing**

Com s'ha mencionat en l'apartat de disseny, el sistema actual de mailing amb "*Gmail*" té certs inconvenients força perjudicials. La implementació d'un altre sistema mailing com "*MailChimp"*, ens donaria tot l'anàlisi del funcionament dels correus electrònics enviats i ens soluciona també el fet que el sistema ens detecti com una adreça de "*spam*" a causa d'enviar una alta quantitat de correus electrònics en un període curt de temps.

- **Passarel·les de pagament**

El fet d'implementar passarel·les de pagament dins l'aplicació permetria una gran automatització dels processos interns del sistema i facilitaria les transaccions als usuaris i a l'hora als administradors de l'aplicació. Donaria molt més dinamisme a les comandes i fins i tot es podria donar l'opció de permetre als clients (que estan actualment al local) pagar des del dispositiu mòbil sense haver d'anar a caixa.

- **Optimització del recomanador de productes**

En un futur proper, si s'analitza que les dades recaptades dels usuaris són suficients, es podria realitzar una millora del sistema de recomanació. Actualment i com s'ha explicat en l'apartat d'Implementació, el sistema de recomanació només té en compte les comandes passades i els ingredients que no li agraden a l'usuari. Si es denota que els usuaris acostumen a marcar diferents productes com a preferits, es podria afegir aquest paràmetre a tenir en compte a l'hora de calcular la recomanació. Anant més enllà i si les dades que s'han generat ho permeten, també es podria basar la recomanació per semblança d'usuaris. On es podria establir que dos usuaris són semblants segons el nombre de productes preferits i ingredients com a negatius tenen en comú. Aquesta correlació de semblança entre usuaris, els productes que li agraden a aquest, juntament amb els ingredients que no li agraden i l'històric de comandes de la resta d'usuaris són dades que ens permetrien donar unes recomanacions molt més ajustades als usuaris.

8. Conclusions

Com a conclusions extretes un cop acabat el projecte, podríem dir que hem aconseguit assolir els objectius plantejats a l'inici d'aquesta etapa. S'han posat en pràctica tots els coneixements adquirits al grau i a més a més s'ha adquirit gran part de coneixement sobre el funcionament dels "*frameworks"* utilitzats i els llenguatges que requereixen.

S'ha assolit un bon nivell de modularitat en el codi, permetent així futures implementacions i millorant el manteniment de tot el sistema que envolta l'aplicació. Tot aquest codi, controla de forma força eficient com s'ha vist en els tests de resultat, tota la interfície de l'aplicació que també s'ha analitzat i s'ha vist que és força usable, tot i que gràcies a l'anàlisi dels resultats sabem que aquesta interfície pot ser optimitzada en determinats aspectes per a millorar encara més la usabilitat.

Aquesta usabilitat i eficiència assolida prové en gran part gràcies a l'ús dels diferents patrons tant de programació com de disseny d'interfícies, per estructurar i donar en estil a seguir tant com en el codi com en la part visual de la pàgina.

Respecte a l'organització i planificació de les diferents històries i èpiques d'usuari dictades pel nostre "*Product Owner"*, podem dir que ha sigut un encert repartir els diferents "*Sprints"* en tres setmanes en comptes de dues, ja que aquesta setmana de marge ens ha donat el temps necessari per acabar de desenvolupar certes parts d'aquell "*sprint"* en concret, o bé ens ha ajudat a testejar i valorar l'estat del codi en aquell moment. Aquest breu marge de temps també ens ha permès realitzar petites implementacions que no estaven pensades a l'inici de la planificació. Encara que la planificació inicial no fos del tot acurada, s'ha seguit amb molta convicció i s'han pogut anar implementant les funcionalitats i mostrant el producte funcional resultant al "*Product Owner"* que en tot moment ha estat informat de l'estat del projecte.

Posar en pràctica una planificació la qual has de dur a terme el llarg d'aquest projecte, el fet d'informar i mantenir satisfet al "*Product Owner"* amb el producte que estàs realitzant, aprendre i ampliar coneixements amb nous "*frameworks"* per a construcció d'aplicacions web fa veure la complexitat real que té un projecte de desenvolupament de software. A l'altra banda de tota la programació que implica la web, aquest projecte ha sigut un encert no només per posar en pràctica i aprendre els diferents aspectes comentats anteriorment, sinó també per apropar-te al món real i professional del desenvolupament de software i preparar-se pel que el futur pugui oferir.

9. Bibliografia

- Comunitat de programadors Web Laravel, PHP i Vue
  https://rimorsoft.com/

- Integració sistema Gmail Mailing amb Laravel
  https://programacionymas.com/blog/como-enviar-mails-correos-desde-laravel

- Curs iniciació Laravel i Vue combinant tecnologies
  https://www.youtube.com/playlist?list=PLhCiuvlix-rSduJ-vKGpPj5xxcmeLabXc

- Documentació oficial de Laravel: Migracions de la base de dades
  https://laravel.com/docs/5.6/migrations

- Documentació oficial de Laravel: Relacions
  https://laravel.com/docs/5.6/eloquent-relationships

- Documentació oficial de Laravel: Sessions HTTP
  https://laravel.com/docs/5.6/session

- Documentació oficial de Laravel: Emmagatzematge i gestió de fitxers
  https://laravel.com/docs/5.6/filesystem

- Documentació oficial de Laravel: "*Middlewares*"
  https://laravel.com/docs/5.6/middleware

- Documentació oficial de Laravel: Controladors
  https://laravel.com/docs/5.6/controllers

- Documentació oficial de Laravel: Peticions
  https://laravel.com/docs/5.6/requests

- Documentació oficial de Laravel: Respostes
  https://laravel.com/docs/5.6/responses

- Documentació oficial de Laravel: Fitxers "*Blade*"
  https://laravel.com/docs/5.6/blade

- Documentació oficial de Vue JS: Declaració component i propietats + cicle de vida
  https://vuejs.org/v2/guide/instance.html

- Vue JS Rest API amb llibreria Axios
  https://alligator.io/vuejs/rest-api-axios/

- Estudi eMarketer estudiant l'eficiència dels mètodes de comunicació amb clients mencionat a la pàgina 32:
  https://www.emarketer.com/Article/Email-Continues-Deliver-Strong-ROI-Value-Marketers/1014461

- "*Laracasts*": Font d'informació en dubtes concrets sobre Laravel i Vue
  https://laracasts.com/

- "*StackOverflow*": Font d'informació en dubtes concrets sobre Laravel i Vue
  https://stackoverflow.com/

- Comunitat de "*Slack"* on un gran grup de programadors s'ajuden a resoldre tasques concretes sobre Laravel i Vue.
  https://rimorsoft.slack.com/