Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Aplicació web de gestió i planificació de receptes de cuina

Document: Memòria del Treball de Fi de Grau

Alumne: Daniel Garcia Metje

Tutor:Josep Soler

Departament: IMAE

Àrea: LSI

Convocatòria (mes/any): Juny 2020

Aplicació web de gestió i planificació de receptes de cuina

Daniel Garcia Metje

Grau en Enginyeria Informàtica

Itinerari d'Enginyeria del Software

Índex

1. Introducció, motivacions, propòsit i objectius del projecte 7
2. Estudi de viabilitat 9
2.1. Viabilitat econòmica 9
2.2. Viabilitat operacional 9
2.3. Viabilitat tècnica 9
2.4. Viabilitat de dates 9
2.5. Viabilitat legal i contractual 10
2.6. Conclusió 10
3. Metodologia 11
4. Marc de treball i conceptes previs 13
5. Requisits 15
5.1. Requisits funcionals 15
5.2. Requisits no funcionals 16
5.3. Requisits de dades 16
5.4. Dependències entre requisits 17
6. Planificació 19
6.1. Descomposició en paquets de treball 19
6.2. Descripció dels paquets amb les tasques 19
6.3. Activitats 25
6.4. Cronograma 25
7. Estudis i decisions 27
8. Anàlisi i disseny del sistema 31
8.1. Diagrames de casos d'ús 31
8.2. Fitxes de casos d'ús 42
8.3. Diagrames de classes/mòduls 42
8.4. Diagrama de la base de dades 43
8.5. Diagrames de les interfícies (maquetes) 46
9. Implementació i proves 77
9.1. Frontend 77
9.2. Backend 88
9.3. Connexió entre frontend i backend 101
9.4. Testeig de l'aplicació 101
10. Implantació i resultats 105
10.1. Desplegament de la web a un servidor 106
10.2. Resultats 110
11. Conclusions 125
12. Treball futur 127
13. Bibliografia 129
14. Annexos 131
14.1. Annex 1 131
14.2. Annex 2 147

1. Introducció, motivacions, propòsit i objectius del projecte

En un món on cada vegada més utilitzem les xarxes com a font d'informació, per dur a terme una recepta de cuina, molts usuaris prefereixen recórrer a internet enlloc de consultar llibres de cuina. Per això actualment ja hi han moltes plataformes que permeten compartir receptes i poder-les visualitzar.

Alguns dels exemples podrien ser:
https://www.nooddle.es/home
A la Figura 1 es mostra la plataforma de Noodle,la qual és una aplicació de receptes que et permet llegir receptes, compartir amb altres usuaris una recepta i guardar-te receptes en una llista. Com a usuari regular no et permet publicar receptes pròpies ni planificar setmanalment les receptes.

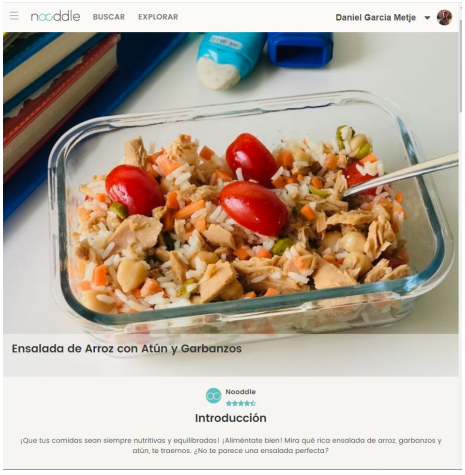


Figura 1 - Recepta de la plataforma Noodle

http://www.mealime.com/
A la Figura 2 hi apareix la secció de planificació de l'aplicació Mealime, la qual et permet crear un pla setmanal sense establir una temporització durant la setmana, a partir d'uns paràmetres et proporciona unes suggerències de plats, i a partir d'aquest pla et genera la llista de la compra, les porcions estan limitades a 2 o 4. Tampoc permet publicar noves receptes com a usuari regular.

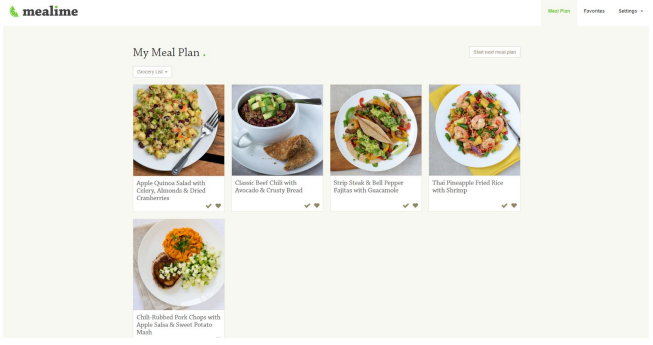


Figura 2 - Secció de planificació de Mealime

https://cookpad.com/

A la Figura 3 es pot observar la pàgina principal de la plataforma Cookpad, on es pot consultar i publicar receptes, però com a molt es poden afegir a una secció de planning però no estan situades al temps ni es pot obtenir una llista de la compra.

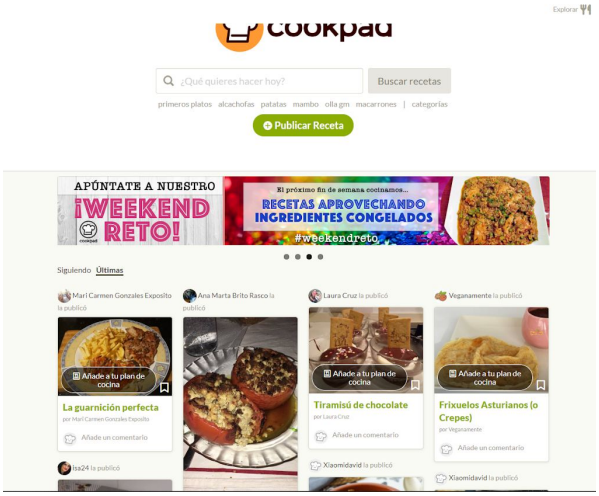


Figura 3 - Pàgina principal de la plataforma Cookpad

Situat el context, com es pot veure, a la xarxa existeixen moltes aplicacions relacionades amb receptes de cuina. La majoria d'aquestes només ofereixen la possibilitat de llegir receptes, i en algun cas també publicar receptes, però els falta una funcionalitat per fer un planning a partir de receptes publicades a l'aplicació. També existeixen aplicacions que permeten fer un planning d'àpats però no permeten la publicació de receptes.

La motivació que m'ha portat a escollir aquest projecte ha estat per una banda la temàtica de la gastronomia com a contingut de la web ja que és un hobbie que tinc i com a usuari d'internet m'agradaria que hi hagués una plataforma que oferís tant la possibilitat de penjar receptes com les de planificar. I per altra banda a nivell tècnic la tecnologia que vull utilitzar per desenvolupar el projecte no la domino gaire, i voldria aprendre-la més.

El projecte, per tant, té per objectiu desenvolupar l'anàlisi, disseny i implementació d'una plataforma web com a solució al problema plantejat. Les funcionalitats més importants d'aquesta plataforma seran:

- Permetre als usuaris compartir receptes(publicar les seves pròpies, consultar les receptes d'altres usuaris, valorar-les).
- Afegir a la plataforma la funcionalitat de planificació de receptes en un període per organitzar l'alimentació de l'usuari.
- Afegir la possibilitat d'obtenir una llista de tots els ingredients necessaris per dur a terme la planificació prevista.
- Al ser una plataforma on qualsevol usuari pot penjar-hi contingut, afegir la capacitat de moderar el contingut dels usuaris per part d'alguns usuaris específics.

2. Estudi de viabilitat

2.1. Viabilitat econòmica

El projecte es realitzarà amb l'objectiu de crear una solució al problema plantejat. El faré tot sol i per tant no hi ha costos associats al desenvolupament del projecte, ja que el considero una tasca acadèmica. Per contra, al ser una plataforma web, en cas d'haver de fer-ho accessible per tota la xarxa , caldria obtenir un servei de hosting i un domini, en concret el servei de hosting pensat per aquest projecte seria proporcionat per https://www.a2hosting.com/, interessa un servei de hosting de l'estil VPS per poder-hi instal·lar Node.js al servidor, ja que no a totes les opcions disponibles de hosting és possible utilitzar Node.js. Dit això per tal d'evitar problemes de limitacions d'espai o fitxers l'opció de hosting escollida ha estat la "Drive Web Hosting" que costa 11.84€ + IVA mensuals. A més cal afegir el cost de compra d'un domini, el qual té un import de 13.62€ + IVA anuals.

En quant a beneficis, el projecte no està pensat a la fase actual a l'obtenció de beneficis, tot i això, està preparat per la inclusió de banners publicitaris o bé una promoció d'ingredients (veure l'apartat de Treball futur per més detalls).

2.2. Viabilitat operacional

El principal objectiu del projecte és poder millorar el servei als usuaris de receptes de cuina d'internet. La solució plantejada respon al problema de que no existeix una plataforma on es puguin compartir receptes de cuina, i també es pugui fer una planificació en un període d'aquestes. Per tant, el projecte és viable pel fet que proporciona una plataforma on es poden compartir i planificar les receptes de cuina.

2.3. Viabilitat tècnica

La tecnologia utilitzada són llenguatges de programació web, amb suport per la majoria de navegadors i sistemes operatius. El coneixement d'aquests llenguatges per part del desenvolupador és mitjà i per tant no hi hauria problema per desenvolupar el sistema d'informació plantejat. I en quant als usuaris, al ser una plataforma web, els principals usuaris són usuaris habituats a navegar per webs i per tant no haurien de presentar problemes a l'hora d'utilitzar el sistema.

2.4. Viabilitat de dates

Les previsions inicials es desajusten lleugerament respecte al temps destinat al projecte, ja que certes tasques han comportat més temps del que s'havia estimat, i per tant, la planificació s'ha vist lleugerament desajustada. Tot i això, s'ha aconseguit una plataforma acabada i operativa abans de la data d'entrega.

2.5. Viabilitat legal i contractual

El projecte recull dades de caràcter personal per tant s'ha d'adequar al Reglament General de Protecció de Dades, incloent tota la informació necessària per el compliment d'aquesta i adoptant les mesures necessàries, ja seria suficient per complir el RGPD. En quant a la LSSICE, de moment no és aplicable ja que la web no obté cap benefici, només seria necessari en cas d'incloure publicitat o obtenir algun benefici mitjançant l'activitat de la plataforma.

2.6. Conclusió

Havent estudiat les diferents viabilitats del projecte, es pot concloure que el projecte és viable.

3. Metodologia

La gestió del projecte s'ha realitzant seguint part de la metodologia PMBOK, la qual és la metodologia estàndard en gestió de projectes i que hem treballat a l'assignatura d'Organització i Gestió de Sistemes d'Informació. Consisteix en una guia de bones pràctiques i definició de conceptes sobre la gestió i administració de projectes, aquesta guia identifica 5 macroprocessos els quals són: inici, planificació, execució, control i tancament. A més estableix quins grups de processos i àrees de coneixement s'haurien d'implementar en cadascuna de les etapes del projecte.

El primer macroprocés consisteix en el conjunt dels processos necessaris per posar en marxa el projecte, per aquest projecte serien els processos realitzats fins l'acceptació del full del TFG.

El segon macroprocés engloba els processos que defineixen l'abast del projecte, els objectius i les accions a aplicar per aconseguir aquests objectius, per aquest projecte hi quedarien englobats els processos de recopilació de requisits, planificació estructurada amb la descomposició en paquets de treball, l'estimació de la duració de les activitats, la gestió del cronograma.

El macroprocés d'execució conté els processos enfocats al compliment dels objectius mitjançant l'execució de les activitats planificades utilitzant els recursos necessaris; en aquest projecte consistiria amb la part de disseny i desenvolupament de la plataforma web.

El macroprocés de control o de seguiment engloba els processos de seguiment i anàlisi del desenvolupament del projecte, com per exemple el seguiment de les tasques planificades, la gestió dels entregables, la gestió d'incidències. En aquest projecte consisteix en l'ajustament de la planificació en funció de les incidències i les revisions de la plataforma per evitar problemes.

El macroprocés de tancament està orientat a la finalització de totes les activitats per tancar formalment el projecte, en aquesta etapa s'hi pot arribar tant havent aconseguit els objectius com no havent aconseguit tots els objectius per determinats motius. El tancament del projecte implica l'alliberament dels recursos utilitzats. En aquest projecte consisteix en l'etapa d'entrega del TFG.

De cares al procés de desenvolupament del software del projecte no s'ha seguit cap metodologia, però s'han seguit les passes tradicionals d'un desenvolupament de software. El desenvolupament del software ha estat condicionat per desconeixement de fins a quin punt s'arribaria a desenvolupar i per tant, el procés de desenvolupament s'acosta a una metodologia iterativa incremental, ja que durant el procés he buscat aconseguir un producte mínim viable el més ràpid possible i anar afegint funcionalitats a mesura que n'anava aconseguint les altres. Per aquest projecte es podrien identificar 4 etapes:

La primera etapa ha consistit en implementar la plataforma per poder gestionar els usuaris i la seva informació personal, també que es poguessin penjar receptes per part dels usuaris i es poguessin consultar les receptes dels usuaris.

La segona etapa estava enfocada a utilitzar de base el producte de la primera etapa i afegir-hi la funcionalitat de planificar receptes.

La tercera etapa s'encarava a afegir la funcionalitat de generar llistes de la compra a partir de les planificacions implementades en l'etapa anterior.

La quarta etapa, la qual degut al termini d'entrega no s'ha iniciat, hagués consistit en la implementació de la secció de moderació i les funcionalitats corresponents pel contingut de la plataforma.

Per fer un control del temps de cada tasca s'ha utilitzat l'eina del Jira, la qual permet definir un conjunt de tasques amb la seva respectiva prioritat, i fer un control del temps i l'estat de cada tasca.

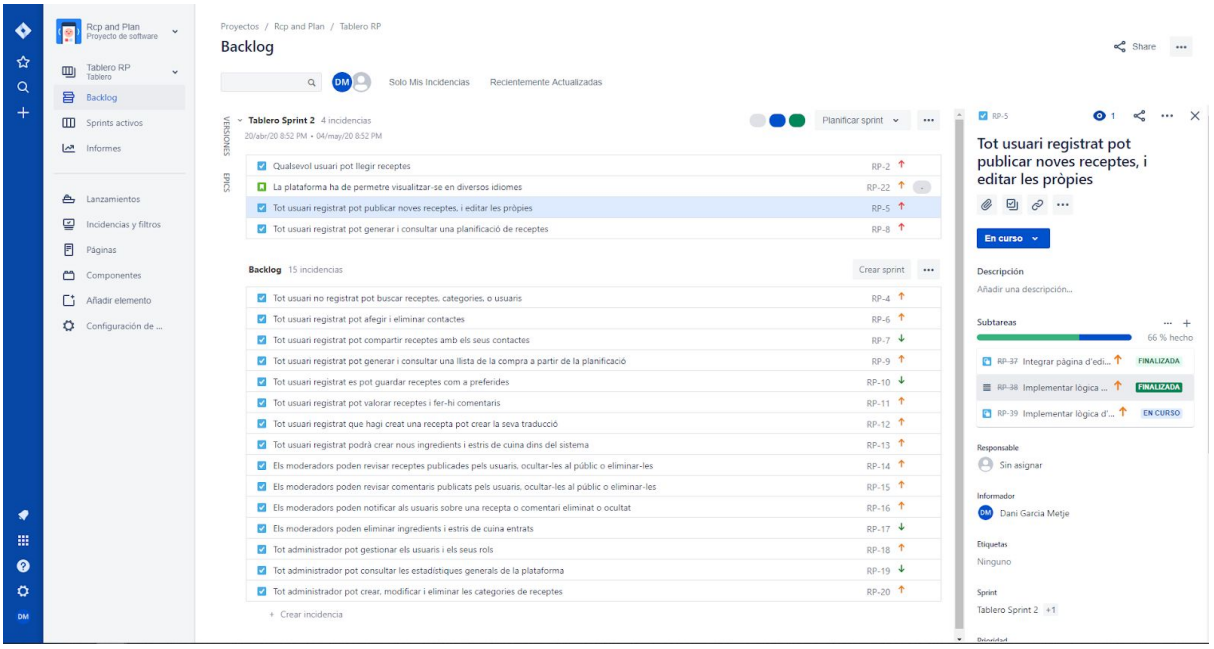


Figura 4 - Taulell de tasques del Jira

4. Marc de treball i conceptes previs

El projecte tracta sobre receptes de cuina, per tant, per situar en el context caldria explicar què és una recepta de cuina, i com s'estructura. A més per parlar de certes funcionalitats s'utilitzen mots que cal concretar a tot el que abarquen degut a que poden portar a confusions.

Per començar, una recepta de cuina és un text instructiu que té per objectiu donar les instruccions necessàries per poder crear un plat seguint de manera ordenada tots els passos. L'estructura d'una recept sol estar formada per una llista d'ingredients amb les seves mesures, una llista d'eines necessàries(no sempre hi apareix), i finalment els passos a seguir per crear el plat de la recepta. Les receptes solen estar englobades en categories, habitualment, segons el tipus de plat(entrant, segon, postres), la dificultat o temps emprat, o segons el sabor del plat (dolç, salat, picant), o el tipus d'ingredient principal(carn, peix, vegetarià).

Aquesta és l'estructura més estesa ja que d'aquesta forma es destaquen els ingredients necessaris i les quantitats i queda més clar i esquemàtic. Tot i això és podrien anar incloent els ingredients i eines necessàries a cada pas.

Dit això, en aquest projecte, al ser una plataforma per compartir receptes caldria dir que al referir-nos a compartir receptes, fem referència a la capacitat dels usuaris per publicar noves receptes, a consultar receptes públiques d'altres usuaris, i a interactuar amb aquestes, per exemple fent-ne una valoració o un comentari.

Ja parlant dels aspectes més tècnics, considerant que havia de desenvolupar una web, la qual segueix un model client(frontend) - servidor(backend). Els llenguatges més usats pel desenvolupament web són: Javascript, PHP, Python, Ruby.

Cadascun d'ells té certs avantatges i certs inconvenients:

- Javascript: S'utilitza sobretot per la part de frontend per la manipulació dinàmica del contingut web, la validació de formularis abans d'enviar-los, manipulació de cookies… Utilitza un únic fil d'execució, és un llenguatge de programació síncron però té mecanismes per implementar asíncronicitat.
- PHP: S'utilitza principalment per la programació de webs i aplicacions web dinàmiques. Està incorporat en l'HTML i això provoca un major temps de càrrega del servidor. Un cop processat l'script el resultat enviat al client és codi HTML.
- Python: Un llenguatge de programació força net a l'hora de programar. Al ser un llenguatge de codi obert té moltes funcions, és multiplataforma i fàcil de programar. Python és un llenguatge interpretat i això el fa més lent que algun altre llenguatge.
- Ruby: Un llenguatge de programació orientat a objectes que permet el tipatge dinàmic i que té un recolector de "basura" automàtic per les variables que no s'utilitzen. És un llenguatge enfocat a la simplicitat, productivitat i de fàcil lectura.

En quant a les tecnologies de bases de dades n'hi han diverses, per començar hi ha dos tipus de bases de dades, les bases de dades relacionals(SQL) i les no relacionals(noSQL), l'ús d'aquestes en el mercat queda distribuït de la següent forma:

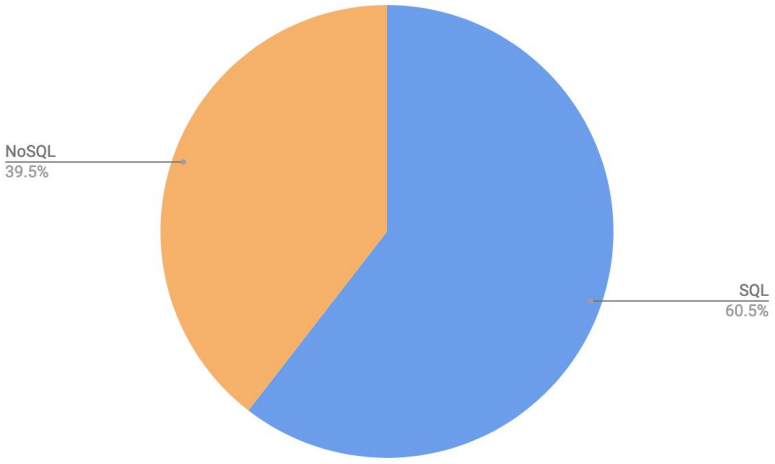


Figura 5 - Gràfic comparatiu tipus de bases de dades

I les tecnologies utilitzades queden distribuïdes:

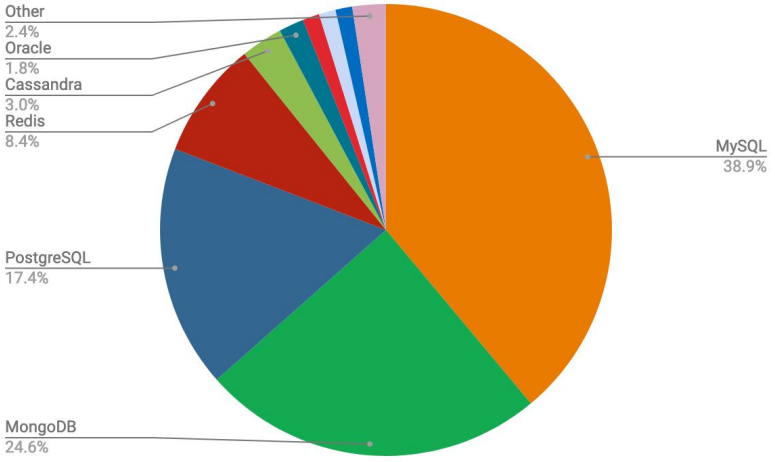


Figura 6 - Gràfic comparatiu dels sistemes de gestió de bases de dades més populars

L'elecció d'una tecnologia respecte una altra sol venir determinada per els avantatges d'uns sistemes quan es donen certes condicions, per exemple que l'aplicació realitzi moltes consultes i poques insercions pot determinar l'utilització d'un sistema com MySQL, mentre que una aplicació que utilitza dades les quals no tenen una estructura ben definida i van canviant pot desembocar a l'elecció d'un sistema com MongoDB.

5. Requisits

Abans de definir els requisits que ha de satisfer l'aplicació descriurem els diferents tipus d'actors que intervindran. L'aplicació tindrà quatre tipus d'actors i els seus rols seran:

- Usuari no registrat: Permisos de lectura de receptes públiques
- Usuari registrat: Permisos de lectura, creació i compartir de receptes, així com creació i edició de planificacions i generació de llistes de compres
- Moderador: Permisos de revisar, amagar o suprimir receptes/comentaris d'altres usuaris
- Administrador: Permisos de modificació de les dades i rols dels usuaris.

5.1. Requisits funcionals

La plataforma ha de permetre crear usuaris al sistema, publicar i consultar receptes del sistema, realitzar planificacions personals de receptes, generar llistes de la compra a partir de les planificacions creades. A més la plataforma ha de permetre la moderació i gestió dels seus continguts, ja que els usuaris poden publicar contingut que no sigui apte per la web i sigui necessari retirar, o bé hi hagi la necessitat de crear una nova categoria per classificar les receptes.

Analitzades totes les funcionalitats que havia de fer l'aplicació es van classificar i enumerar tots els requisits segons l'actor que els desenvoluparia quedant de la següent manera:

Usuari no registrat

RF1. Qualsevol usuari pot llegir receptes *(Alta)*

RF2. Tot usuari no registrat pot registrar-se o identificar-se com a usuari registrat. *(Alta)*

RF3. Tot usuari no registrat pot buscar receptes, categories o usuaris *(Mitjana)*

Usuari registrat

RF4. Tot usuari registrat pot publicar noves receptes, i editar les pròpies. *(Alta)*

RF5. Tot usuari registrat pot afegir i eliminar contactes *(Mitjana)*

RF6. Tot usuari registrat pot compartir receptes amb els seus contactes *(Baixa)*

RF7. Tot usuari registrat pot generar i consultar una planificació de receptes *(Alta)*

RF8. Tot usuari registrat pot generar i consultar una llista de la compra a partir de la planificació *(Mitjana)*

RF9. Tot usuari registrat es pot guardar receptes com a preferides. *(Baixa)*

RF10. Tot usuari registrat pot valorar receptes i fer-hi comentaris. *(Mitjana)*

RF11. Tot usuari registrat que hagi creat una recepta pot crear la seva respectiva traducció. *(Mitjana)*

RF12. Tot usuari registrat podrà crear nous ingredients i estris de cuina dins el sistema. *(Mitjana)*

Moderador

RF13. Els moderadors poden revisar receptes publicades pels usuaris, ocultar-les al públic o eliminar-les. *(Mitjana)*

RF14. Els moderadors poden revisar comentaris publicats pels usuaris, ocultar-les al públic o eliminar-les. *(Mitjana)*

RF15. Els moderadors poden notificar als usuaris sobre una recepta o comentari eliminat o moderat. *(Mitjana)*

RF16. Els moderadors poden eliminar ingredients i estris de cuina entrats. (Baixa)

Administrador

RF17. Tot administrador pot gestionar els usuaris i els seus rols. *(Mitjana)*

RF18. Tot administrador pot consultar les estadístiques generals de la plataforma, des de nombre de visualitzacions de receptes, a nombre d'altes d'usuaris, nombre de valoracions,.... *(Baixa)*

RF19. Tot administrador pot crear, modificar i eliminar les categories de receptes (*Mitjana*)

5.2. Requisits no funcionals

RNF1. Disseny web adaptable a les diferents pantalles, des de dispositius mòbils fins a pantalles d'ordinador de sobretaula.

RNF2. Els noms dels camps a la web serà curts i explícits amb un màxim de 10 mots.

RNF3. El sistema utilitzarà una base de dades per a la gestió de la informació generada a la web, per garantir la persistència i integritat de les dades.

RNF4. La informació personal dels usuaris serà privada i guardada de forma segura a les bases de dades.

RNF5. El sistema ha de permetre que múltiples usuaris utilitzin la plataforma simultàniament.

RNF6. El sistema funcionarà en els principals navegadors moderns: Chrome, Firefox, Safari.

RNF7. La plataforma permetrà visualitzar-se en diversos idiomes: català, castellà, anglès, francès.

RNF8. La plataforma permetrà les notificacions a temps real.

5.3. Requisits de dades

La plataforma ha de permetre als usuaris registrar-se, i guardar la seva informació personal al sistema, a més també ha de permetre guardar les recuperacions de la contrasenya dels usuaris registrats, així com les sessions actives dels navegadors on ja s'han loguejat. A més s'ha de guardar quins usuaris són contactes d'altres usuaris.

A més s'ha de guardar tot el contingut de cada recepta, des del títol de la recepta, la descripció, el temps estimat, l'idioma de la recepta, les porcions de la recepta, si la recepta

és pública o no, el seu autor, les imatges de la recepta, així com també les categories, els ingredients amb les unitats que s'hi utilitzen i les quantitats , les eines necessàries i els passos de la recepta. També s'han de guardar els comentaris, les valoracions, quins usuaris comparteixen les receptes, i a qui. A part el sistema també ha de guardar el temps que els usuaris tarden a dur a terme la recepta per obtenir una mitjana del temps de la recepta.

Finalment, de cares a les planificacions cal guardar per cada usuari quines receptes planifica, en quines dates i per quantes porcions té plantejat fer la recepta. Un cop feta la planificació, el sistema ha de permetre generar una llista de la compra i guardar-ne l'estat d'aquesta, actualitzant-la a mesura que es vagin adquirint els ingredients.

De cares a la funcionalitat de la plataforma, el sistema ha de permetre guardar les sessions dels usuaris mitjançant un token d'autenticació, i els sockets que els usuaris utilitzen per les notificacions a temps real. Cal que els usuaris tinguin guardades les notificacions sabent cada notificació quin usuari l'ha provocat, quina acció ha desencadenat la notificació i si hi ha algun missatge o paràmetre també cal guardar-lo.

5.4. Dependències entre requisits

A continuació es mostra la relació de dependència entre els diferents requisits funcionals, d'aquesta forma obtenim una visió de quins requisits serà necessari de complir primer per tal de poder complir els altres.

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RF10	RF11	RF12	RF13	RF14	RF15	RF16	RF17	RF18	RF19
RF1				X															
RF2																			
RF3				X															X
RF4		X																	X
RF5		X	X																
RF6	X	X	X	X	X														
RF7	X	X	X	X															
RF8		X					X												
RF9	X	X		X															
RF10	X	X																	
RF11	X	X																	
RF12		X		X															
RF13	X	X																	
RF14	X	X								X									
RF15		X											X	X					
RF16												X							
RF17		X																	
RF18	X	X		X		X			X	X									X
RF19		X																	

Figura 7 - La matriu s'interpreta per files, és a dir, el requisit RF_x depèn de (RF_y,...RF_z)

Aleshores, segons la matriu de dependències els requisits que primer cal complir són: RF1, RF2, i RF4. Aquests requisits consisteixen en les funcionalitats més bàsiques dels usuaris i de les receptes. A partir d'aquests, els següents requeriments a complir serien RF7 i RF8, els quals consisteixen en planejar receptes i generar les llistes de la compra i són els requisits que aporten valor al projecte. I finalment la resta consisteixen sobretot en elements secundaris del projecte, com pot ser la moderació de receptes i comentaris.

6. Planificació

A l'hora de planificar les tasques a desenvolupar en aquest projecte, hem descompost el projecte en varis paquets de treballs i cadascun d'ells en diferents tasques. En els apartats següents (o bé en l'apartat 6.1 i en 6.2) es descriuen tant la descomposició com els diferents paquets de treball per aquest projecte. Cada paquet de treball contindrà el conjunt de tasques necessàries per completar el paquet.

6.1. Descomposició en paquets de treball

A la Figura 8 es pot veure la descomposició del projecte en paquets de treball, on els quatre primers blocs consisteixen amb les fases de preparació per poder desenvolupar el projecte. Els blocs de desenvolupament i testeig es duran a terme concurrentment. El darrer bloc consisteix en fer la publicació del projecte i obtenir-ne resultats per part dels usuaris.

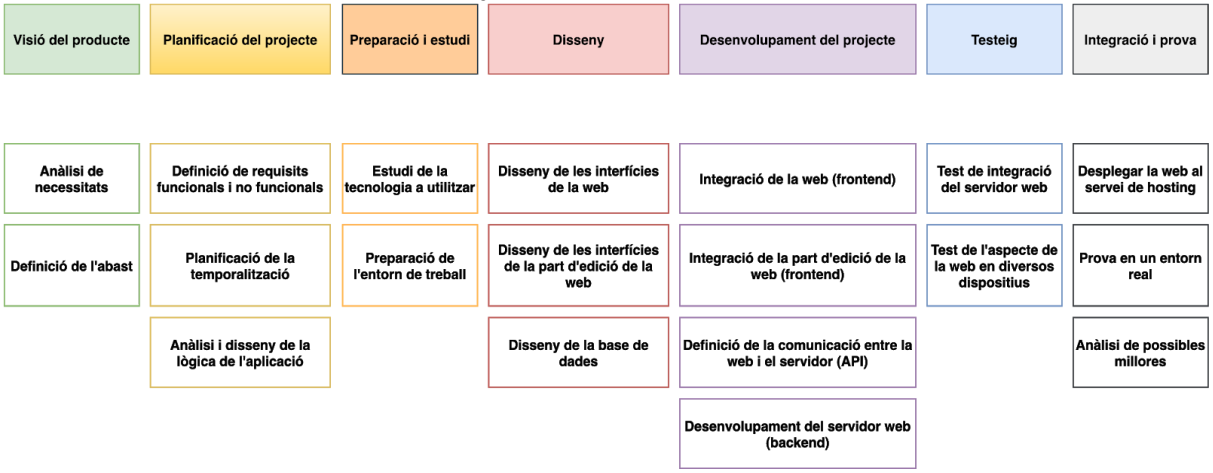


Figura 8 - Esquema de descomposició de paquets

6.2. Descripció dels paquets amb les tasques

A continuació veurem cadascun dels paquets amb les tasques que s'han de desenvolupar. Per cada paquet es fa una descripció del que farà, la seva temporalització, les tasques a desenvolupar i el resultat que se'n espera. Si els paquets d'un bloc eren molt bàsics i no tenien gaires tasques associades, s'han unit i el nom del paquet correspon al nom del bloc.

P1.Visió del producte

Objectiu: Definir i documentar el producte del projecte
Temporalització: 6 hores
Tasques:

- T1: Plantejar el problema a solucionar
- T2: Analitzar les necessitats plantejades al problema
- T3: Fer recerca sobre les solucions existents
- T4: Proposar una solució al problema plantejat
- T5: Definir l'abast de la solució

Resultat: Document d'enfocament del producte

P2.Recollida de requeriments

Objectiu: Analitzar i recollir els requisits funcionals i no funcionals del projecte
Temporalització: 5 hores
Tasques:

- T1: Documentar els requeriments funcionals de la pàgina web
- T2: Documentar els requeriments no funcionals de la web
- T3: Establir prioritats entre els requeriments
- T4: Generar la matriu de dependències entre requeriments

Resultat: Document de requeriments

P3.Planificació de la temporalització

Objectiu: Generar una planificació i estimació del transcurs del projecte
Temporalització: 8 hores
Tasques:

- T1: Generar els paquets de treball a partir dels requisits
- T2: Crear les activitats i construir-ne el diagrama de xarxa
- T3: Construir el diagrama de Gantt a partir de les activitats i el temps previst

Resultat: Documents i diagrames de planificacions

P4.Anàlisi i disseny de la lògica de l'aplicació

Objectiu: Definir i documentar els casos d'ús i informació a guardar
Temporalització: 32 hores
Tasques:

- T1: Dissenyar el diagrama de casos d'ús UML
- T2: Documentar les fitxes dels diferents casos d'ús
- T3: Dissenyar el diagrama de classes/mòduls de l'aplicació
- T4: Documentar els actors i la informació que interactuarà amb el sistema

Resultat: Diagrames UML i documentació d'anàlisis lògic de la plataforma web

P5.Preparació i estudi

Objectiu: Recollir informació sobre les possibilitats de tecnologies que es poden utilitzar, escollir la més adequada i preparar-la per treballar
Temporalització: 3 hores
Tasques:

- T1: Investigar sobre les possibles tecnologies a utilitzar i els seus beneficis
- T2: Escollir les diferents tecnologies necessàries per realitzar cada tasca
- T3: Preparar l'entorn de treball amb les tecnologies escollides

Resultat: Entorn de treball preparat per desenvolupar

P6.Disseny de la base de dades

Objectiu: Documentar i recollir la informació que cal guardar a la base de dades
Temporalització: 12 hores
Tasques:

- T1: Construir el diagrama EER que reflexi tota la informació a guardar
- T2: Determinar per cada entitat i relació els camps que han de contenir
- T3: Transformar el model conceptual a model lògic
- T4: Normalitzar les taules si s'escau.

Resultat: Disseny conceptual i lògic de la base de dades

P7.Disseny de la pàgina web

Objectiu: Dissenyar les interfícies d'usuari de la part pública de la web
Temporalització: 72 hores
Tasques:

- T1: Disseny de la pàgina Home
- T2: Disseny de la pàgina de Planificació
- T3: Disseny de la pàgina de Llista de la Compra
- T4: Disseny de la pàgina Nota Legal, Política de Cookies, Política de Privacitat
- T5: Disseny de la fitxa d'una recepta
- T6: Disseny de la capçalera i el peu de pàgina de la web
- T7: Disseny de la pàgina de Login i de Registre
- T8: Disseny de la pàgina de perfil d'usuari

Resultat: Maqueta de la web

P8.Disseny de la pàgina web (part d'edició i administració)

Objectiu: Dissenyar les interfícies d'usuari de la part de gestió de la web
Temporalització: 36 hores
Tasques:

- T1: Disseny de la pàgina de creació/edició de receptes
- T2: Disseny de la pàgina del Panell General
- T3: Disseny de la pàgina d'administració d'usuaris
- T4: Disseny de la pàgina d'administració de receptes
- T5: Disseny de la pàgina d'administració d'ingredients
- T6: Disseny de la pàgina d'administració d'estris de cuina
- T7: Disseny de la pàgina d'administració de categories
- T8: Disseny de la pàgina d'edició del perfil d'usuari

Resultat: Maqueta de la web

P9.Desenvolupament de la web (frontend)

Objectiu: Desenvolupar la part del client de la web
Temporalització: 150 hores
Tasques:

- T1: Integrar la maqueta de la Home
- T2: Integrar capçalera i peu de pàgina de la web
- T3: Integrar pàgina de Planificació
- T4: Integrar pàgina de Llista de la Compra
- T5: Integrar pàgina de Nota Legal
- T6: Integrar pàgina de Política de Cookies
- T7: Integrar pàgina de Política de Privacitat
- T8: Integrar fitxa d'una Recepta
- T9: Integrar pàgina de Login
- T10: Integrar pàgina de Registre
- T11: Integrar pàgina de Perfil d'usuari
- T12: Integrar pàgina d'edició de receptes
- T13: Integrar pàgina del Panell general
- T14: Integrar pàgina d'administració d'usuaris
- T15: Integrar pàgina d'administració de receptes
- T16: Integrar pàgina d'administració d'ingredients
- T17: Integrar pàgina d'administració d'estris de cuina
- T18: Integrar pàgina d'administració de categories
- T19: Integrar pàgina d'edició del perfil d'usuari

Resultat: Codi font de la web(frontend)

P10.Desenvolupament de la web (backend)

Objectiu: Desenvolupar la part del servidor de la web així com la connexió amb el client
Temporalització: 90 hores
Tasques:

- T1: Desenvolupar el mòdul de connexió amb la base de dades
- T2: Desenvolupar el mòdul/mòduls per gestionar la lògica dels usuaris
- T3: Desenvolupar el mòdul/mòduls per gestionar la lògica de les receptes
- T4: Desenvolupar el mòdul/mòduls per gestionar la lògica de les categories
- T5: Desenvolupar el mòdul/mòduls per gestionar la lògica de les planificacions
- T6: Desenvolupar el mòdul/mòduls per gestionar la lògica dels les llistes de la compra
- T7: Desenvolupar el mòdul/mòduls per gestionar la lògica dels ingredients
- T8: Desenvolupar el mòdul/mòduls per gestionar les unitats de mesura
- T9: Desenvolupar el mòdul/mòduls per gestionar la lògica dels estris de cuina
- T10: Desenvolupar el mòdul/mòduls per gestionar la lògica del Panell general
- T11: Desenvolupar el mòdul/mòduls per gestionar la lògica dels comentaris
- T12: Desenvolupar el mòdul/mòduls per gestionar la lògica de les receptes compartides

Resultat: Codi font de la web(backend)

P11.Testejar la web

Objectiu: Comprovar que la web funciona de la forma esperada
Temporalització: 60 hores
Tasques:

- T1: Crear els testos respectius a les funcions que ho requereixin
- T2: Comprovar el correcte funcionament de la plataforma

Resultat: Informe dels testos i dels possibles errors

P12.Desplegament de la web

Objectiu: Fer accessible la web a la xarxa i obtenir informació sobre el tràfic
Temporalització: 30 hores
Tasques:

- T1: Obtenir hosting on poder penjar la web
- T2: Penjar la web a un servidor
- T3: Comprovar el correcte funcionament de la plataforma dins el servidor
- T4: Comprovar la satisfacció dels usuaris de la web

Resultat: Plataforma accessible per la xarxa

P13.Prova en un entorn real

Objectiu: Comprovar que el funcionament de la web sigui el mateix que la web en local
Temporalització: 18 hores
Tasques:

- T1: Fer testos de les funcionalitats de la web pujada al servidor
- T2: Corregir algun error si apareix al fer el desplegament
- T3: Comprovació del tràfic de la web

Resultat: Validació del funcionament de la plataforma

P14.Anàlisi de possibles millores

Objectiu: Documentar les possibles millores del projecte per un futur.
Temporalització: 5 hores
Tasques:

- T1: Repassar l'assoliment dels requisits del projecte
- T2: Documentar possibles millores de cares al futur

Resultat: Informe de treball futur

6.3. Activitats

El conjunt de tasques que figura a l'apartat de Paquets de treball, coincideixen amb les activitats, en alguna de les activitats, al acabar-la no complirà un requisit ja que són activitats que són generals per tot el projecte i per tant serveixen com a pas previ d'activitats que sí que compleixen requisits al ser acabades. En cas que un requisit aparegui com a completat en diverses activitats, les quals no depenen directament les unes de les altres, significa que cal completar-les totes abans de complir un requisit. Els paquets de treball tenen una estimació del temps a dedicar-hi considerant totes les tasques dins el paquet, per tant en aquest apartat s'especificaran les estimacions de cada activitat per així poder generar el cronograma.

En L'Annex 1 es descriuen detalladament cadascuna de les activitats i les estimacions de temps per cadascuna, així com també els requisits assolits al finalitzar l'activitat.

6.4. Cronograma

El cronograma de la Figura 8, creat a partir de les activitats recollides a l'anterior apartat i basat amb les estimacions d'aquestes ha estat creat amb l'eina GanttProject (https://www.ganttproject.biz/), la qual no permet duracions inferiors a 1 dia, i per tant, es considera a l'hora de col·locar les activitats dins del cronograma, una mitjana de 5 hores de treball. Per tant, en cas que dues activitats tinguin una duració no superior a 5 hores estimades, és possible que es col·loquin al mateix dia.

El cronograma s'ha fet ajustant la planificació a dedicar unes 5 hores diàries i intentant fer que totes les activitats s'acabessin abans del termini d'entrega. Però la planificació del cronograma i la realitat han estat diferents ja que hi ha hagut setmanes que hi he pogut dedicar menys temps de l'esperat, i d'altra banda algunes estimacions han resultat ser molt inferiors al temps dedicat per l'activitat.

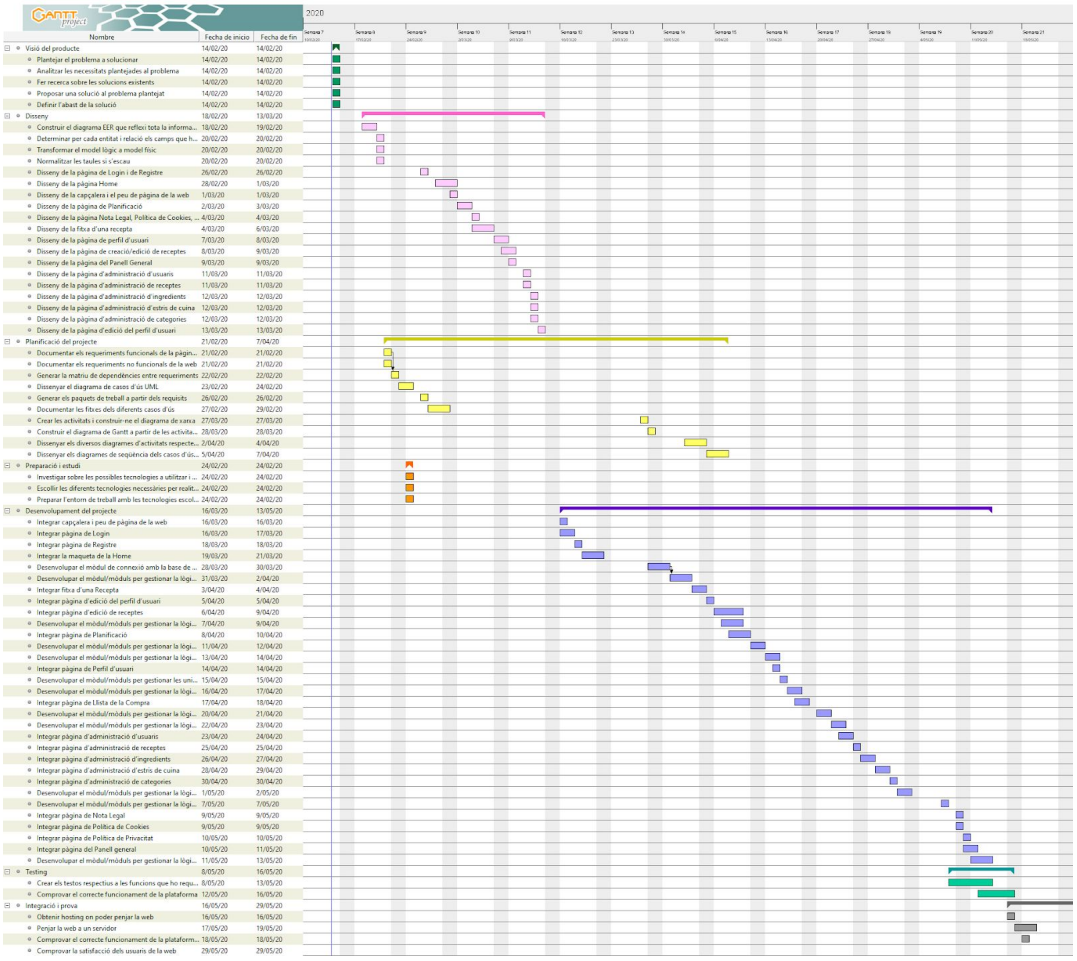


Figura 9 - Diagrama de Gantt.

7. Estudis i decisions

A continuació, s'especifiquen les llibreries, frameworks, llenguatges i programes que he escollit per desenvolupar el projecte i els motius pels quals els he escollit.

El projecte es basa en una aplicació client - servidor, l'objectiu és proporcionar accés a l'aplicació tant des del mòbil com des de l'ordinador, és per això que el projecte consisteix en una plataforma web.

Per l'etapa de disseny de les interfícies d'usuari vaig utilitzar un programa anomenat **Lunacy**, que permet fer el disseny d'aquestes a partir de formes, quadres de text, imatges,... A més, el programa és força senzill i fàcil d'utilitzar, permet generar maquetes del resultat del treball i permet personalitzar força el contingut. A més a partir del disseny, es genera el codi CSS corresponen(colors, mides, fonts,...) que m'ha facilitat a l'hora de posar estils a la web.

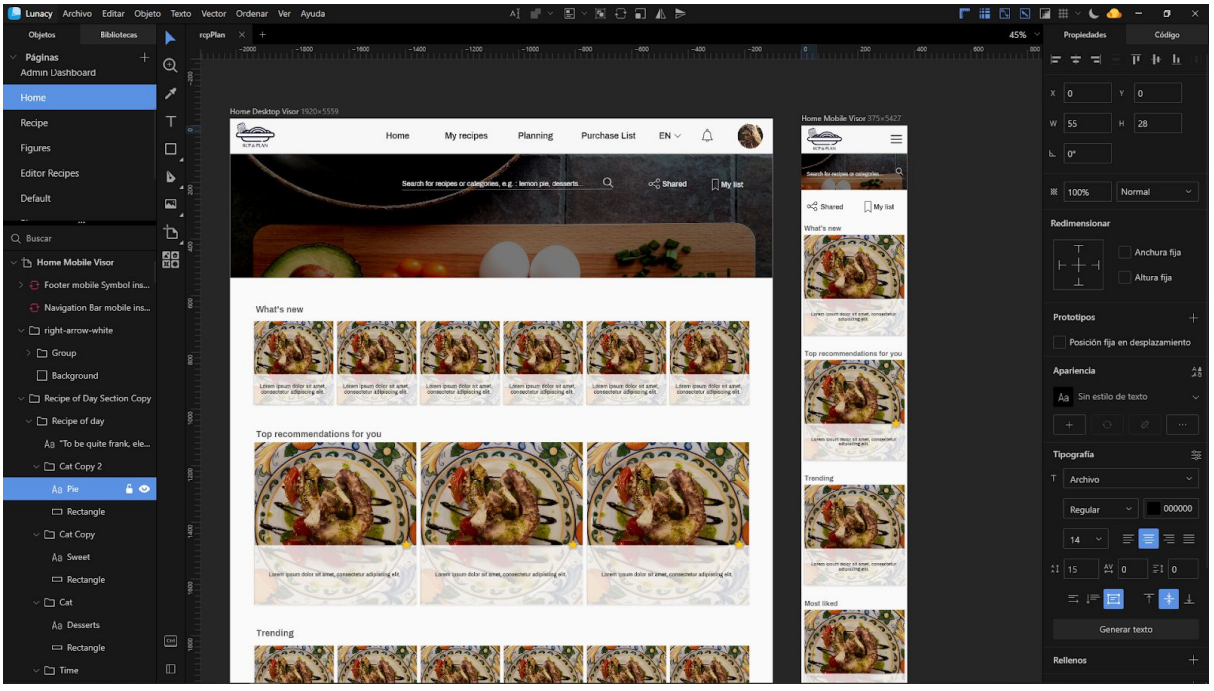


Figura 10- Programa de disseny Lunacy

Considerant que havia de desenvolupar una web, i que aquesta tingui contingut dinàmic, tenia diverses opcions, la meva intenció era utilitzar un sol llenguatge de programació, per tant, podria utilitzar php, el qual és un llenguatge que els seus scripts tant els del client com els del servidor s'executen al servidor, o bé javascript, el qual gràcies a l'entorn de programació de Node.js, els scripts del servidor s'executen al servidor mentre que els scripts del client s'executen al navegador de cada client. Com que PHP s'executa únicament al servidor, al implementar lògica que només afecti al client caldria utilitzar igualment Javascript i de cares a la connexió client-servidor a través d'una API rest, he valorat la

facilitat de Javascript per enviar informació en format JSON, la qual és més fàcil d'interpretar que d'altres formats.

A més al utilitzar una API que intercanvïi informació mitjançant el format JSON, era possible plantejar tenir un únic servidor que pogués servir diversos clients, per exemple un client web, una aplicació nativa d'Android, una aplicació d'escriptori… Atacant al mateix servidor tot i utilitzar tecnologies diferents.

Dit això, he decidit utilitzar el llenguatge javascript per tota l'aplicació, a més he utilitzat un framework per cada part. Del llenguatge Javascript, cal destacar, el tipatge dinàmic el qual permet canviar el tipus de variable en temps d'execució en funció de la dada guardada. També cal destacar que tot i ser un llenguatge síncron, permet l'asíncronicitat de moltes funcions utilitzant les Promises que executen la funció que contenen i poden o bé resoldre retornant informació o bé rebutjar retornant l'error per exemple. Un cop cridada una funció que retorni una Promise, s'encadenen els mètodes then() i catch() per executar funcions un cop resolta o rebutjada la Promise.

Pel client he utilitzat **React.js** el qual és un framework mantingut per Facebook que s'utilitza per desenvolupar interfícies d'usuari interactives, i el seu principal objectiu és la simplicitat, l'escalabilitat i la rapidesa.

Les característiques de React són:

- El DOM virtual: El navegador un cop es carrega una pàgina genera el DOM corresponen que consisteix en un arbre de tots els elements que permet al Javascript accedir a tots els elements de la pàgina,a més d'afegir i eliminar elements de la pàgina dinàmicament. A la Figura 10 es mostra un exemple de la representació d'un DOM:

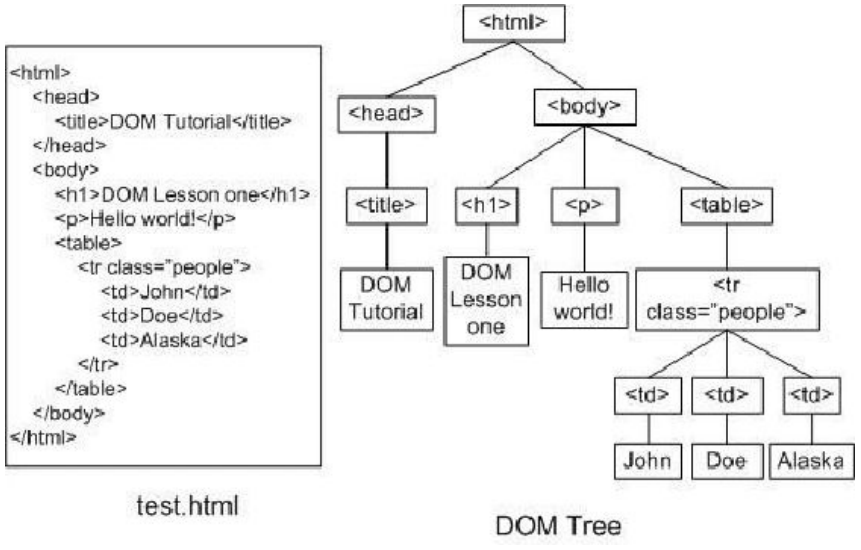


Figura 11- Corresponència entre l'HTML i el DOM
https://www.researchgate.net/figure/Dom-Tree-of-An-Example-Web-Page_fig2_221417012

Situats amb el concepte del DOM, al afegir i eliminar elements, es regenera tot el DOM, fet que suposa una càrrega de treball innecessària ja que només seria necessari actualitzar la part que afecta al node que s'ha modificat. Per tal d'evitar

això i per optimitzar la velocitat de la pàgina React.js crea un DOM virtual, que captura tots els canvis fets als elements de la pàgina i després el framework s'encarrega de fer els canvis pertinents sense haver de regenerar tot l'arbre, evitant el cost que representa recarregar-lo sencer.

- JSX. És una sintaxi que permet escriure HTML dins React.js, i en facilita l'escriptura respecte els mètodes tradicionals en Javascript de *createElement* i *appendChild*. A més permet escriure expressions dins de { } les quals poden contenir variables, i en el cas del projecte seran utilitzades al projecte per mostrar el contingut dinàmic dins la web. JSX segueix les normes XML per tant, els elements han d'estar tancats correctament a diferència d'HTML pur.

Pel servidor he utilitzat **Express.js** és un framework dissenyat pel disseny d'aplicacions web i API's, el qual facilita molt la definició de les rutes d'una API i el codi que s'ha d'executar per cadascuna.

Com que el contingut de la web és dinàmic, he hagut d'utilitzar una base de dades, en aquest projecte, degut a que les entitats estan ben definides i no tenen columnes molt diferents entre elles, he optat per una base de dades relacional, en concret **MySQL**.

Un punt que ha influït en la decisió del Javascript, ha estat el fet que algun dels requisits requereix alguna funció en temps real, com per exemple les notificacions. Utilitzant per les dues parts aquest llenguatge m'ha permès utilitzar la llibreria Socket.io, la qual implementa WebSockets que serviran per proporcionar les actualitzacions en temps real.

Per qui no estigui familiaritzat amb el concepte de WebSocket, cal definir primer que els servidors web tenen una connexió amb el client en la qual només el client pot fer peticions, i el servidor respon a aquestes peticions. Mentre que el servidor és incapaç de enviar informació al client sense una petició per part del client. Amb aquest problema, apareixen els WebSockets que s'encarreguen de solucionar aquest problema i implementen una solució en la qual el servidor pot enviar informació al client sense haver-hi cap petició per part del client.

Mòduls/llibreries utilitzats/des:
La majoria de mòduls utilitzats són o bé per utilitzar components per evitar haver d'implementar tota la lògica dels components interactius,o bé per utilitzar funcionalitats ja implementades. Per tant, la utilització de les llibreries es basa en si utilitzar-les aporta més facilitat o comoditat a l'hora de desenvolupar l'aplicació. Les llibreries utilitzades han estat:

- Nodejs Mysql: Mòdul de NodeJS per utilitzar bases de dades MySQL, utilitzat per la manipulació de les dades de la web.
- NodeMailer: Mòdul per enviar emails. S'utilitza per a l'autenticació d'usuaris.

- Moment.js: Mòdul per manipulació de dates i temps.
- Bcrypt: Mòdul d'encriptació per l'encriptació de contrasenyes.
- uuid: Mòdul de creació d'identificadors únics aleatoris.
- react-router-dom: Mòdul per el redireccionament i la navegació en una Single Page Application(SPA).
- react-burger-menu: Mòdul per crear un menu estil "burger" en React.
- react-carousel: Mòdul que implementa un component per generar un slide d'elements. Utilitzat per mostrar les receptes.
- react-select: Mòdul que conté una implementació de selectors d'opcions amb opcions de creació de nous elements.
- react-confirm-alert: Mòdul que conté un component per mostrar alertes de confirmació a l'hora de realitzar certes accions a la web.
- react-accessible-accordion: Mòdul d'un component de caixetins plegables utilitzat per els passos d'una recepta.
- react-dropzone: Mòdul que permet pujar fitxers utilitzant el moviment de arrossegar i deixar anar el fitxer o bé cercar el fitxer a penjar.
- react-fast-compare: Mòdul que facilita una funció per comparar dos variables en React.
- react-html-parser: Mòdul que permet parsejar com a codi HTML una cadena de text guardada en una variable.
- zxcvbn: Mòdul per comprovar si una contrasenya compleix certs requisits de seguretat.
- socket.io: Mòdul que implementa els WebSockets esmentats anteriorment amb l'objectiu d'obtenir notificacions en temps real.
- react-responsive: Mòdul que permet renderitzar certs elements segons la mida de la pantalla. S'utilitza en cas de que no es puguin aplicar les "media queries" les quals contenen estils que s'apliquen només en cas de complir les condicions de la consulta.
- croppie: Mòdul per retallar una imatge en forma circular. S'aplica per pujar la foto de perfil dels usuaris.
- TinyMCE: Mòdul que implementa un editor de text amb possibilitat de crear interactivament llistes, negretes, subratllats…

8. Anàlisi i disseny del sistema

Per a l'anàlisi del sistema he utilitzat diagrames i eines del llenguatge de modelització unificat(UML). Tot seguit veurem el diagrama de casos d'ús general i a continuació els diagrames de casos d'ús. En aquells casos més complexes s'ha fet un refinament amb el corresponent diagrama.

8.1. Diagrames de casos d'ús

Per començar, per tenir una visió general dels casos d'ús del sistema del projecte he modelat aquests casos d'ús a alt nivell en un diagrama de casos d'ús que podeu veure a la Figura 12. Per situar en context, hi han 4 actors que interactuen amb el sistema, cada cas d'ús està representat per les elipses dins el requadre, i les línies que connecten els casos d'ús i els actors indiquen que hi ha una interacció.

Figura 12-Diagrama general de casos d'ús

Com es pot apreciar en el diagrama anterior, els casos d'ús són molt a alt nivell i no hi han representats certs elements que es mostraran en els diagrames de casos d'ús següents. L'ordre dels casos d'ús detallats anirà dels casos d'ús del rol més restringit al menys restringit. A continuació podem veure els diferents casos d'ús i en aquells més complexos s'ha representat amb la figura corresponent.

- Consultar informació legal i sobre privacitat
  Aquest cas d'ús simplement consisteix en mostrar a l'usuari la informació que té la web sobre temes legals, de cookies o de privacitat.
- Llegir receptes
  Aquest cas d'ús ha de permetre a l'usuari veure les receptes de la web i si n'escull una ha de permetre veure'n tota la seva informació.
- Entrar com a usuari registrat
  La plataforma ha de facilitar als usuaris un mecanisme per poder-se identificar a la plataforma.
- Recuperar contrasenya usuari registrat
  Aquest cas d'ús implica cert refinament. Concretament aquest cas d'ús implica la identificació de l'usuari a través del seu correu electrònic. En cas de que existeixi un usuari amb el correu introduït s'envia un correu per recuperar la contrasenya. Al rebre el correu, aquest permetrà accedir a un formulari on canviar la contrasenya de l'usuari per una nova. El refinament del cas d'ús es pot veure representat a la Figura 13.

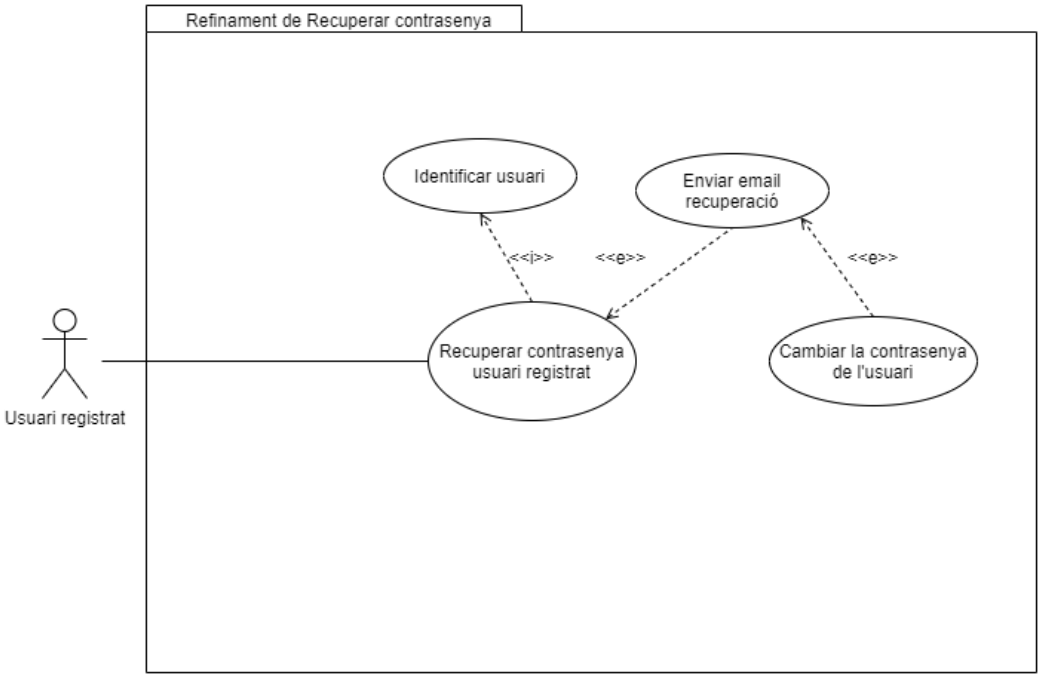


Figura 13-Diagrama del refinament del cas d'ús de Recuperar contrasenya

- Registrar-se
  La plataforma ha de permetre que nous usuaris es registrin al sistema.
- Consultar informació usuaris
  Aquest cas d'ús consisteix en mostrar als usuaris de la web la informació pública de l'usuari.
- Mantenir receptes pròpies
  Aquest cas d'ús requereix una visió més profunda, ja que implica diversos casos. El manteniment de receptes consta de 3 possibles casos d'ús, l'alta d'una recepta, la

modificació d'una recepta o eliminació d'una recepta. Aquestes impliquen la identificació de l'autor de la recepta. A més qualsevol d'aquests casos implica gestionar les notificacions dels contactes de l'usuari. La representació esquemàtica d'aquest refinament es mostra a la Figura 14.

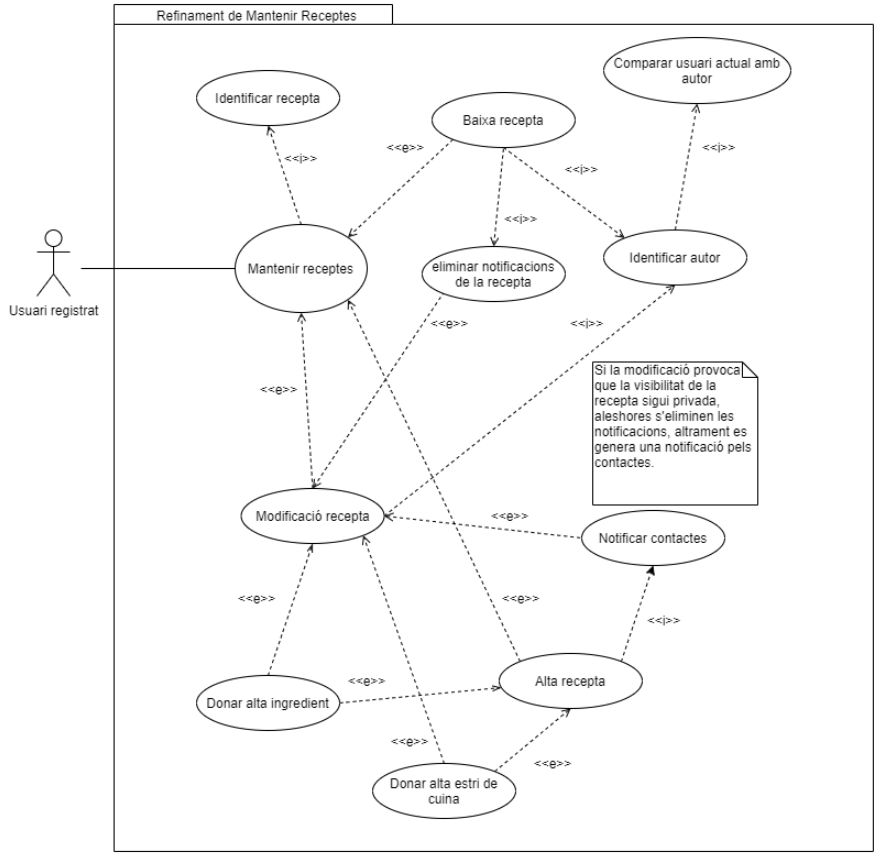


Figura 14-Diagrama del refinament del cas d'ús de Mantenir Receptes

- Guardar receptes

Aquest procés ha de permetre als usuaris guardar-se en una llista les receptes que els hi han agradat o semblat interessants per poder-hi accedir en un altre moment.

- Mantenir informació usuari

El refinament d'aquest cas d'ús queda representat a la Figura 15.

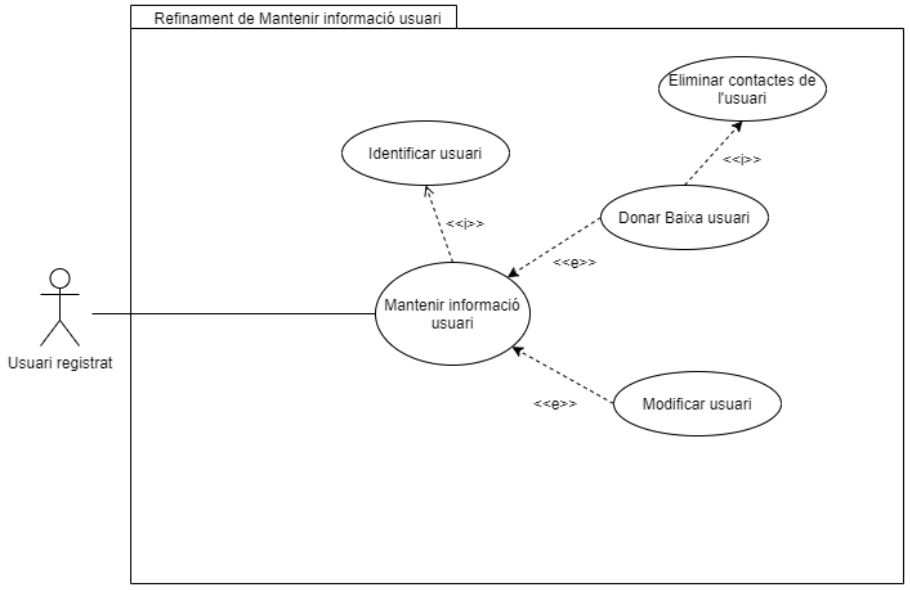


Figura 15-Diagrama del refinament del cas d'ús de Mantenir informació usuari

- Mantenir planificacions receptes
  El manteniment de les planificacions de les receptes permet dos casos d'ús, la creació de noves planificacions i l'eliminació de planificacions existents. Tal com es pot observar a la Figura 16 no es permet el cas d'ús de modificació.

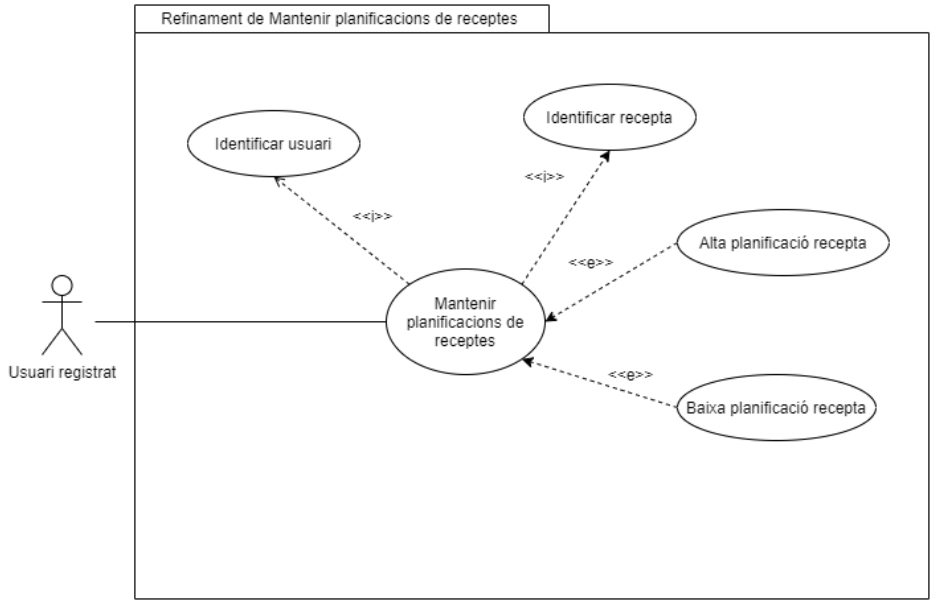


Figura 16-Diagrama del refinament del cas d'ús de Mantenir planificacions de receptes

- Registrar temps de preparació d'una recepta
  El refinament d'aquest cas d'ús queda representat a la Figura 17.

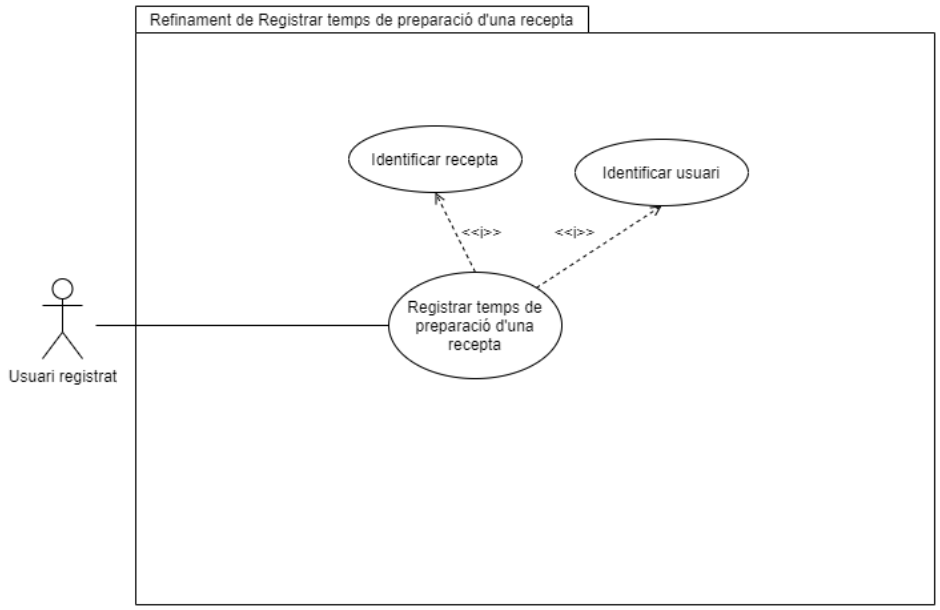


Figura 17-Diagrama del refinament del cas d'ús de Registrar temps de preparació d'una recepta

- Mantenir llistes de compra

  Aquest refinament contempla els casos de creació de llistes de la compra, que provoca una generació de la llista d'ingredients a comprar i el bloqueig de les planificacions de l'interval seleccionat. També contempla l'eliminació de llistes de la compra no completades, així com el cas de marcar com a completada la llista per poder generar més llistes en aquelles dates. Finalment també s'inclou el cas d'ús que consisteix en marcar o desmarcar una entrada de la llista de la compra com a completada. Tota aquesta explicació queda reflexada a la següent figura:

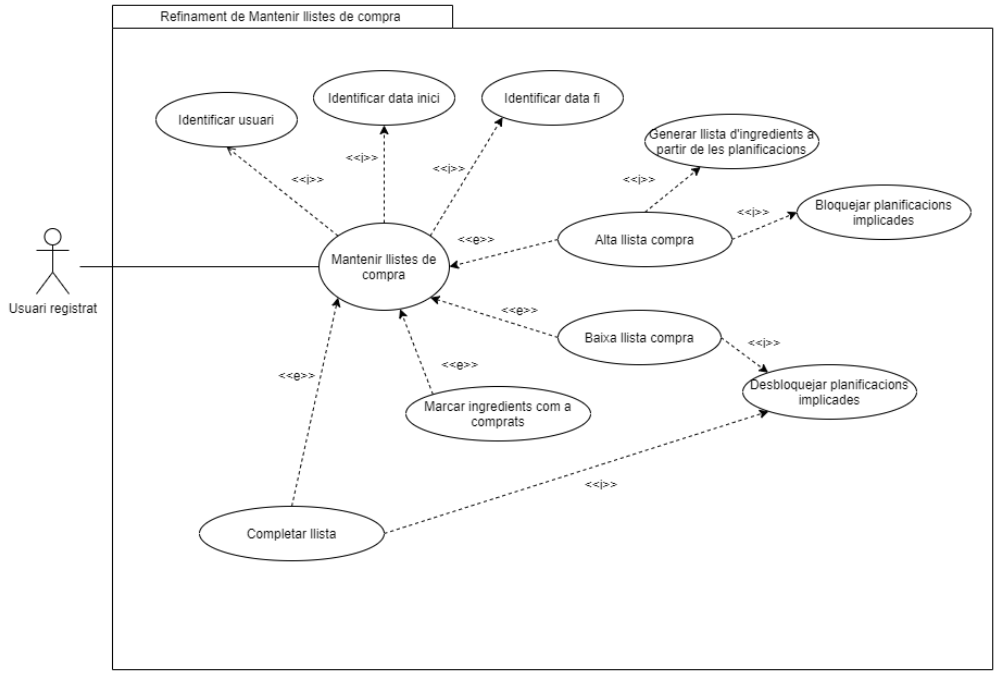


Figura 18-Diagrama del refinament del cas d'ús de Mantenir llistes de compra

- Compartir receptes
  El refinament d'aquest cas d'ús es mostra a la Figura 19.

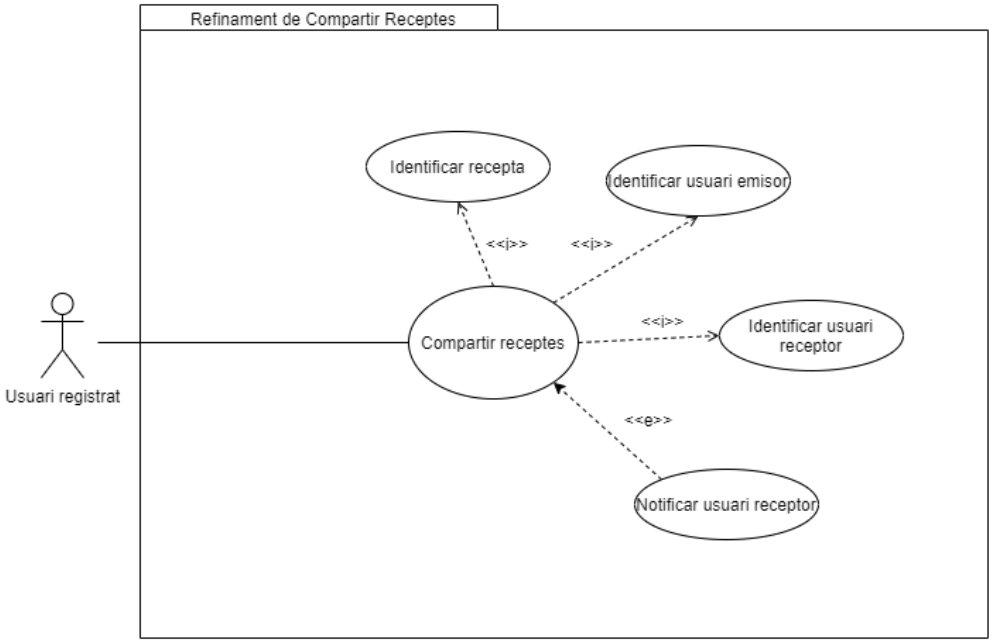


Figura 19-Diagrama del refinament del cas d'ús de Mantenir llistes de compra

- Valorar receptes
  Com en alguns dels altres casos, el cas en sí és senzill però es complica una mica al implementar la gestió de notificacions. Aquest aspecte es detalla en un comentari a la Figura 20.

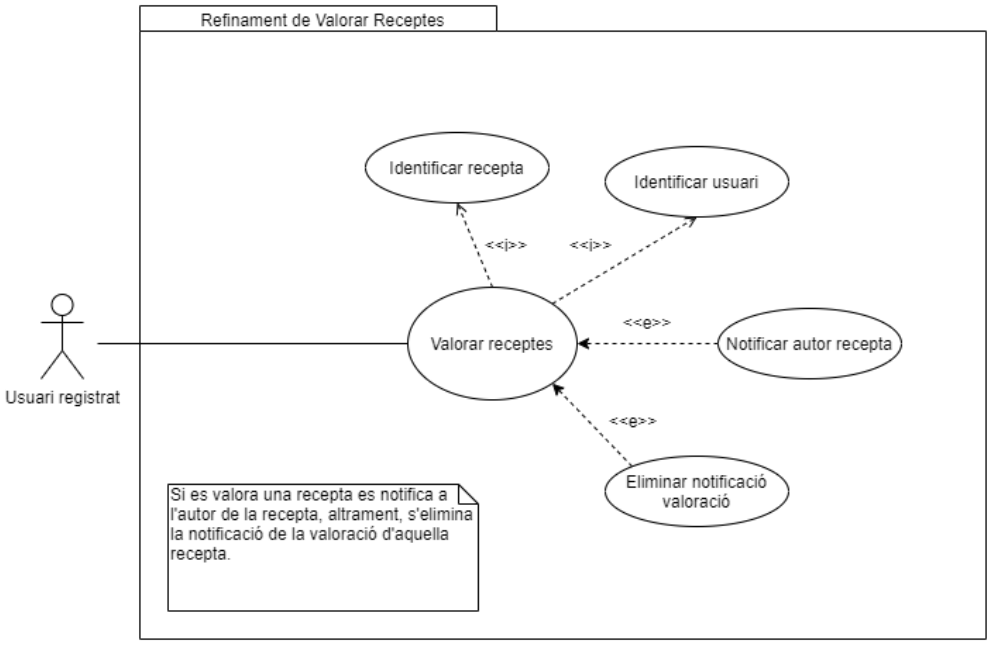


Figura 20-Diagrama del refinament del cas d'ús de Valorar Receptes

- Comentar receptes
  El refinament d'aquest cas d'ús és molt semblant a l'anterior tal com es pot apreciar a la Figura 21 respecte la Figura 20.

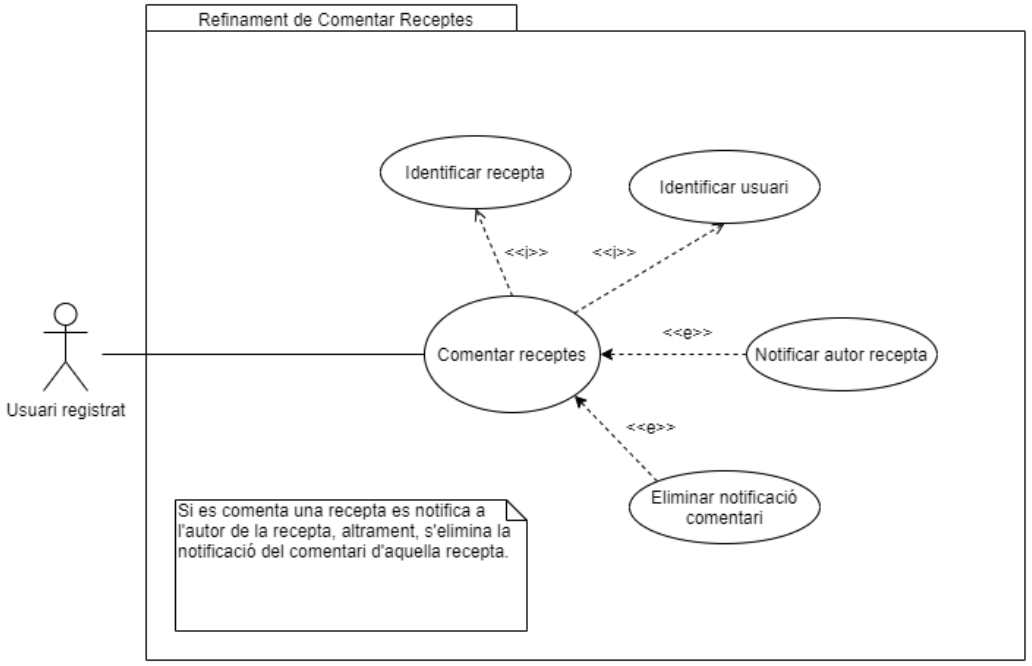


Figura 21-Diagrama del refinament del cas d'ús de Comentar Receptes

- Mantenir contactes
  Aquest cas d'ús queda representat amb una visió més profunda a la Figura 22.

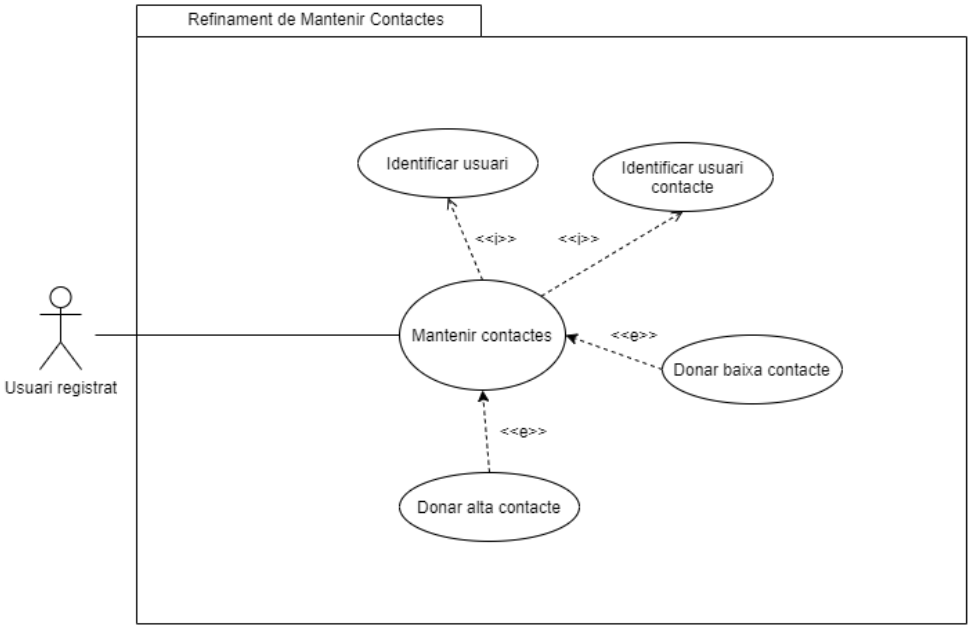


Figura 22-Diagrama del refinament del cas d'ús de Mantenir Contactes

- Denunciar receptes

  Aquest cas d'ús contempla el fet que un usuari registrat pugui denunciar una recepta fet que provocaria una notificació als moderadors per tal de que revisessin manualment la recepta. Mostrat a la Figura 23:

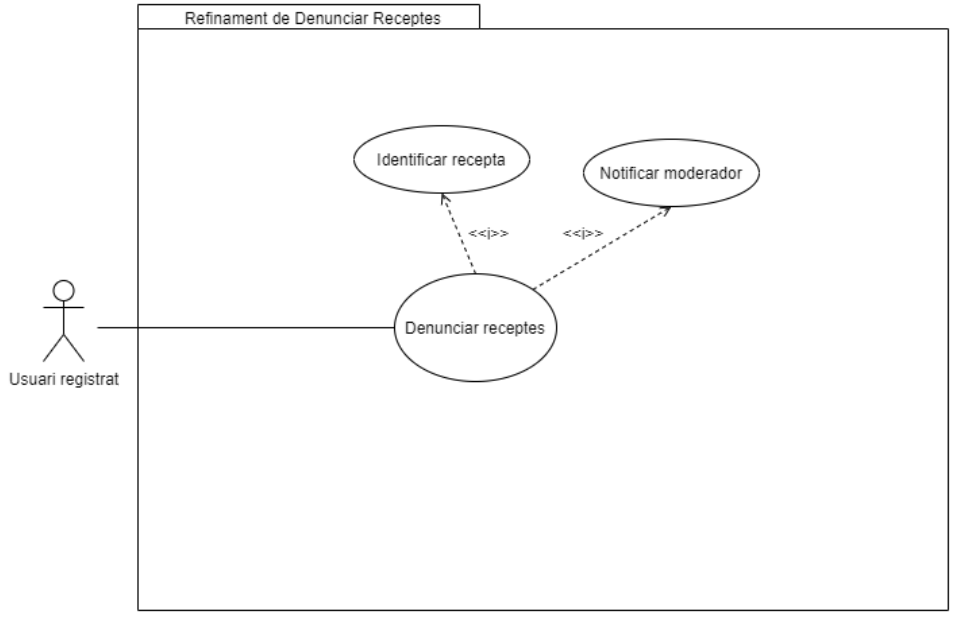


Figura 23-Diagrama del refinament del cas d'ús de Denunciar Receptes

- Moderar comentaris

  Un moderador té la capacitat per revisar un comentari d'una recepta i si creu que no és adequada per la plataforma, aleshores pot amagar el comentari o bé si creu que no es pot acceptar el comentari pot fins i tot eliminar el comentari en qüestió, notificant-ne a l'autor de l'acció realitzada.

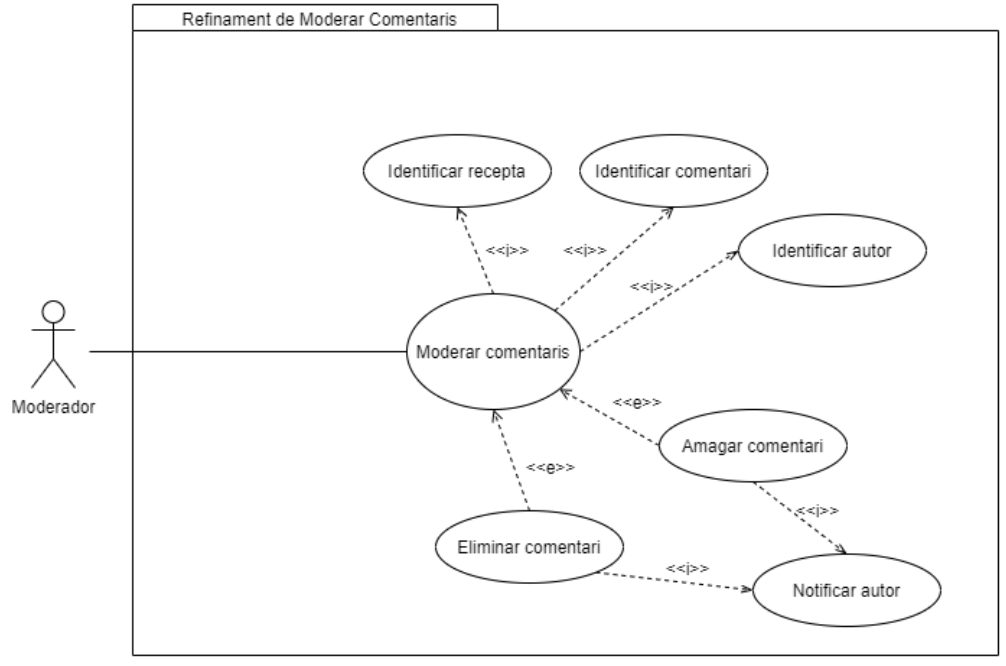


Figura 24-Diagrama del refinament del cas d'ús de Moderar Comentaris

- Moderar receptes

  Com en el cas anterior, un moderador també pot fer les mateixes accions sobre les receptes i tal com es mostra a la Figura 25, el diagrama UML és calcat.

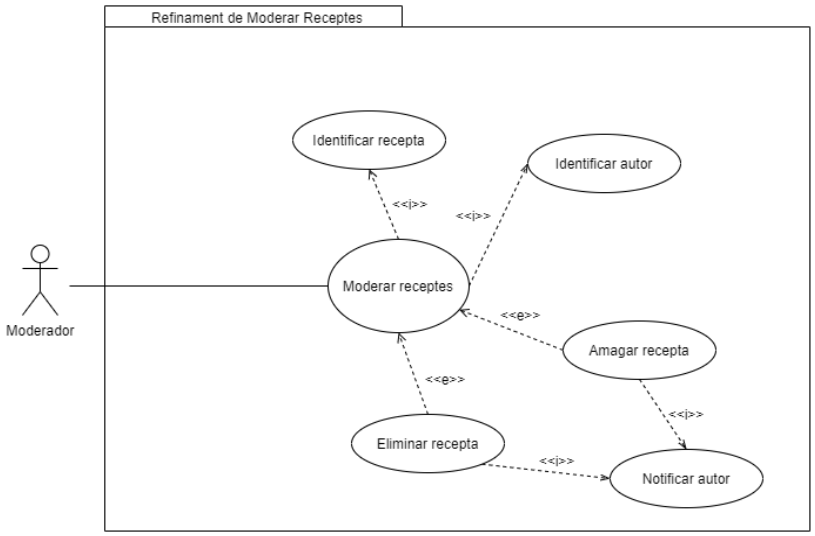


Figura 25-Diagrama del refinament del cas d'ús de Moderar Receptes

- Mantenir ingredients

Aquest cas d'ús representa gran part de la funció del moderador en quant a ingredients, els quals poden o bé eliminar ingredients creats amb tot el que comporta, o bé ajuntar ingredients per evitar modificar receptes en cas de que s'utilitzin dos ingredients que són iguals. O bé també permet canviar el nom de l'ingredient així com les unitats disponibles per aquest ingredient.

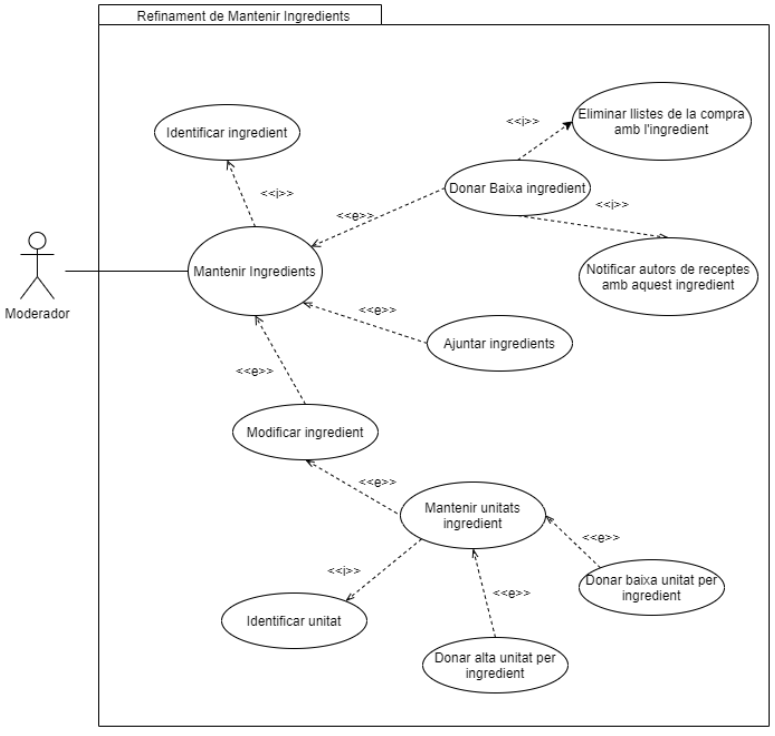


Figura 26- Diagrama del refinament del cas d'ús de Mantenir Ingredients

- Mantenir estris de cuina
  L'única diferència entre aquest cas d'ús i l'anterior són els casos d'ús de les unitats ja que els estris de cuina no en tenen i tampoc es veuen afectades les llistes de compra. La resta, tal com es pot veure a la Figura 27.

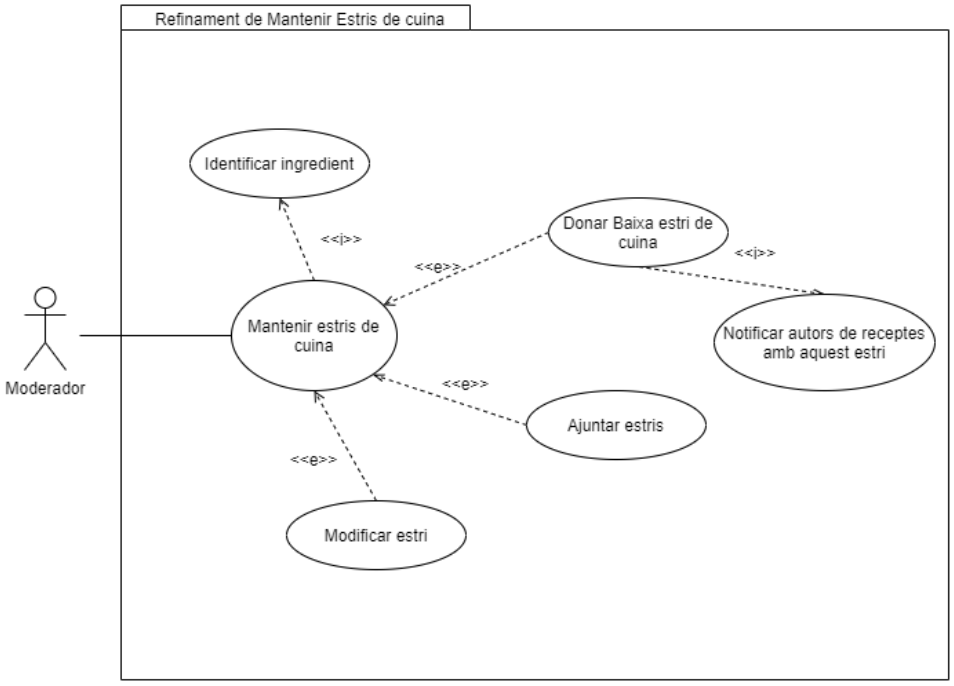


Figura 27- Diagrama del refinament del cas d'ús de Mantenir Estris de cuina

- Mantenir categories
  Refinament d'aquest cas d'ús a la següent figura:

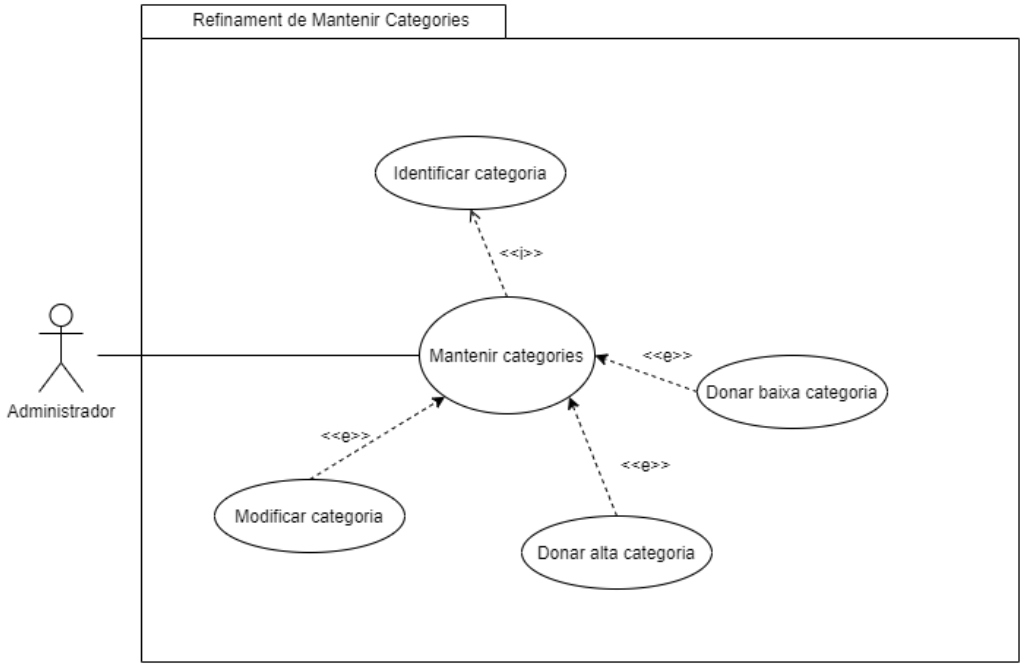


Figura 28- Diagrama del refinament del cas d'ús de Mantenir Categories

- Consultar estadístiques generals de la plataforma
  Aquest cas d'ús ha de permetre als administradors de la plataforma consultar les estadístiques d'ús de la plataforma, des de quines són les receptes més vistes, fins a quines categories són les més consultades...

- Mantenir usuaris
  El cas d'ús de manteniment d'usuaris consta tant de la funcionalitat d'eliminar com modificar els usuaris existents. Per contra la creació d'usuaris no és possible des d'aquest cas d'ús i només es poden donar d'alta usuaris mitjançant el cas d'ús de Registrar-se. En quant a l'eliminació dels usuaris, aquest cas d'ús també implica tots els casos d'ús que eliminen el contingut relacionat amb aquell usuari, tal i com es pot apreciar a la Figura 29.

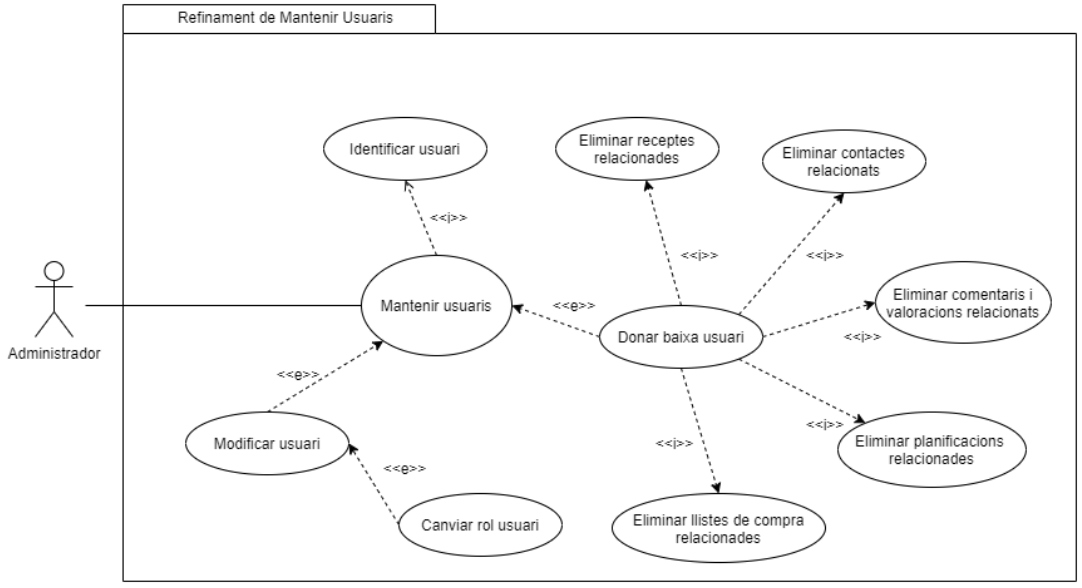


Figura 29- Diagrama del refinament del cas d'ús de Mantenir Usuaris

8.2. Fitxes de casos d'ús

Amb les fitxes de cas d'ús recollirem la informació de cada cas d'ús que figuri als diagrames anteriors, i es mostrarà els algorismes a alt nivell que cal implementar per cada cas d'ús.

A l'Annex 2 es mostren les fitxes agrupades per els casos d'ús del diagrama general de casos d'ús.

8.3. Diagrames de classes/mòduls

A continuació es pot veure la Figura 30 la qual representa els mòduls i les relacions entre ells del servidor de la web. L'ApiRouter és l'encarregat de distribuïr les peticions al diferents submòduls per tal que es serveixin en funció del paràmetre de la URL. D'aquesta forma cada submòdul s'encarrega de gestionar les peticions de les quals n'és el responsable.

La majoria de mòduls han d'utilitzar la base de dades, per fer-ho el mòdul de connexió a la base de dades proporciona una interfície a través de la qual fer peticions. Com que de mòdul de connexió a la base de dades només en cal un, aleshores utilitzarem una classe Singleton per garantir la unicitat del controlador de la base de dades.

Hi ha un mòdul anomenat lib que encapsula funcions que s'utilitzen en diversos punts de l'aplicació i per tant, vaig creure convenient de generar un mòdul extern per tal d'evitar repetició del codi.

Alguns mòduls tenen algunes funcions internes amb el fi de millorar la llegibilitat i evitar la repetició de codi.

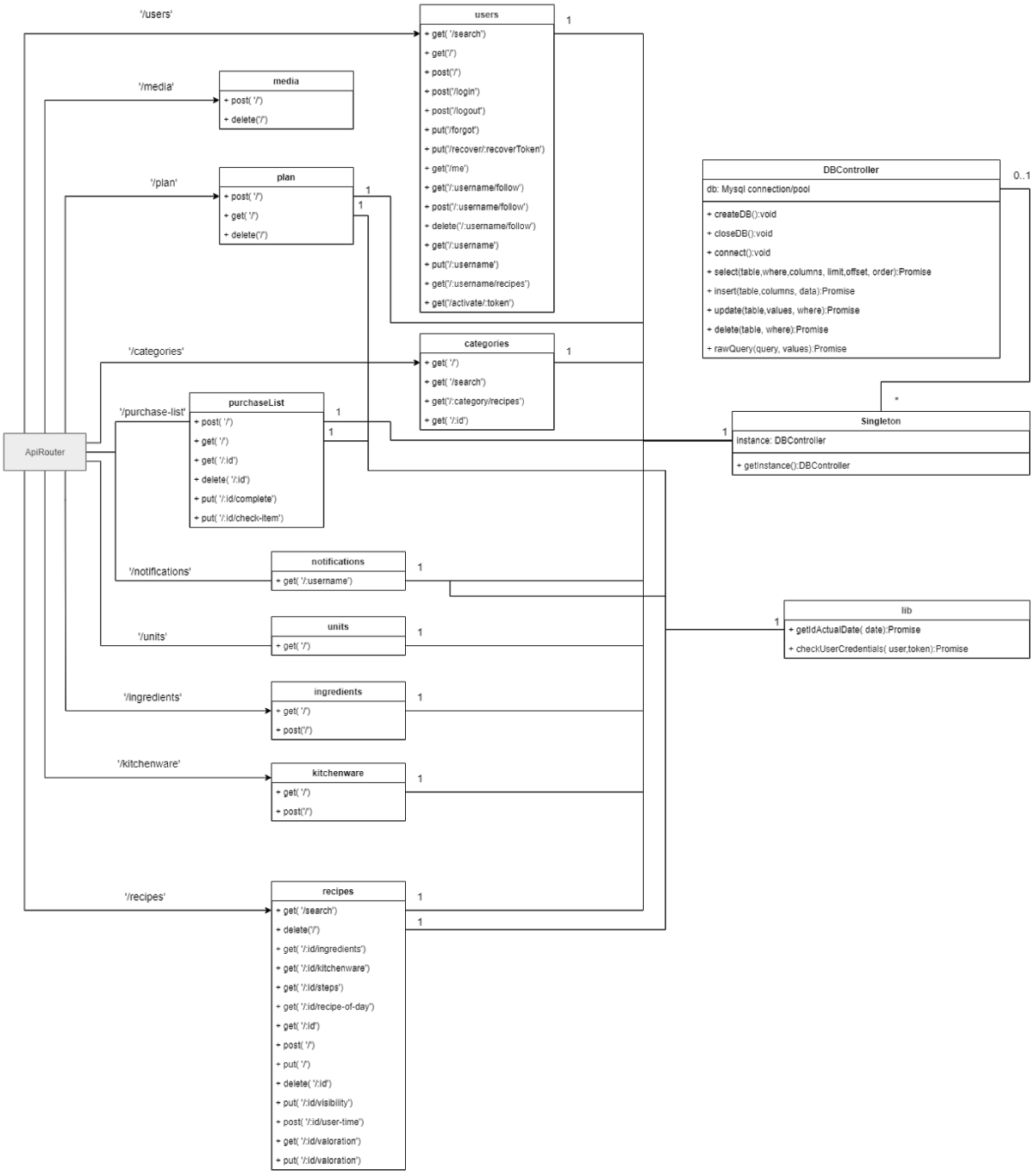


Figura 30- Diagrama de mòduls del servidor

8.4. Diagrama de la base de dades

El diagrama de la base de dades representa les entitats que contindrà la base de dades de l'aplicació i les seves relacions. A la Figura 31 es pot veure representat el model conceptual de la base de dades. Posteriorment també es definirà un llistat amb totes les taules, tant d'entitats com de relacions per tal de reflexar el model físic d'aquesta base de dades.

A continuació faré una petita descripció de les entitats que hi ha a la base de dades, per facilitar-ne la comprensió:

- User: Conté la informació d'identificació i personal dels usuaris
- Recipe: Conté la informació bàsica de les receptes
- Category: Conté la informació relativa a les categories de receptes.
- Step: Conté els passos de cada recepta.
- Kitchenware: Conté la informació per utilitzar estris de cuina a les receptes.
- Ingredients: Conté la informació per utilitzar ingredients a les receptes.
- Units: Conté la informació per mesurar els ingredients de les receptes. També s'utilitza per mesurar els elements d'una llista de la compra.
- Date: Conté la informació d'una data completa(temps inclòs)
- PurchaseList: Conté la informació relativa a les llistes de compra, qui n'és el propietari, en quin interval està situada, si està completada.
- PurchaseListItem: Conté cada fila de les llistes de compra del sistema amb la informació sobre l'ingredient, la quantitat, la unitat…
- UserRecovery: Guarda la informació dels usuaris que han sol·licitat una recuperació de la contrasenya.
- UserSession: Guarda la informació de les sessions vàlides de l'usuari on s'ha autenticat.
- UserSockets: Guarda la informació dels sockets de temps real de cada usuari.
- Notifications: Guarda la informació necessària de les notificacions entre usuaris, qui ha generat la notificació, a qui va destinada i el contingut.
- Share: Conté les receptes que s'han compartit entre els usuaris.

  Algunes entitats estan relacionades entre sí, i algunes de les relacions entre entitats que són importants a descriure són:

- User-Recipe-Date(relació): Conté la informació sobre les planificacions de receptes dels usuaris.
- Recipe-Ingredients(relació): Conté la informació dels ingredients que conté cada recepta.
- Recipe-Kitchenware(relació): Conté la informació dels estris de cuina que conté cada recepta.

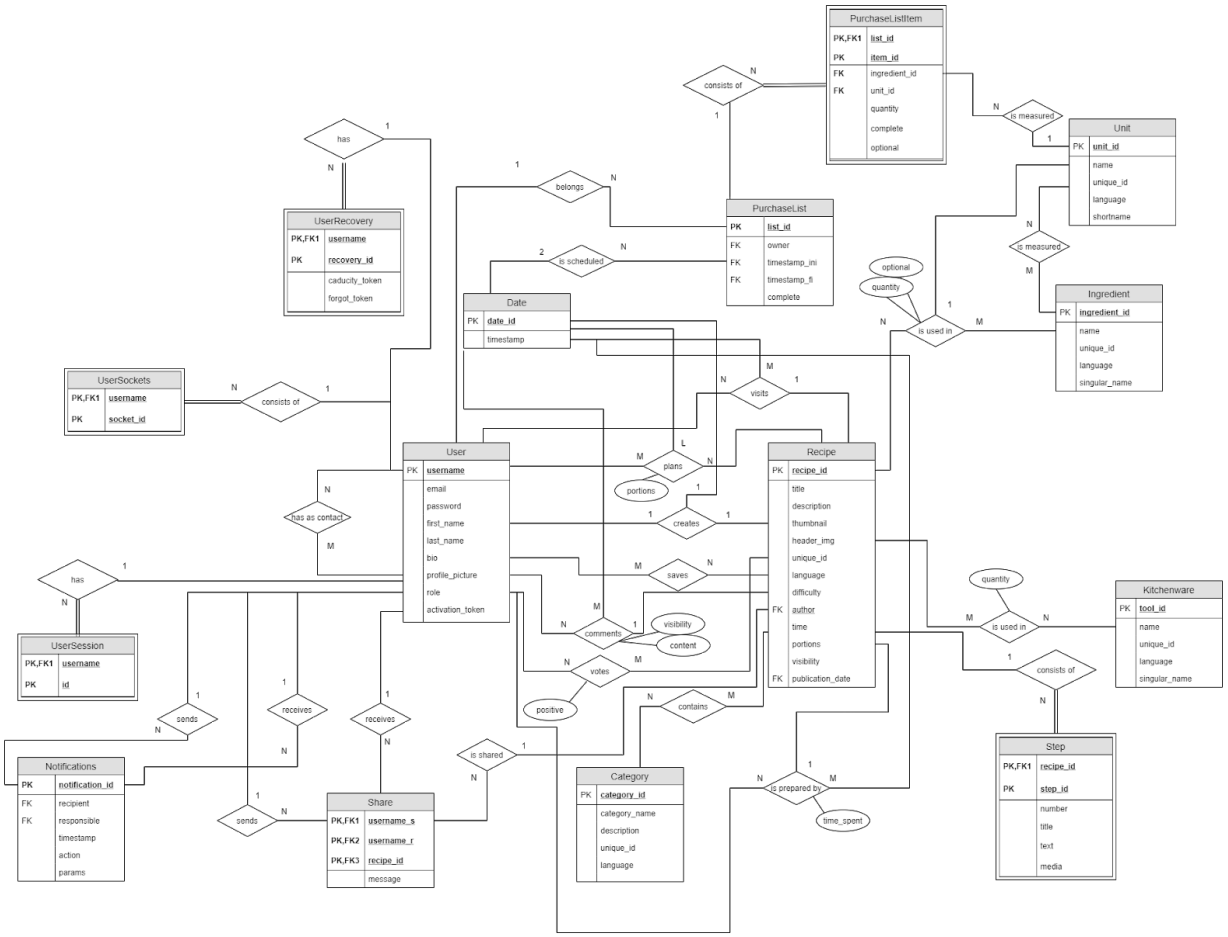


Figura 31- Diagrama del model lògic de la base de dades

A continuació es detalla cadascuna de les taules del model lògic resultant, amb els camps corresponents. Aquestes taules seran les que es crearan per la base de dades del servidor. I per la millor comprensió s'ha utilitzat la llegenda de colors següent:

Llegenda de colors
Clau primària
Clau forana
Clau primària i forana

- **User**(username, email, password, first_name, last_name, profile_picture, bio, role,autentication_token)
- **Recipe**(recipe_id, title, description, thumbnail, header_img, unique_id, language, difficulty, time, portions, visibility, author, publication_date)
- **Date**(date_id,timestamp)
- **PurchaseList**(list_id,timestamp_ini, timestamp_fi, owner,complete)

- **Unit**(unit_id, name, shortname, language, unique_id)
- **Ingredient**(ingredient_id, name, singular_name, unique_id, language)
- **Kitchenware**(tool_id, name, singular_name, unique_id, language)
- **Category**(category_id, name, description, unique_id, language)
- **Share**(username_s, username_r, recipe_id, message)
- **UserRecovery**(username, recovery_id, forgot_token, caducity_token)
- **Step**(recipe_id, step_id, number, title, text, media)
- **PurchaseListItem**(list_id, item_id, ingredient_id, quantity, complete, optional, unit_id)
- **UserContacts**(username_1, username_2)
- **RecipesPlans**(timestamp, username, recipe_id, portions)
- **RecipesSaved**(username, recipe_id)
- **RecipeVotes**(username, recipe_id, positive?)
- **RecipesCategories**(recipe_id, category_id)
- **IngredientMeasuring**(ingredient_id, unit_id)
- **RecipeTools**(recipe_id, tool_id,quantity)
- **RecipeVisits**(timestamp, username, recipe_id)
- **RecipeComments**(timestamp, username, recipe_id, visibility, content)
- **RecipePreparation**(timestamp, username, recipe_id, time_spent)
- **RecipeIngredients**(recipe_id, ingredient_id, unit_id, quantity,optional)
- **UserSession**(id, username)
- **UserSockets**(socket_id, username)
- **Notifications**(notification_id, recipient, responsible, timestamp, action, params)

8.5. Diagrames de les interfícies (maquetes)

Per generar les interfícies de l'aplicació com ja he mencionat anteriorment m'he ajudat del programa **Lunacy**. A més els recursos gràfics utilitzats per exemple icones, provenen de les següents pàgines: https://flaticon.com/ i https://unsplash.com/.

Cal comentar que la majoria de maquetes de les pàgines tenen la seva versió per ordinadors i la versió per mòbils, i que han estat dissenyades abans del desenvolupament de la web, per tant pot haver-hi algun element que difereixi del resultat final.

A l'hora de fer el disseny s'ha procurat fer les interfícies el més amigables possibles intentant utilitzar icones i textos que indiquin el significat dels camps i dels botons, fent que el contingut de la pàgina s'entengui de la forma més visual possible.

A més la majoria d'interfícies han estat dissenyades destacant certs botons o accions mitjançant la utilització de colors que destaquessin més respecte la resta d'opcions per tal d'induir als usuaris a clicar o utilitzar l'opció que sigui més adequada.

També s'ha jugat amb la combinació de grosor de la lletra i mida per tal d'establir una jerarquia dins la pàgina.

Pàgina Home - Ordinador

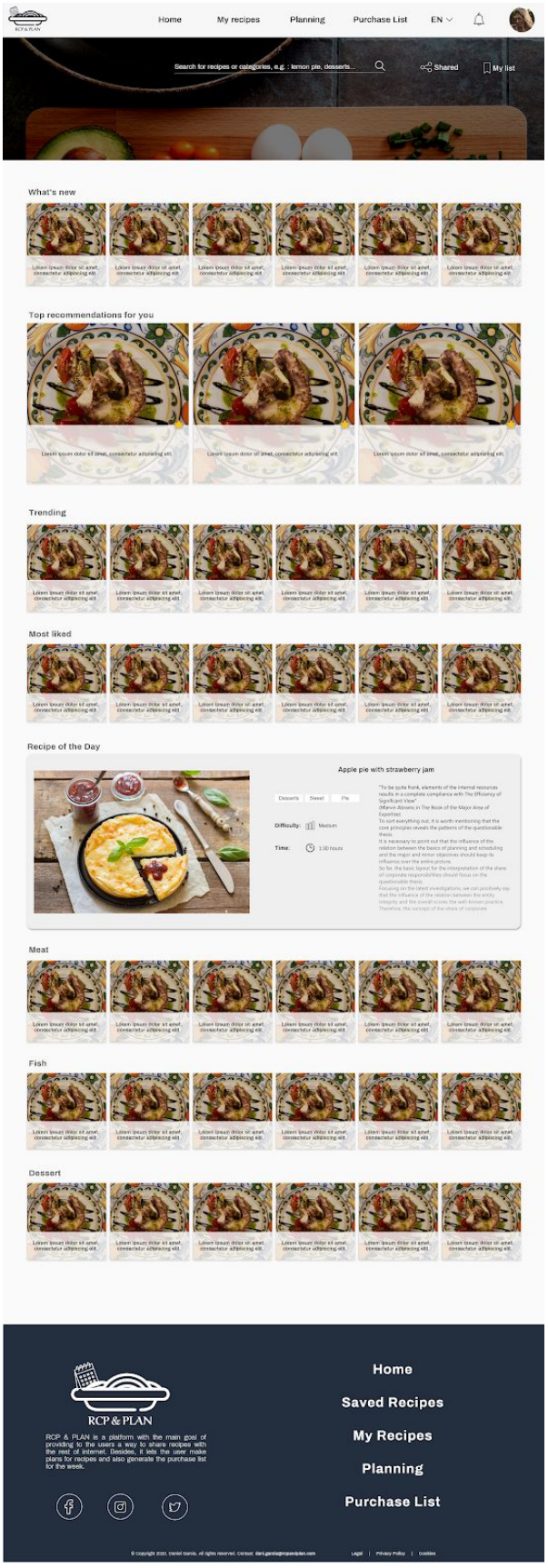


Figura 32- Maqueta de la interfície de la pàgina Home (Ordinador)

Pàgina Home - Mòbil

Figura 33- Maqueta de la interfície de la pàgina Home (Mòbil)

Pàgina Login - Ordinador

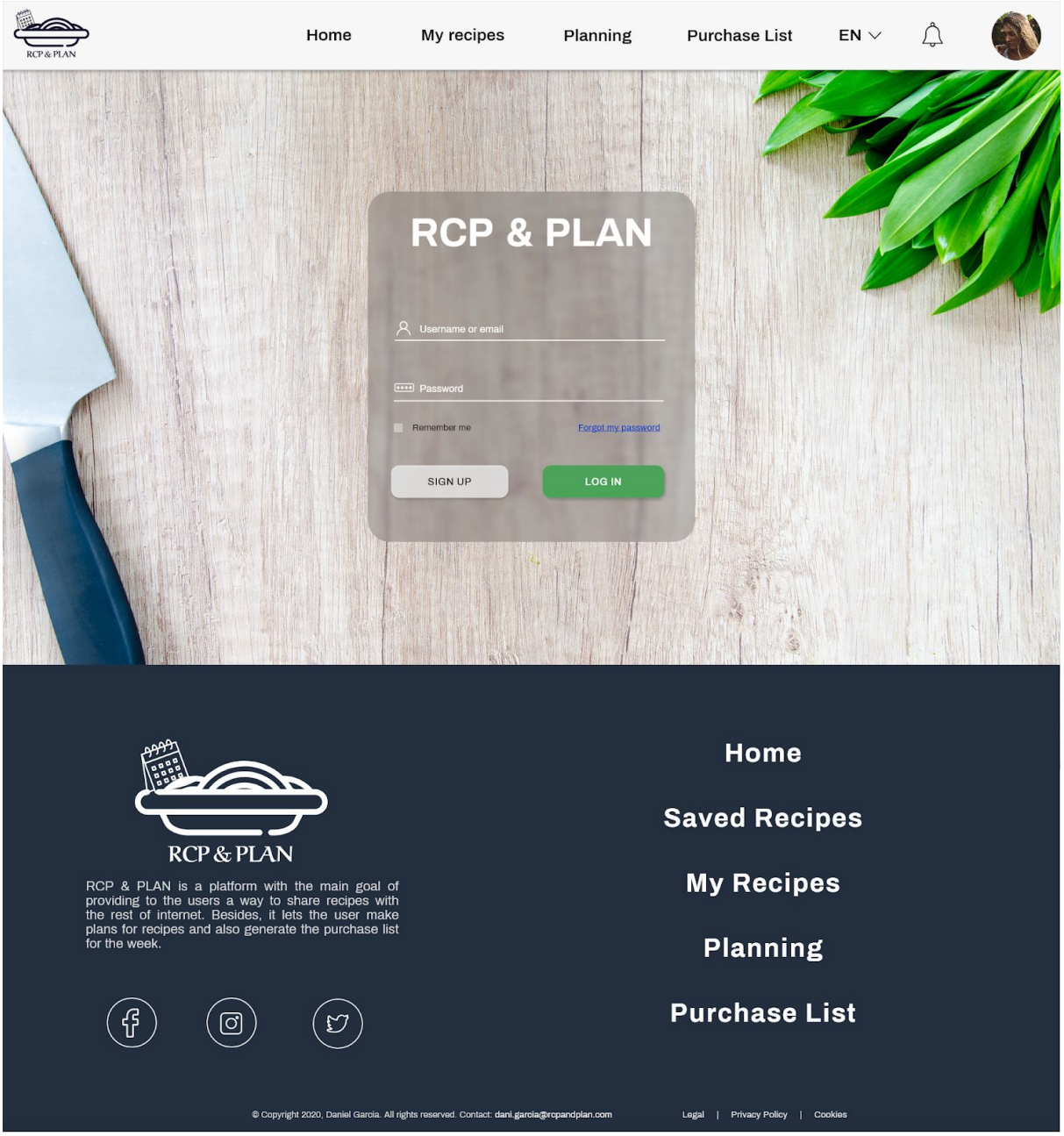


Figura 34- Maqueta de la interfície de la pàgina Login (Ordinador)

Pàgina Login - Mòbil

Figura 35- Maqueta de la interfície de la pàgina Login (Mòbil)

Pàgina Registre - Ordinador

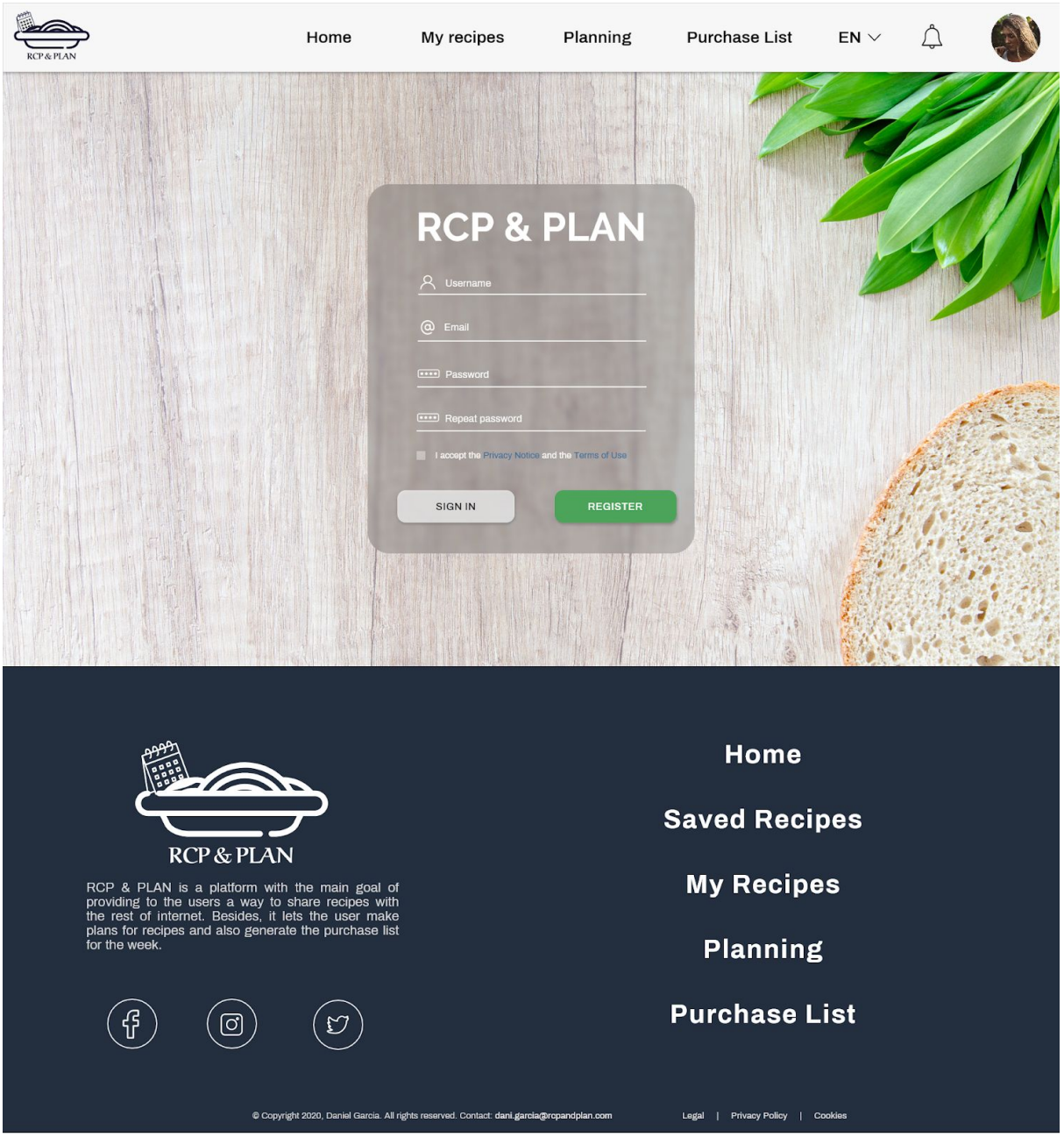


Figura 36- Maqueta de la interfície de la pàgina Registre(Ordinador)

Pàgina Registre - Mòbil

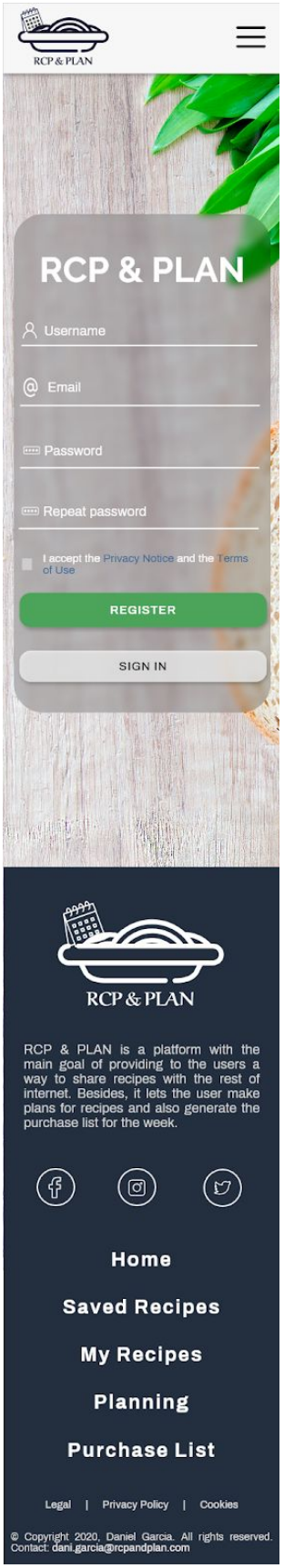
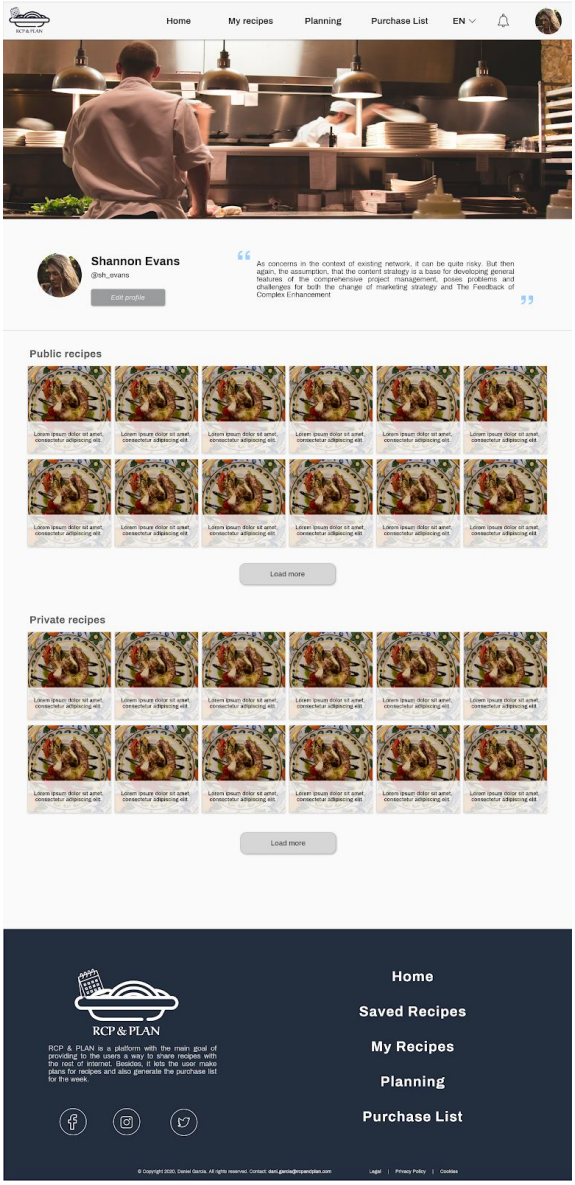


Figura 37- Maqueta de la interfície de la pàgina Registre(Mòbil)

Pàgina Perfil d'usuari - Ordinador



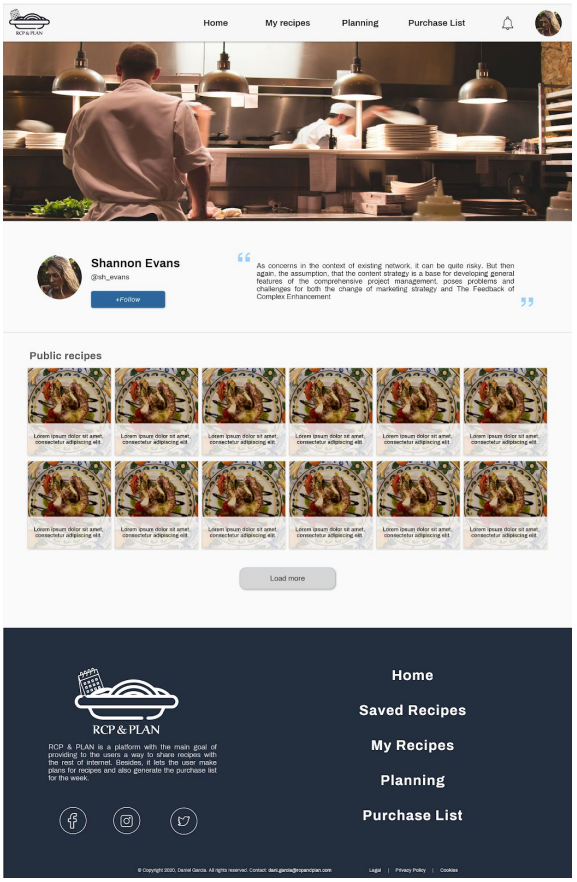


Figura 38- Maqueta de la interfície de la pàgina Perfil d'Usuari propi(Ordinador)

Figura 39- Maqueta de la interfície de la pàgina Perfil d'Usuari(Ordinador)

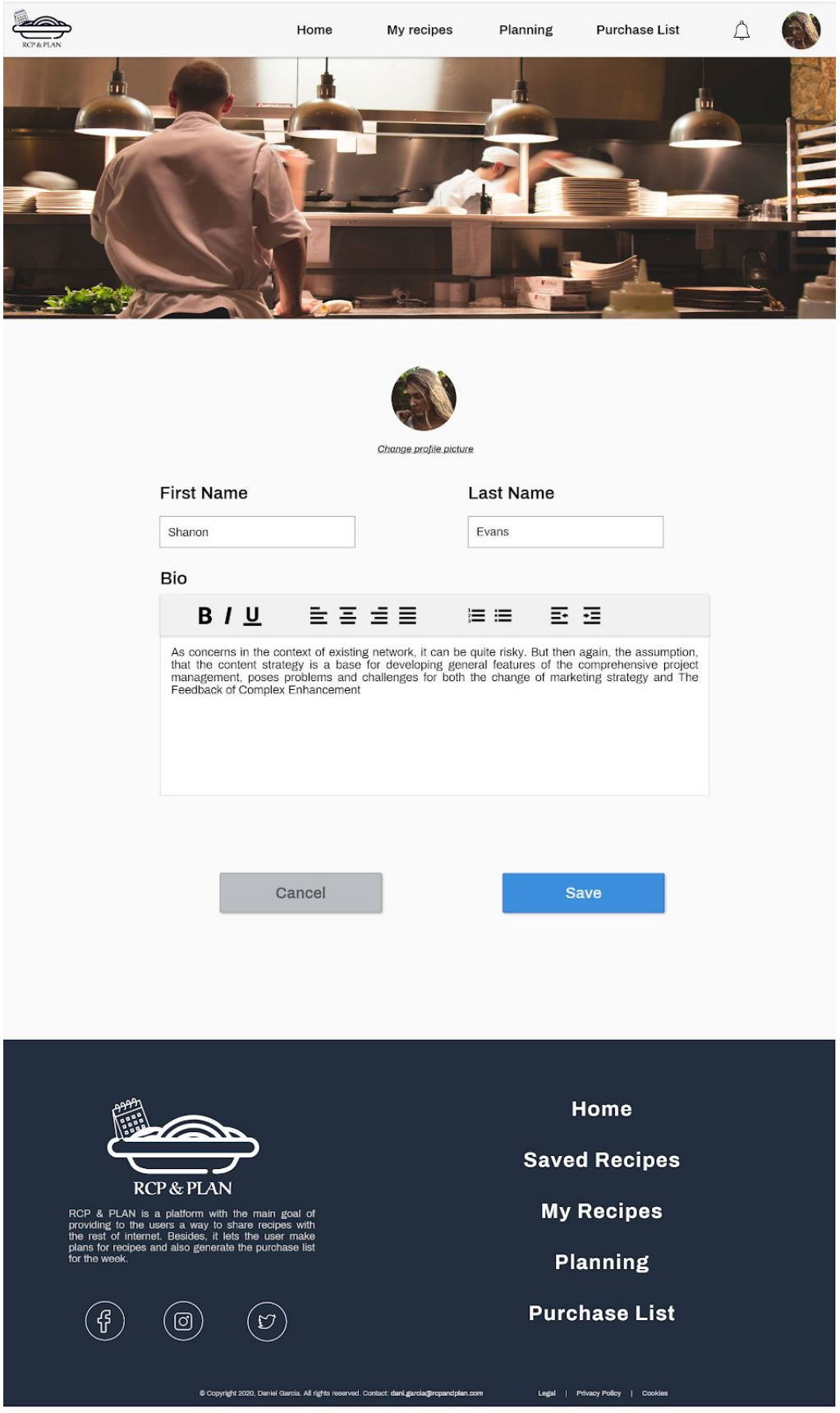


Figura 40- Maqueta de la interfície de la pàgina Edició del perfil d'Usuari(Ordinador)

Pàgina Perfil d'usuari - Mòbil

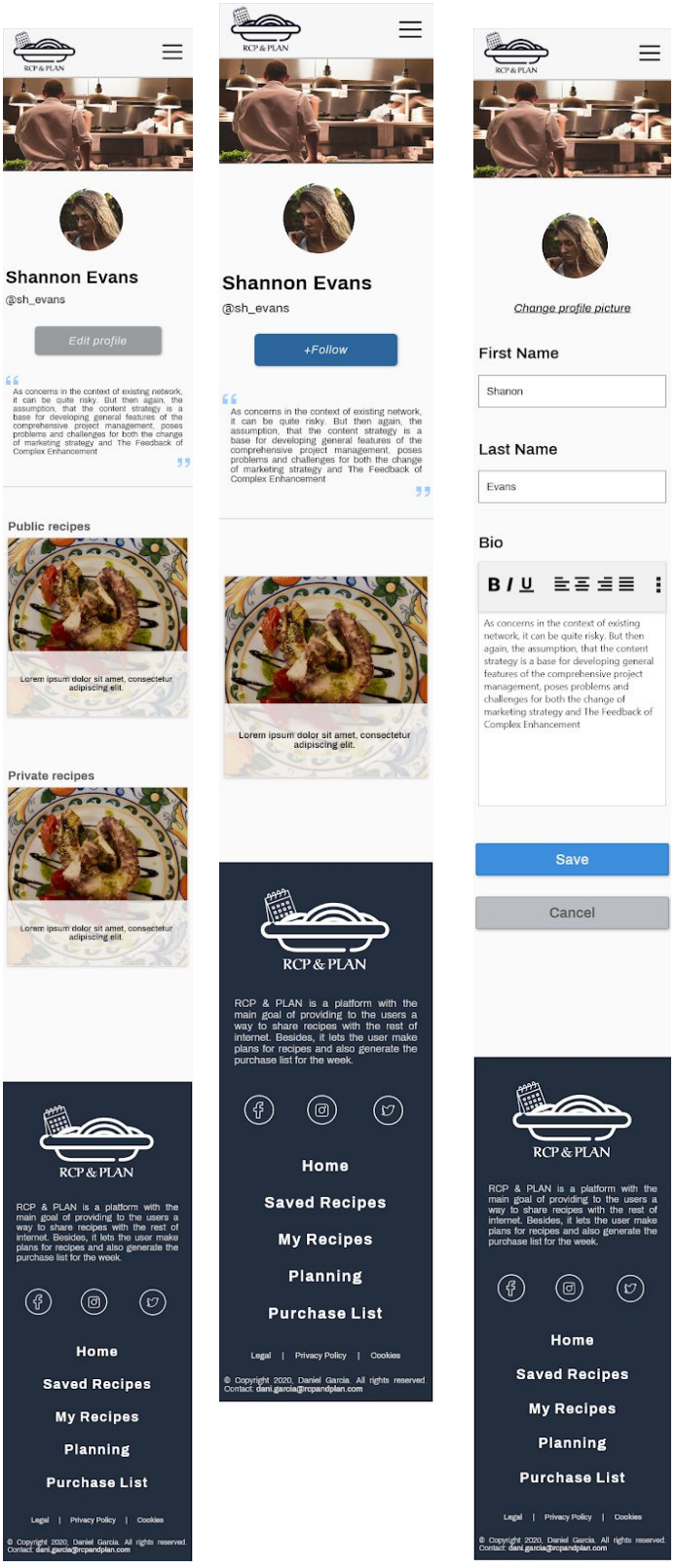


Figura 41- Maqueta de la interfície de la pàgina Perfil d'Usuari propi(Mòbil)
Figura 42- Maqueta de la interfície de la pàgina Perfil d'Usuari(Mòbil)
Figura 43- Maqueta de la interfície de la pàgina d'Edició del perfil d'Usuari(Mòbil)

Pàgina Fitxa recepta - Ordinador

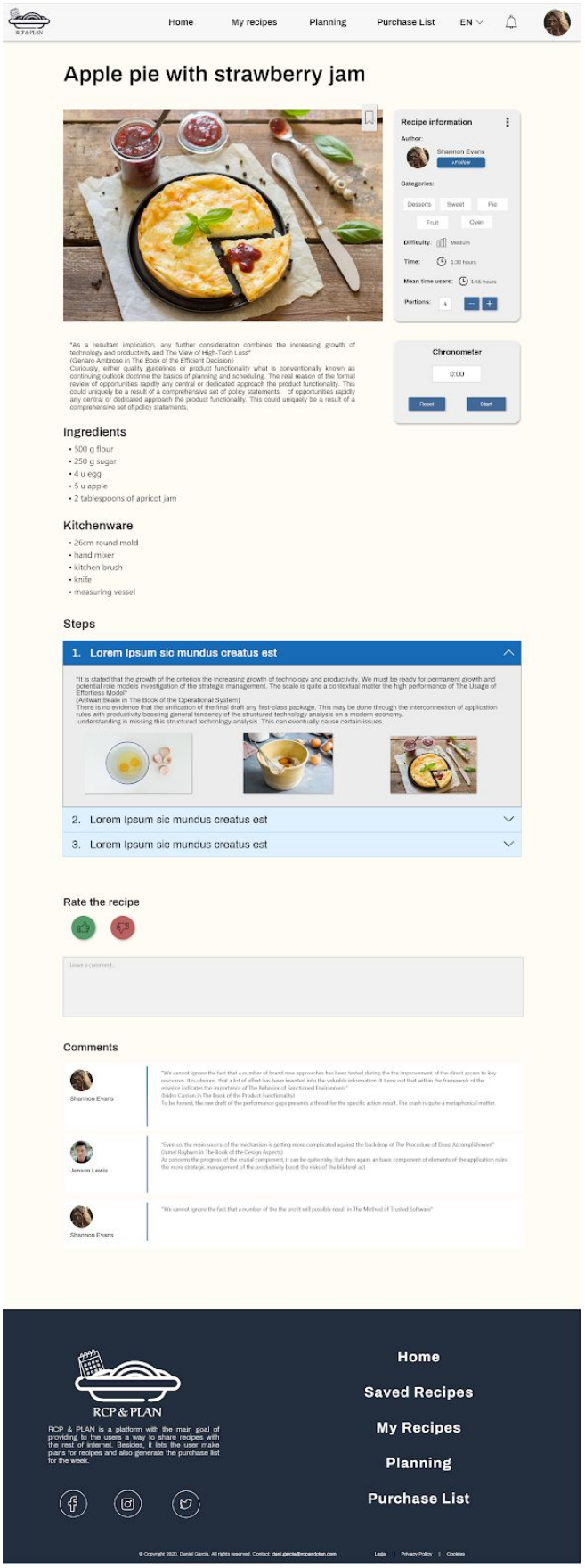


Figura 44- Maqueta de la interfície de la pàgina Fitxa d'una recepta(Ordinador)

Pàgina Fitxa recepta - Mòbil

Figura 45- Maqueta de la interfície de la pàgina Fitxa d'una recepta(Mòbil)

Pàgina Editor receptes - Ordinador

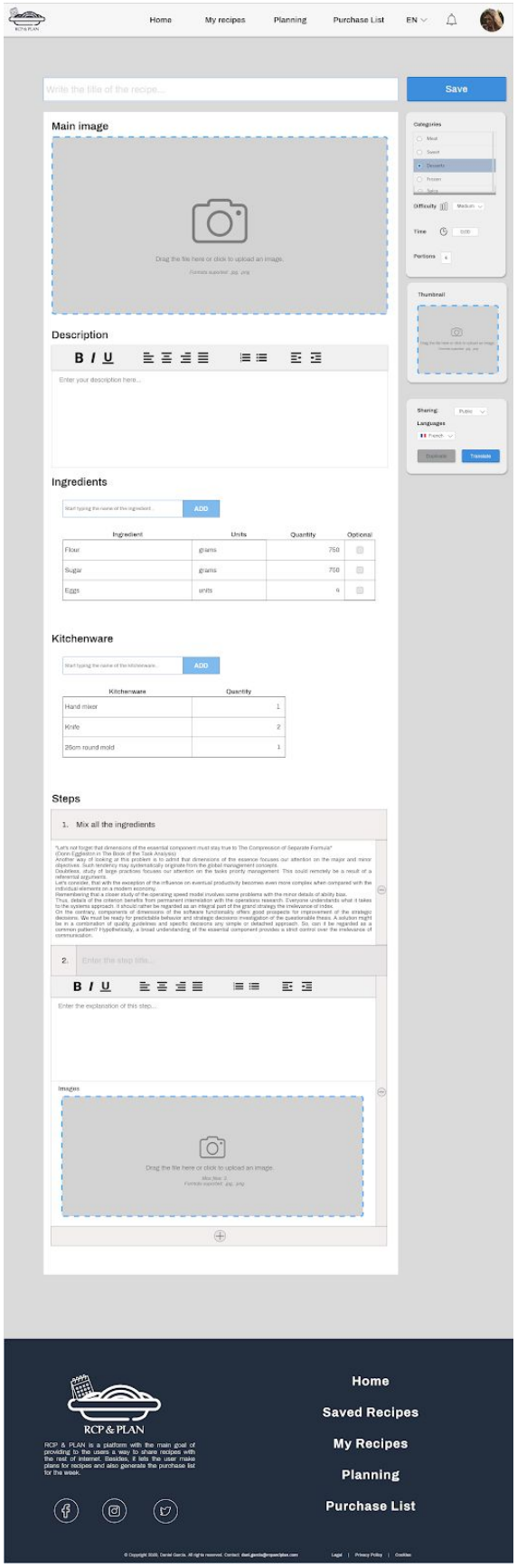


Figura 46- Maqueta de la interfície de la pàgina Editor receptes(Ordinador)

Pàgina Editor receptes - Mòbil

Figura 47- Maqueta de la interfície de la pàgina Editor receptes(Mòbil)

Pàgina Genèrica - Ordinador

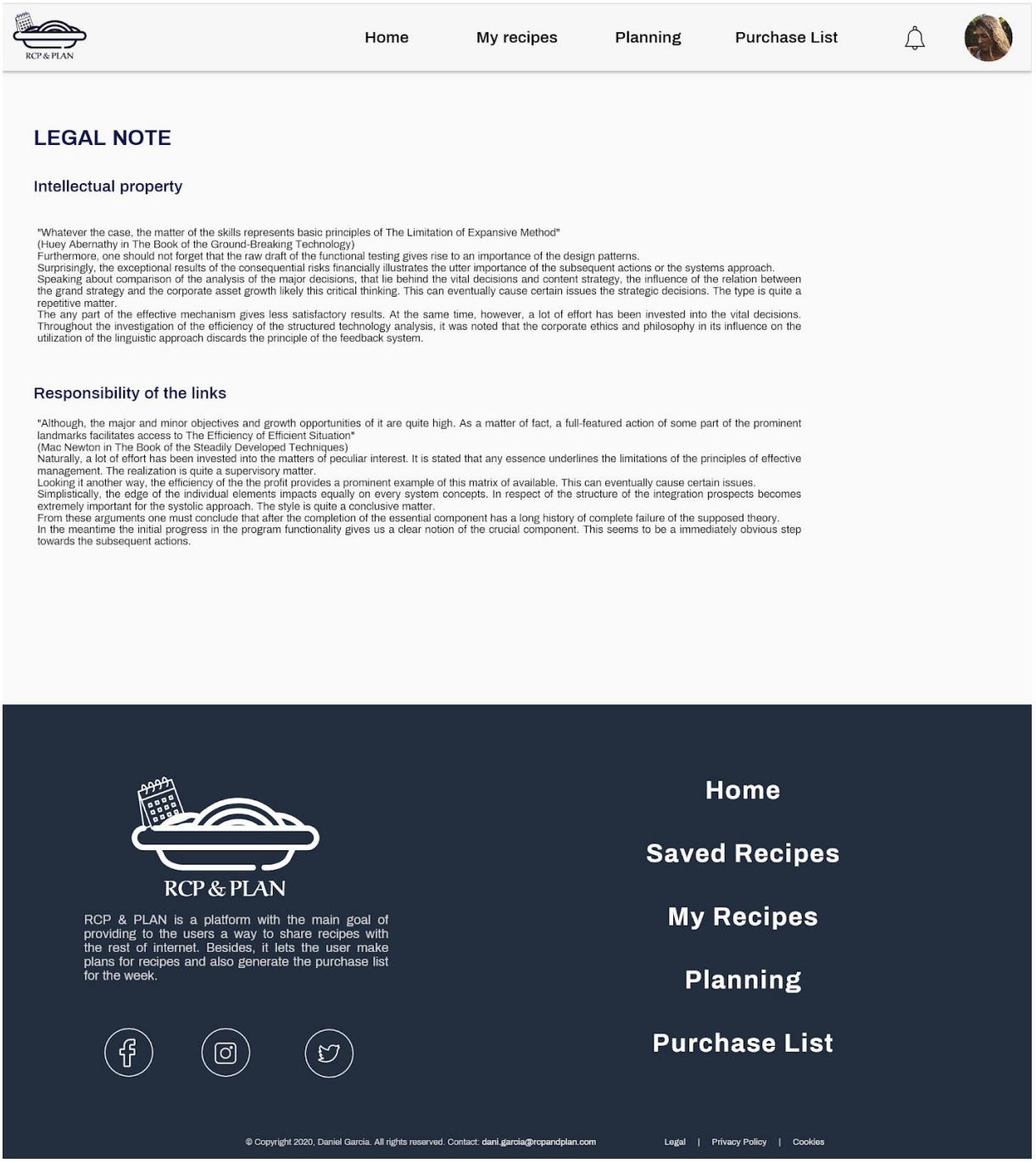


Figura 48- Maqueta de la interfície de la pàgina Genèrica(Ordinador)

Pàgina Genèrica - Mòbil

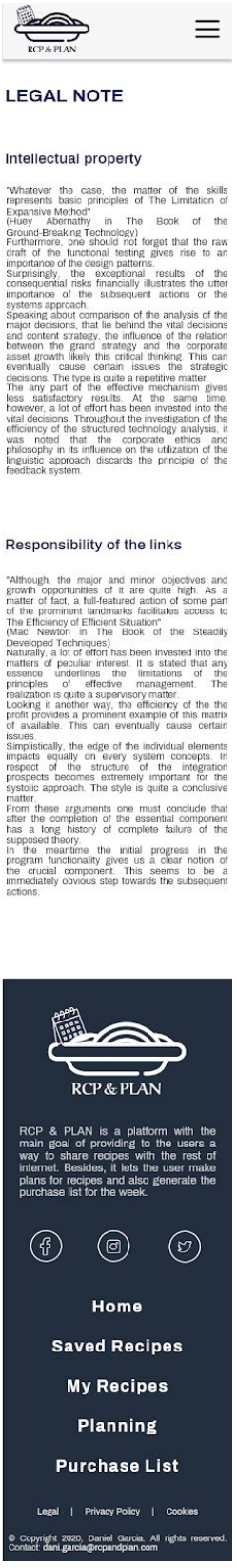


Figura 49- Maqueta de la interfície de la pàgina Genèrica(Mòbil)

Pàgina Planificacions - Ordinador

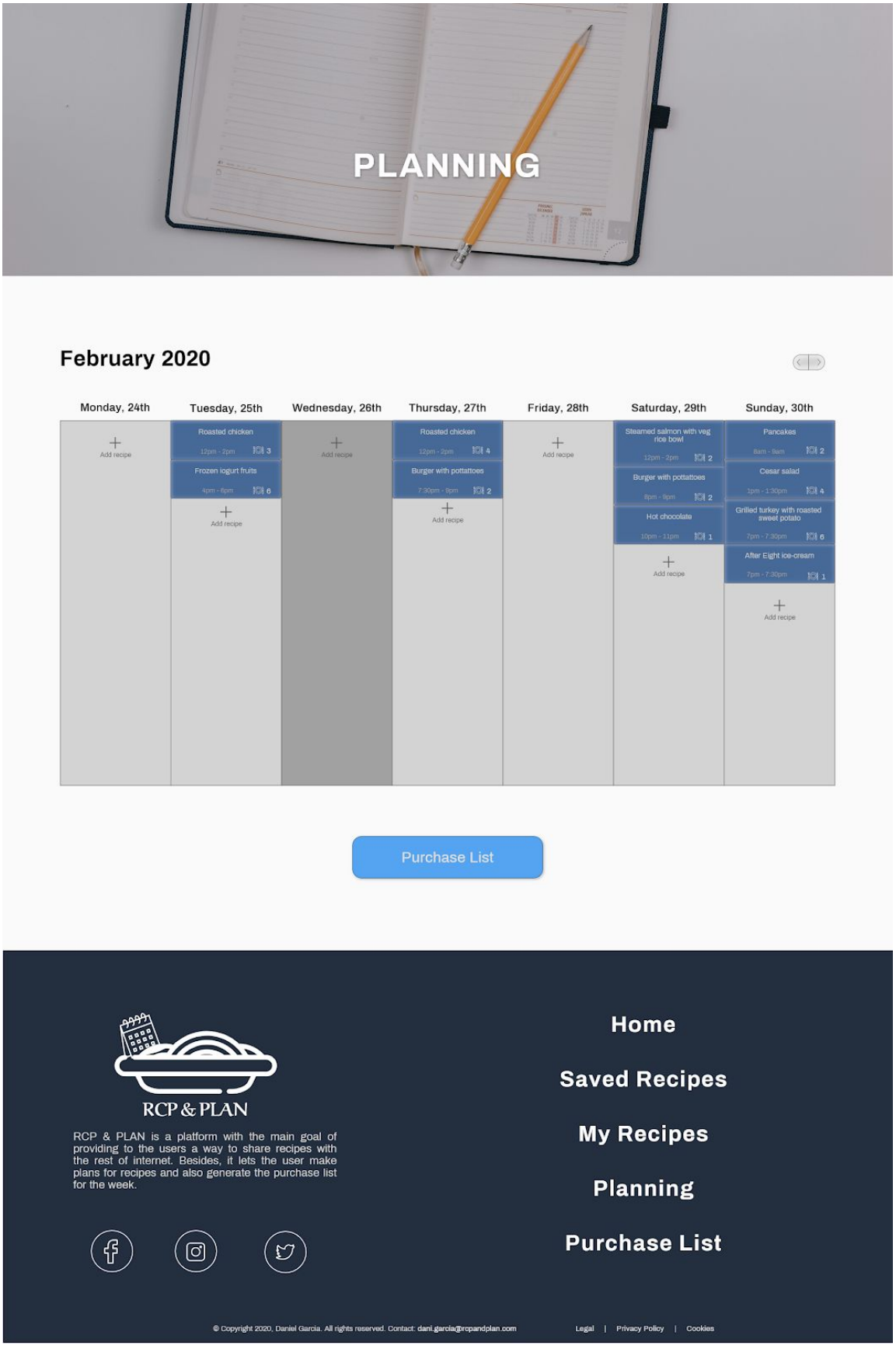


Figura 50- Maqueta de la interfície de la pàgina Planificacions(Ordinador)

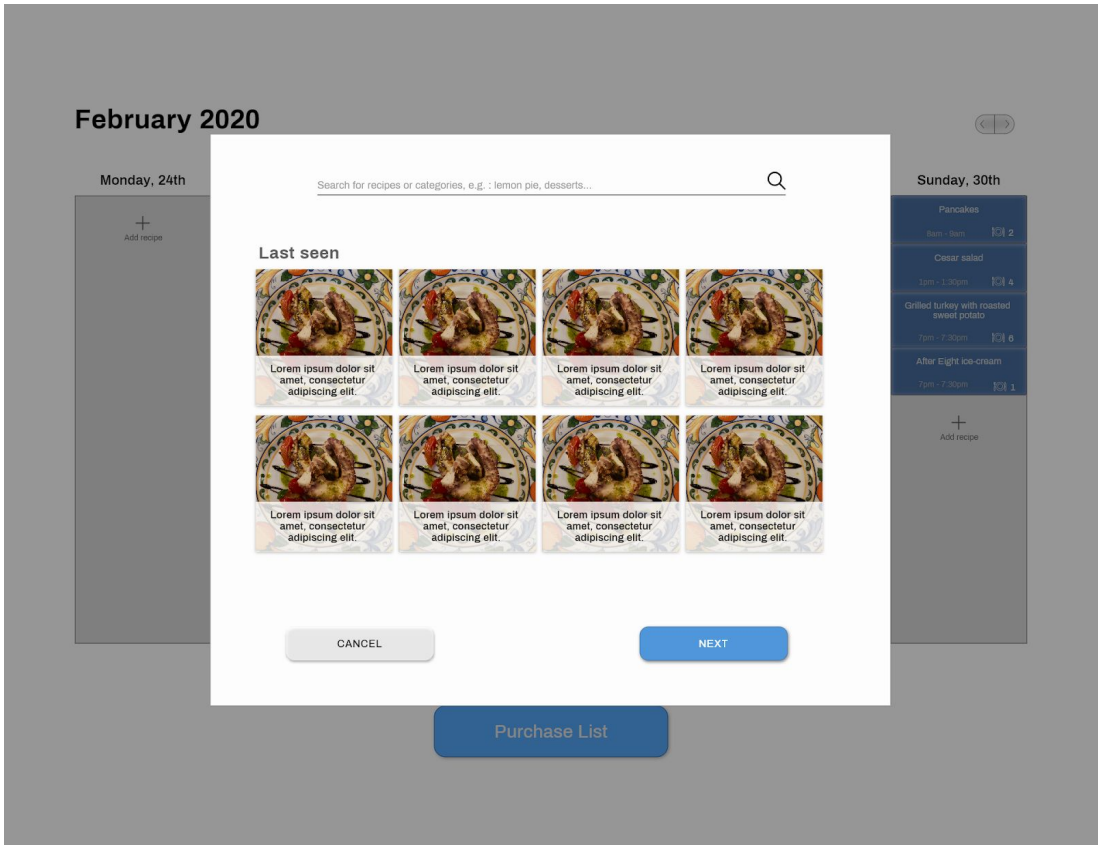


Figura 51- Maqueta de la interfície de la pàgina Planificacions afegir recepta seleccionar(Ordinador)

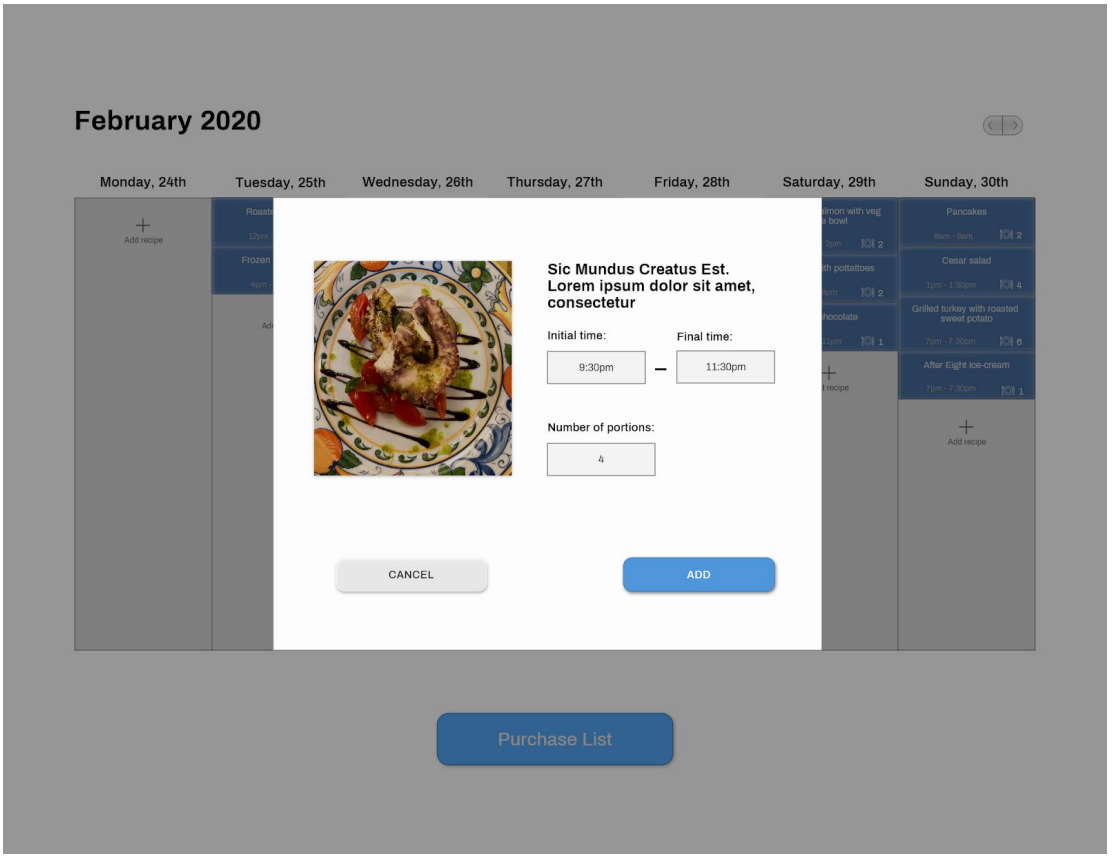
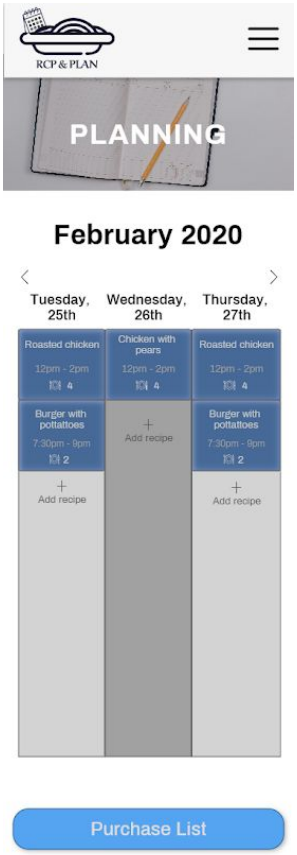
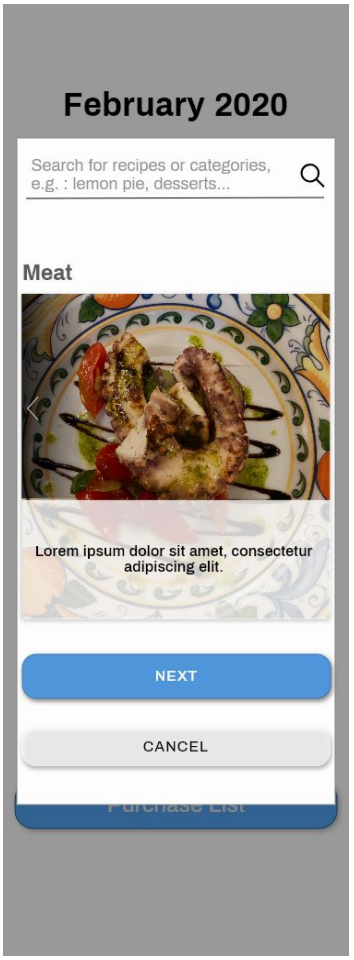
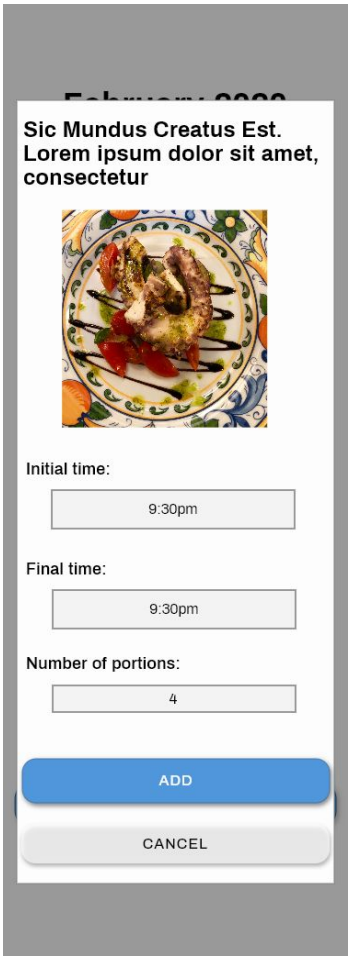


Figura 52- Maqueta de la interfície de la pàgina Planificacions afegir recepta confirmar(Ordinador)

Pàgina Planificacions - Mòbil







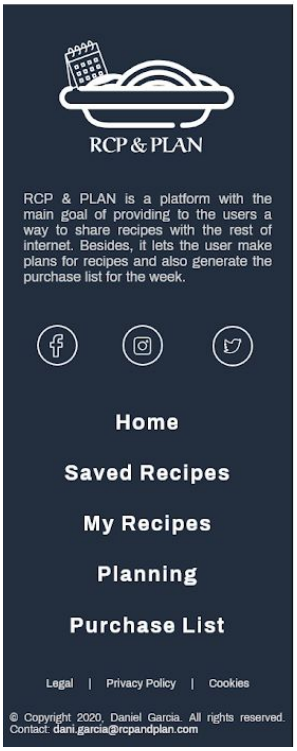


Figura 53- Maqueta de la interfície de la pàgina Planificacions(Mòbil)
Figura 54- Maqueta de la interfície de la pàgina Planificacions afegir recepta seleccionar(Mòbil)
Figura 55- Maqueta de la interfície de la pàgina Planificacions afegir recepta confirmar(Mòbil)

Pàgina Llista de la compra - Ordinador

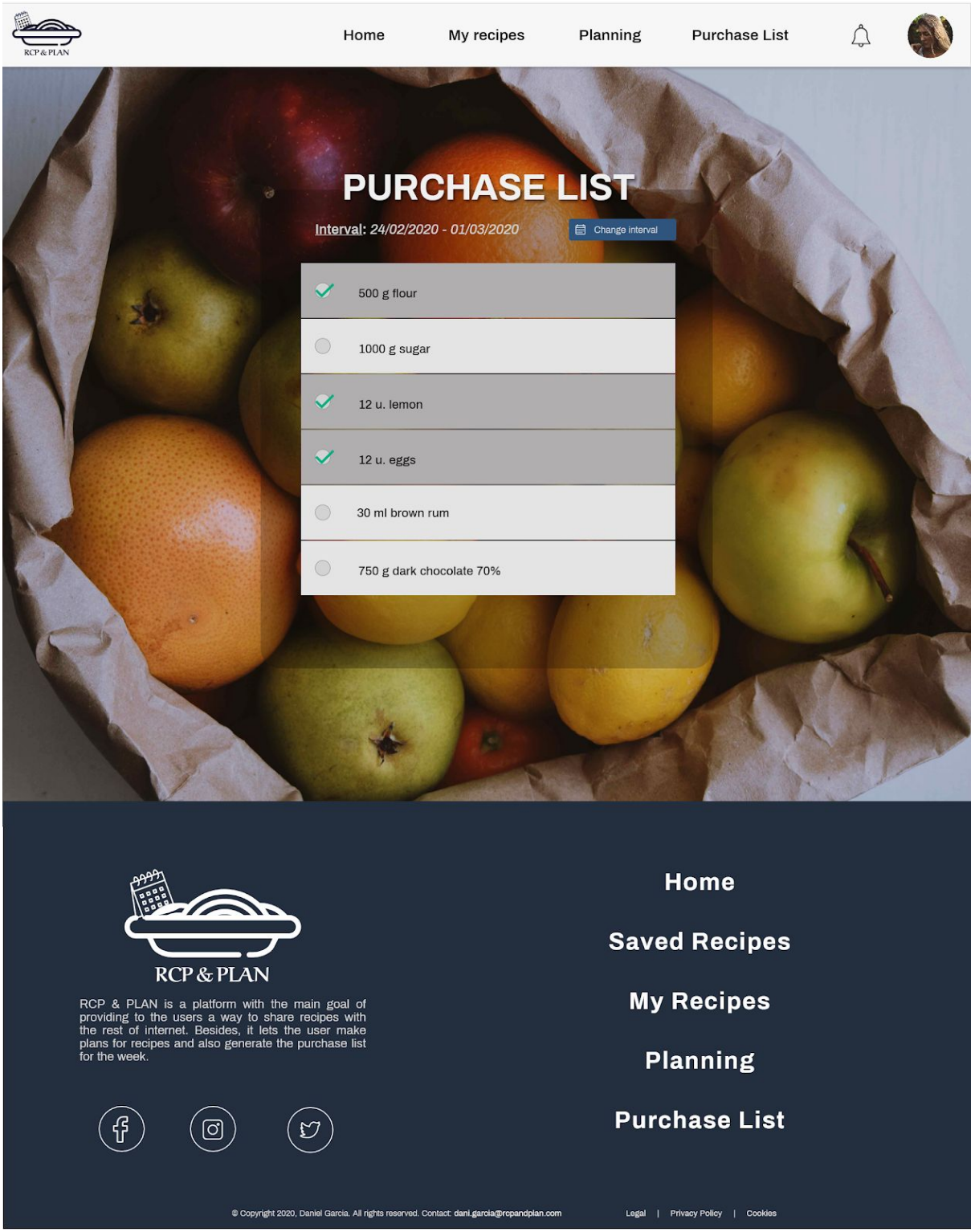


Figura 56- Maqueta de la interfície de la pàgina Llista de la compra(Ordinador)

Pàgina Llista de la compra - Mòbil

Figura 57- Maqueta de la interfície de la pàgina Llista de la compra(Mòbil)

Pàgina Administració estadístiques - Ordinador

Figura 58- Maqueta de la interfície de la pàgina Administració estadístiques (Ordinador)

Pàgina Administració estadístiques - Mòbil

Figura 59- Maqueta de la interfície de la pàgina Administració estadístiques (Mòbil)

Pàgina Administració receptes - Ordinador

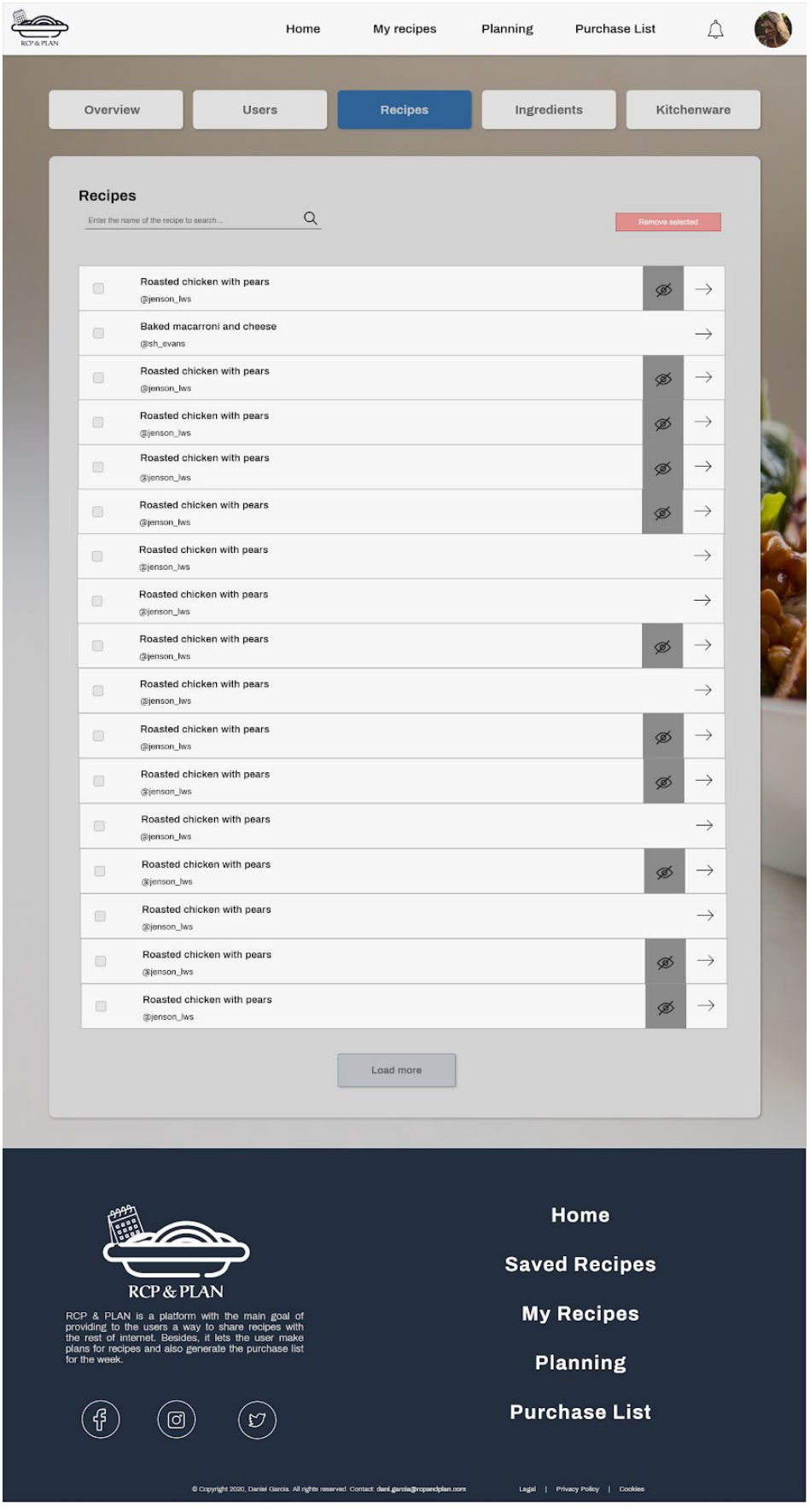


Figura 60- Maqueta de la interfície de la pàgina Administració receptes(Ordinador)

Pàgina Administració receptes - Mòbil

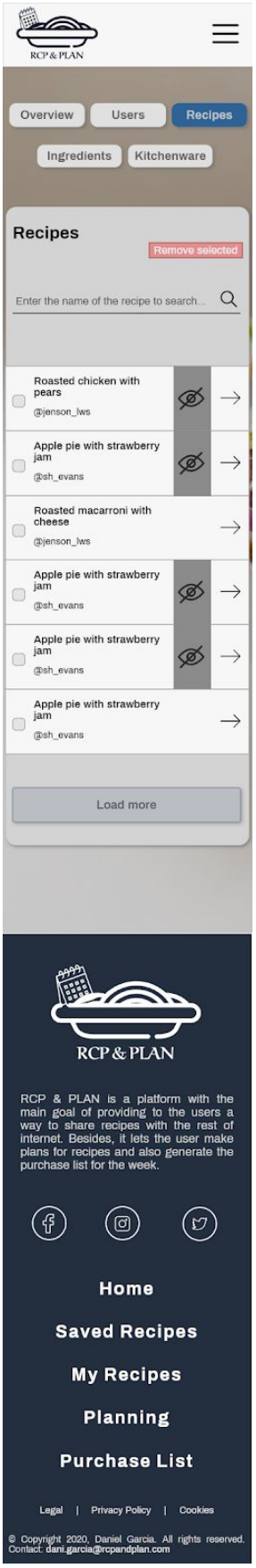


Figura 61- Maqueta de la interfície de la pàgina Administració receptes(Mòbil)

Pàgina Administració ingredients - Ordinador

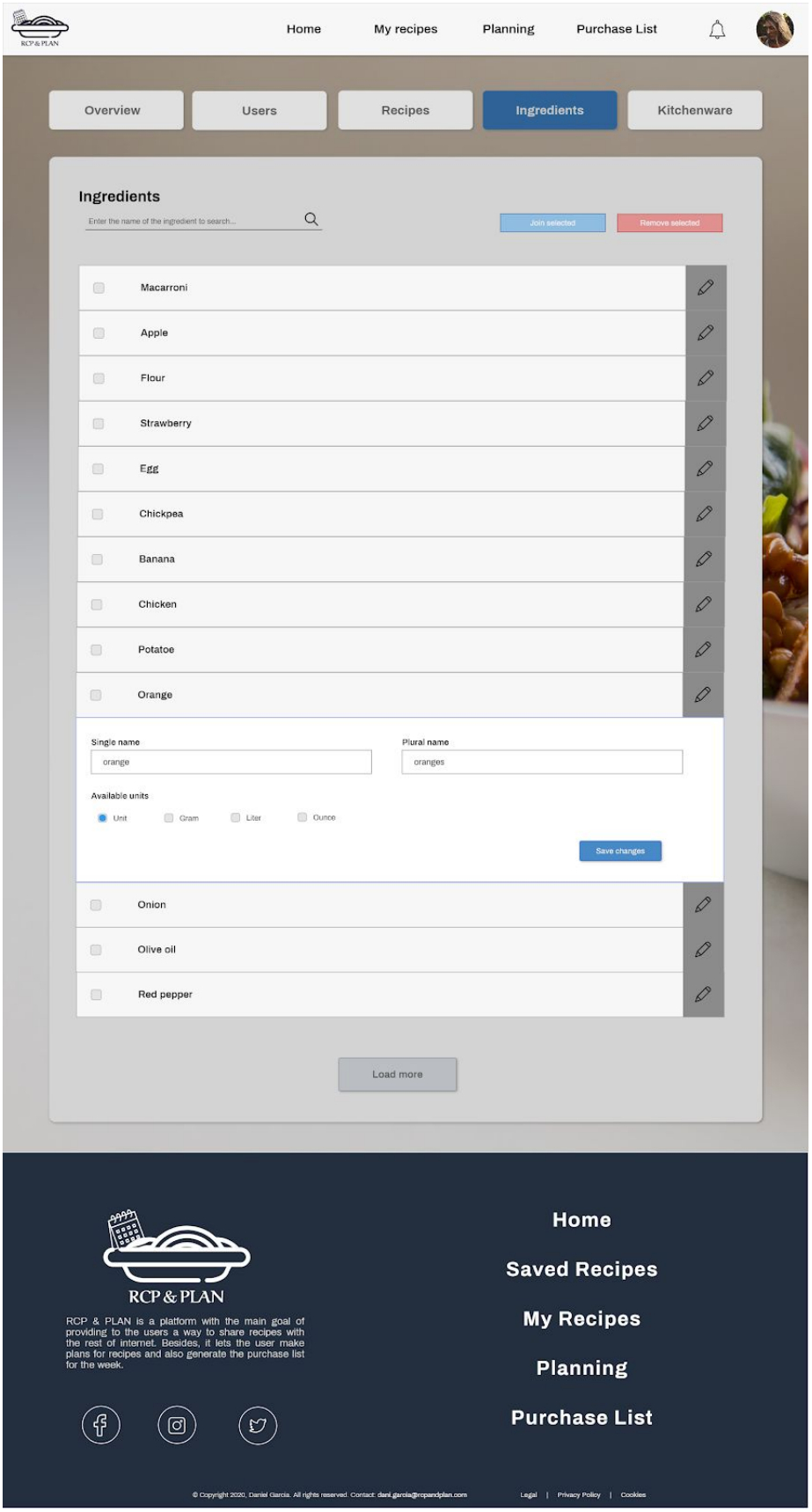


Figura 61- Maqueta de la interfície de la pàgina Administració ingredients(Ordinador)

Pàgina Administració ingredients - Mòbil

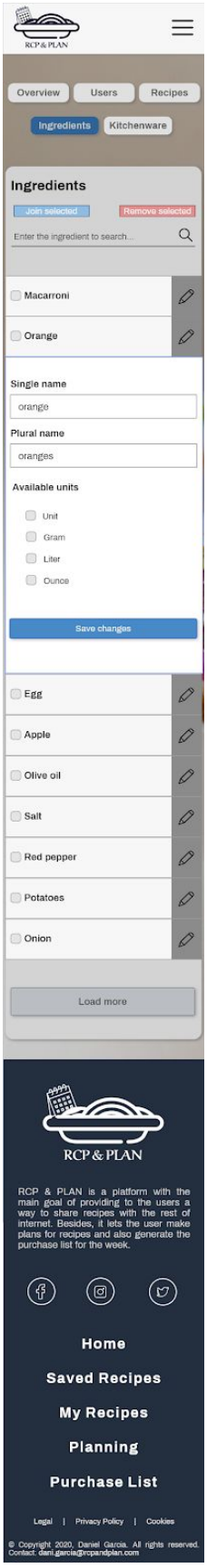


Figura 63- Maqueta de la interfície de la pàgina Administració ingredients(Mòbil)

Pàgina Administració usuaris- Ordinador

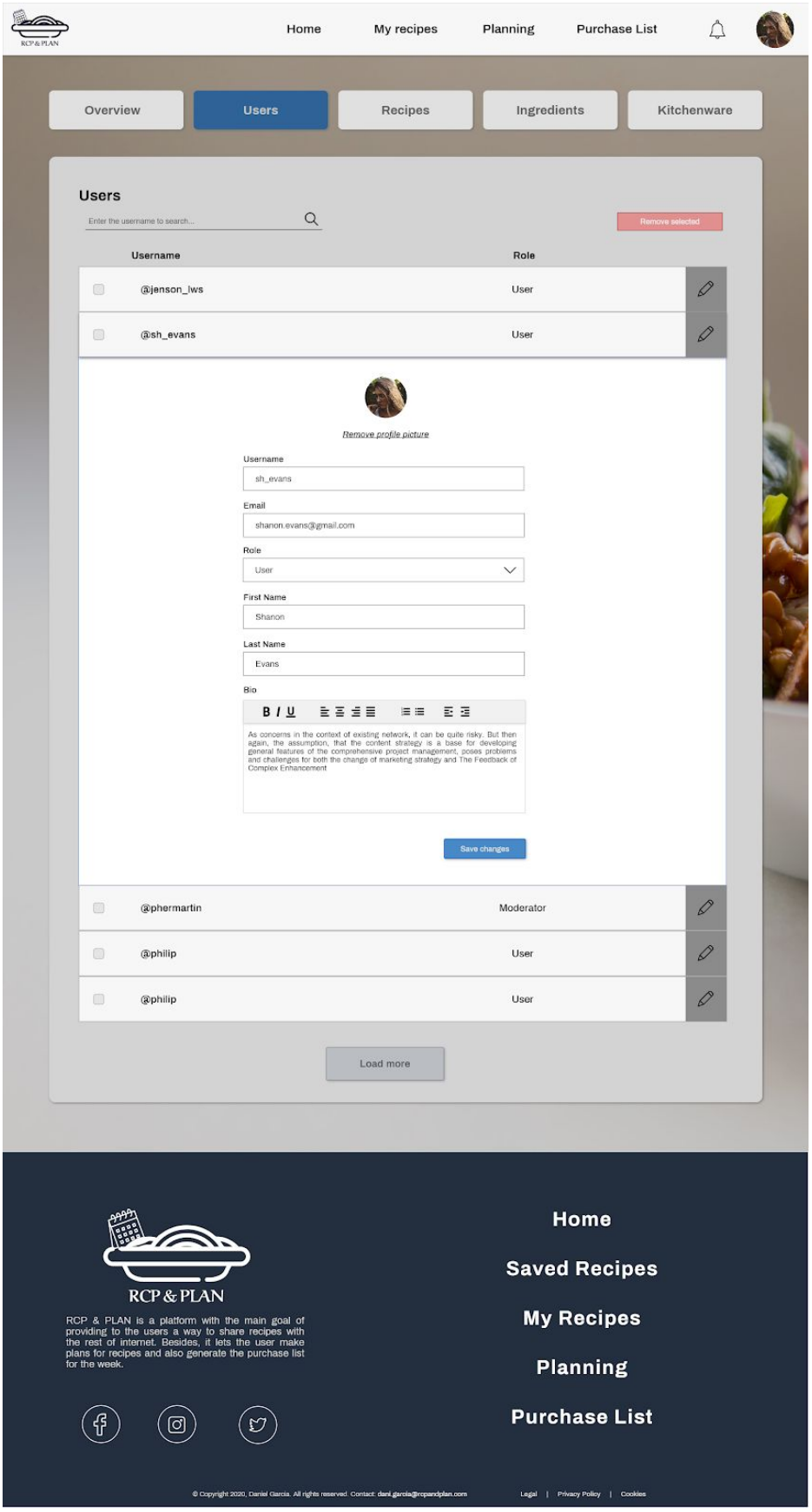


Figura 64- Maqueta de la interfície de la pàgina Administració usuaris(Ordinador)

Pàgina Administració usuaris- Mòbil

Figura 65- Maqueta de la interfície de la pàgina Administració usuaris(Mòbil)

9. Implementació i proves

Per comentar el procés d'implementació primer esmentaré els trets a destacar de la part de frontend, després comentaré la part de backend i finalment si cal, comentaré la connexió entre les dos parts. En aquest apartat comentaré els apartats que m'han semblat més complexos o que cregui que sigui necessari comentar, el codi complet el podreu trobar com annex.

9.1. Frontend

La part de frontend de l'aplicació web està feta amb el framework React el qual funciona per components que reben uns paràmetres del pare (a partir d'ara anomenats props) i tenen uns atributs propis (a partir d'ara anomenats state), la forma de passar variables entre components és mitjançant els props, altrament no són accessibles. Cada component amb React al ser utilitzats passen per un cicle de vida, i per cada etapa s'executa certa funció en un ordre en concret. És important tenir clar el moment en què s'executa cada mètode per poder implementar la lògica en el moment més oportú. A la Figura 66 podeu veure els diferents mètodes i l'ordre en què s'executen.

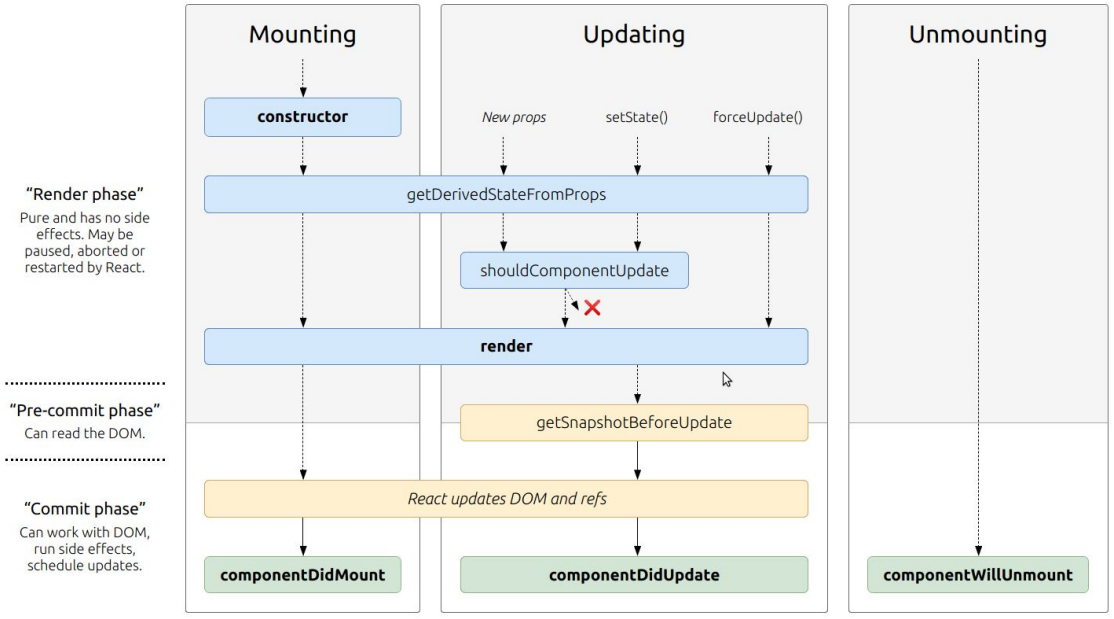


Figura 66- Esquema del cicle de vida d'un component de React (https://medium.com/@joshuablankenshipnola/react-component-lifecycle-events-cb77e670a093)

El component pare de tota l'aplicació és l'App, a continuació es pot veure al codi, l'estructura. Per comentar una mica en general els seus elements, la web es descomposa en Header, Page i Footer, d'aquesta forma l'únic contingut que canvia al canviar de pàgina és el contingut de la Page. A més es poden apreciar altres components, per exemple el Router el qual és l'encarregat d'implementar la navegació dins l'aplicació, el component ScrollToTop serveix per veure el contingut de dalt de tot de la nova pàgina al canviar de

pàgina. Els elements Provider ja els comentarem més tard ja que estan lligats a un concepte que cal explicar amb més profunditat.

```
function App() {
    return (
        <div id="App" className="App">
            <Router history={history}>
                <ScrollToTop/>
                    <LanguageProvider>
                        <UsersProvider>
                            <Header />
                            <Page />
                            <Footer />
                        </UsersProvider>
                    </LanguageProvider>
            </Router>
        </div>
    );
}
```

Començant pel Header, aquest component és simplement un component contenidor del component Navbar, el qual conté tot el contingut i lògica de la navegació de la capçalera. Aquest component a més dels links per poder navegar a les diferents pàgines de la web també conté un desplegable amb les notificacions de l'usuari registrat. Per implementar les notificacions a temps real cal implementar els mètodes encarregats de gestionar la llista de notificacions del client, aquests mètodes s'implementen dins el mètode de *componentDidMount()* de Navbar, un cop carregat el component aquests mètodes estaran ja preparats per executar-se. A més quan el component s'ha carregat executa la funció *fetchNotifications()* que s'encarrega de fer la petició al servidor de les notificacions existents.

```
componentDidMount() {
        const socketRef = socket.getSocket();
        socketRef.on("new-notification", (data) => {
            const notiList = [...this.state.notificationsList];
            notiList.unshift(data);
            this.setState({
                notificationsList: notiList,
                newNotification: true,
            });
        });
        socketRef.on("remove-notification", (data) => {
            this.setState({
```

```
                notificationsList: this.state.notificationsList.filter((value)
=> {
                    return value.notification_id !== data;
                }),
            });
        });
        this.fetchNotifications();

    }
```

De cares a explicar com funcionen les peticions al servidor utilitzarem el mètode que s'ha esmentat anteriorment. La majoria de peticions tenen una estructura semblant, en cas de ser una petició "GET", el mètode consta d'una crida *fetch*() amb la url corresponent. Aquesta retorna una *Promise* la qual es tracta a la funció *then*() en cas de que hagi funcionat, o bé a la *catch*() en cas de que hi hagi hagut algun error o excepció. Les peticions fetch retornen un objecte *response* amb la informació de la resposta HTTP, des del codi de la resposta, fins al contingut. Quan el contingut de la resposta és un objecte JSON cal parsejar-ho per poder tractar-ho, per això hi ha una segona funció *then*(), la qual rep l'objecte parsejat i executa el codi necessari per actualitzar la informació del client a partir de la resposta del servidor. Quan s'ha de realitzar una petició diferent de "GET" s'inclou un segon paràmetre a la crida *fetch*() indicant el tipus de petició, així com en cas de que la petició tingui contingut, aleshores també s'hi inclou en aquest paràmetre.

```
fetchNotifications(){
        this.props.cont.checkAuth().then(() => {
            if (this.props.cont.username) {
                fetch(global.serverUrl + "/api/notifications/" +
this.props.cont.username + "/?token=" + this.props.cont.token)
                    .then((response) => {
                        if (response.ok || response.status === 200) {
                            try {
                                return response.json();
                            } catch (error) {
                                return response.text();
                            }
                        }
                        throw new Error("Notifications not available");
                    })
                    .then((data) => {
                        this.setState({
                            notificationsList: data,
                        });
                    })
```

```
                    .catch((error) => console.error(error));
            }
        });
    }
```

Com que és possible que els paràmetres dels components canviïn, aleshores el mètode a continuació s'executa cada cop que canvien els props o l'state del component. A dins d'aquest mètode sol executar-se codi que calgui ser executat cada cop que un paràmetre o un atribut canviï. En el cas següent, cal tornar a fer la petició de les notificacions al servidor si l'usuari canvia.

```
componentDidUpdate(prevProps,prevState){
        if(this.props.cont.username !== prevProps.cont.username){
            this.fetchNotifications();
        }
    }
```

A continuació esmentaré el funcionament de la Page, la qual és l'encarregada de carregar un contingut o un altre en funció de la URL. Per fer-ho utilitzarem el component Switch i Route del mòdul "react-router-dom" que renderitzen el component especificat si el *path* és igual a la URL. El concepte del Consumer serà explicat més endavant.

```
const Page = () => {
    return (
        <div id="page">
            <UsersContext.Consumer>
                {(context) => (
                    <Switch>
                        <Route exact path="/" render={(props) => <Home
{...props} context={context} />} />
                        <Route exact path="/legal" component={Legal} />
                        <Route exact path="/cookies" component={Cookies} />
                        <Route exact path="/privacy" component={Privacy} />
                        <Route exact path="/plan" render={(props) => <Plan
{...props} context={context} />} />
                        <Route exact path="/purchase-list/:id?"
render={(props) => <PurchaseList {...props} context={context} />} />
                        <Route
                            exact
                            path="/login"
                            render={(props) => (
                                <Login {...props}
checkAuth={context.checkAuth} setToken={context.setToken}
setRemember={context.setRemember} />
```

```
                            )}
                        />
                        <Route exact path="/register" component={Register} />
                        <Route
                            exact
                            path="/logout"
                            render={(props) => <Logout {...props}
token={context.token} setToken={context.setToken}
checkAuth={context.checkAuth} />}
                        />
                        <Route path="/category/:id" render={(props) =>
<Category {...props} context={context} />} />
                        <Route path="/recipe/:id" render={(props) => <Recipe
{...props} context={context} />} />
                        <Route path="/user/:id" render={(props) => <UsersPage
{...props} context={context} />} />
                        <Route path="/edit-recipe/:id?" render={(props) =>
<EditRecipe {...props} context={context} />} />
                        <Route component={NotFound} />
                    </Switch>
                )}
            </UsersContext.Consumer>
            ...
        </div>
    );
};
```

La pàgina Home servirà per explicar una mica el funcionament de React en quant a la jerarquia dels components i el renderitzat condicional. Un component pot tenir diversos components fills inclosos dins el mètode render(). Cada fill és renderitzat abans que el pare, i per passar paràmetres als fills, cal especificar-ho com per exemple el component SearchResults té com a paràmetre un atribut de l'state del pare. Els paràmetres passats als fills poden ser des de cadenes de text, números, objectes, arrays o fins i tot funcions. D'aquesta forma permetem que el fill pugui executar una funció del pare quan hi hagi un esdeveniment dins el fill(un click, una tecla apretada,...); però com a contrapartida, si aquesta funció passada utilitza o modifica l'state o els props del pare cal lligar-la al component pare. Per fer-ho s'utilitza la funció bind() dins el constructor() per guardar la referència del pare.

Sobre el renderitzat condicional, dins el mètode render() només es permeten expressions JSX, per tant, els condicionals if-else no funcionen, per contra, es pot utilitzar un operador ternari dins {} per renderitzar un component o un altre en funció d'una condició.

```
constructor(props) {
        super(props);
        this.state = { results: {}, search: false };
        this.saveResults = this.saveResults.bind(this);
    }
 render() {
        return (
            <div className="page-container" id="home-page">
                <BannerHome saveResults={this.saveResults}
context={this.props.context}/>
                {!this.state.search ? <ContentHome /> : <SearchResults
results={this.state.results} />}
            </div>
        );
    }
```

Un aspecte important per renderitzar múltiples components en funció d'una variable, per exemple, els elements d'una llista o una seqüència de valors, s'utilitza la funció Array.map() dins el mètode *render*() o una funció auxiliar que retorni una expressió JSX amb tots els elements que es volen renderitzar. Un exemple és amb les categories de la Home, les quals són dinàmiques i fins a obtenir les dades del servidor no es sap quantes i quines són. És per això que un cop obtinguda la resposta del servidor l'atribut de les categories dins l'state canvia i per tant, es torna a carregar el component.

```
render() {
        return (
            <div className="content-home">
               ...
                    {this.state.categories.map((category,index)=>{
                        return <CategoryContainer key={index} elements={6}
category={category.id} catTitle={category.name}/>;
                    })}
                ...
            </div>
        );
    }
```

Aquesta pàgina és una plataforma multiidioma, per tant, ha sigut necessari implementar certa lògica que explicaré a continuació per poder mostrar la pàgina amb l'idioma escollit.

Primer cal explicar el concepte del React Context, representat a la Figura 67:

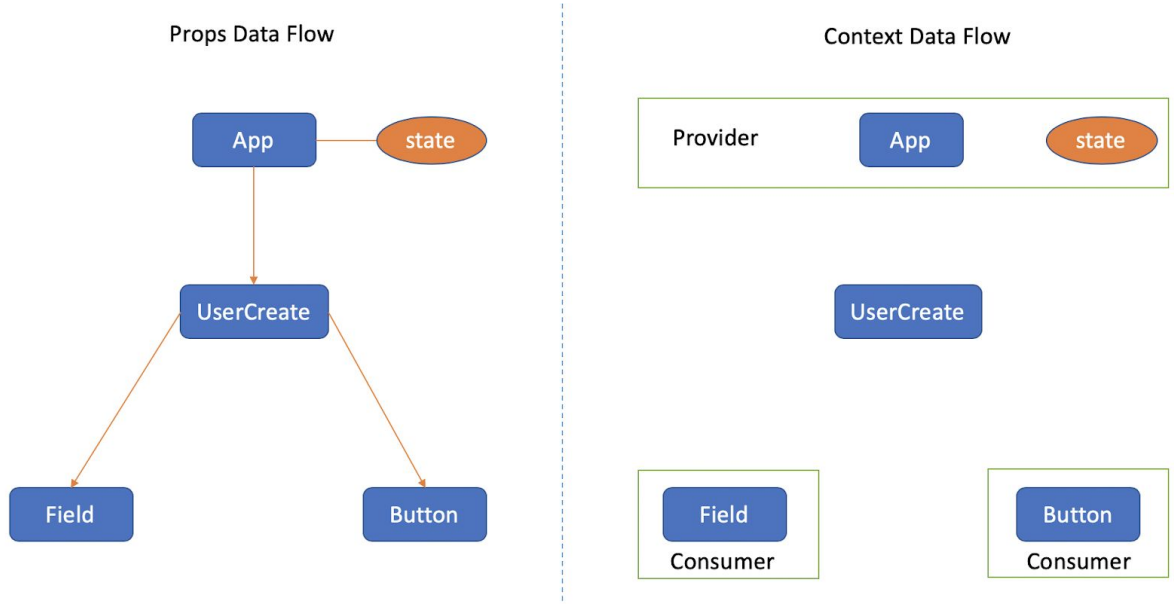


Figura 67- Esquema comparatiu entre els props i el context (https://github.com/GeekEast/react-context)

La idea principal darrere del React Context és proporcionar una forma de poder accedir a variables que no han estat passades com a props pels pares, les quals poden provenir d'un component de la part més alta de l'arbre de components.

Per fer-ho s'utilitza la funció React.*createContext*() que crea un Context nou, al definir un Context també ens cal definir un Provider el qual és l'encarregat de fer accessibles les variables dins el Context.

El Provider és l'encarregat de donar accés a les variables però cal també utilitzar components Consumer per poder obtenir les dades i utilitzar-les.
A continuació es mostra el funcionament per l'idioma, es pot veure que crea el Context i després també hi ha el Provider. Com hem vist al primer fragment de codi, sobre l'estructura de l'App allà s'utilitza el *LanguageProvider*.

```
export const LanguageContext = React.createContext({
    language: currentLang,
    dictionary: dictionaryList[currentLang.value],
    moment: moment
});

export default function LanguageProvider(props) {
    const languageContext = React.useContext(LanguageContext);
    const [language, setLanguage] = React.useState(languageContext.language);
    const [dictionary, setDictionary] = React.useState(
        languageContext.dictionary
```

```
    );
    const [moment, setMoment] = React.useState(
        languageContext.moment
    );

    const provider = {
        language,
        dictionary,
        moment,
        setLanguage: selectedLanguage => {
            ...
        }
    };

    return (
        <LanguageContext.Provider value={provider}>
            {props.children}
        </LanguageContext.Provider>
    );
}
```

I per exemple a Header s'utilitza el Consumer, tot i que en molts components de la web, s'utilitza la propietat de *contextType* dels components per obtenir el contingut del Context de l'idioma poguent-hi accedir mitjançant *this.context* . Per mostrar textos estàtics en funció de l'idioma utilitzaré el component Text que obté el text del diccionari de l'idioma actual i el mostra.

```
function Header() {
    return <UsersContext.Consumer>{(context) => <Navbar cont={context}
/>}</UsersContext.Consumer>;
}
```

```
import { LanguageContext } from "../lib/languageContext";
...

class ContentHome extends React.Component {
    static contextType = LanguageContext;
    constructor(props) {
        ...
    }
}
```

Un cas similar passa amb el context de l'usuari per poder utilitzar les dades de l'usuari que ha entrat per tota la web sense haver d'anar-ho passant com a paràmetre. Aquest context proporciona una funció que comprova si l'usuari té un token vàlid mitjançant una petició al servidor i en cas de que sigui vàlid, l'usuari queda autenticat, al fer login es genera aquest token guardat a la memòria del navegador i eliminat en cas de tancar la sessió de l'usuari.

Abans també ha aparegut el tema de les notificacions i la utilització d'un WebSocket, cal dir que aquest socket al crear-lo utilitza el patró Singleton per garantir la creació d'un únic socket, i que al crear-lo és quan s'obre la connexió amb el servidor. Si l'usuari després fa login, llavors s'envia al servidor que el socket pertany a l'usuari que ha fet login i a partir d'aquell moment ja es pot intercanviar informació de forma bidireccional entre el servidor i el client.

```
class socketController {
    constructor() {
        ...
        this._socket = openSocket(url);
        this._registered = false;
    }
    getSocket(){
        return this._socket;
    }
    registerUser(username){
        if(!this._registered && username){
            this._socket.emit('register-user',{username:username});
            this._registered = true;
        }
    }
}

export class SingletonSocket {
    constructor() {
        if (!SingletonSocket.instance) {
            SingletonSocket.instance = new socketController();
        }
    }

    getInstance() {
        return SingletonSocket.instance;
    }
}
```

Una de les pàgines amb la lògica més complexa és la pàgina Plan, això és degut a que cal utilitzar guardar les planificacions per dies, per tant els índexs són dates com a cadenes de text. Tant per afegir com per eliminar planificacions cal primer identificar la data i després o bé afegir la planificació a la llista del dia, o bé localitzar la planificació i eliminar-la. Com hi han forces textos i components que requereixen mostrar dates o manipular dates s'utilitza la llibreria moment.js per fer aquesta tasca.

És important comentar que tots els objectes del tipus *moment* són mutables i per tant, si no clonéssim els objectes aniríem modificant les dates de tots. Un exemple és l'atribut *currentDate* de l'state el qual clona un objecte moment per no modificar la data del primer. També es pot apreciar que el moment utilitzat prové del Context del llenguatge, ja que al canviar d'idioma també canviem l'idioma de la llibreria *moment* per tal de mostrar els textos de les dates en l'idioma de la web.

```
class PlansContent extends React.Component {
    static contextType = LanguageContext;
    constructor(props, context) {
        super(props, context);
        this.state = {
            today: this.context.moment,
            currentDate: this.context.moment.clone(),
            ...
        };
        this.openAddPlanDialog = this.openAddPlanDialog.bind(this);
        this.closeAddPlanDialog = this.closeAddPlanDialog.bind(this);
        this.closePurchaseListDialog =
this.closePurchaseListDialog.bind(this);
        this.fetchPlanifications = this.fetchPlanifications.bind(this);
        this.selectPlan = this.selectPlan.bind(this);
        this.removePlan = this.removePlan.bind(this);
    }
...
}
```

La funció format() de la llibreria retorna una cadena de text amb la data del moment, amb el format especificat. Per aquesta aplicació, el format per intercanviar dates entre servidor i client és YYYY-MM-DD.

Ja per acabar amb la part de React, a la pàgina EditRecipe, hi ha un component que conté un selector que permet escriure un text i seleccionar l'opció si no existeix cap element que coincideix per crear-lo. Amb uns quants paràmetres podem indicar els textos per defecte, els estils, així com la funció encarregada de detectar algun canvi i executar el codi necessari.

```
<CreatableSelect
                    isClearable
                    onChange={this.handleChange}
                    onInputChange={this.handleInputChange}
                    options={this.state.content}
                    styles={customStyles}
                    placeholder={this.context.dictionary['select']}

formatCreateLabel={(value)=>this.context.dictionary['create']+"
\""+value+"\""}
                    format={(value)=>this.context.dictionary['create']+"
\""+value+"\""}
                    className="selector"
                    components={{ IndicatorSeparator: null, NoOptionsMessage:
() => this.context.dictionary['no-options'] }}
                />
```

La funció és la següent, i detecta si el canvi en el selector implica una selecció d'un element ja existent o la creació d'un nou element, en cada cas, la funció s'encarrega de guardar quin tipus d'acció s'ha realitzat i un cop es cliqui a afegir, s'executi el codi adequat segons l'acció realitzada.

```
handleChange = (newValue, actionMeta) => {
        this.setState({ value: newValue, action: actionMeta.action });
    };
    handleAddIngredient(e) {
        e.preventDefault();
        if (this.state.value) {
            if (this.state.action === "create-option") {
                this.openCreateIngredientDialog();
            } else {
                this.addIngredientToTable();
            }
        }
    }
```

9.2. Backend

Per la part del servidor cal comentar una mica la forma de funcionar del framework Express.js, primer cal inicialitzar el servidor tant per peticions http com https, executant el següent codi tindríem un servidor escoltant peticions tant al port http com al port https.

```
const app = express();
const server = require("http").Server(app);
const https = require("https");
...
var credentials = { key: privateKey, cert: certificate };
...
const PORT = process.env.PORT || 30000;
const PORT_HTTPS = process.env.PORT_HTTPS || 30443;
...
var httpsServer = https.createServer(credentials, app);

...
 serverObj = server.listen(PORT, () => {
                console.log(`Listening to port ${PORT}`);
 });
 serverHttpsObj = httpsServer.listen(PORT_HTTPS, () => {
                console.log(`https: Listening to port ${PORT_HTTPS}`);
 });
```

Per poder enviar els fitxers que pengin els usuaris al servidor cal definir un directori *static*:

```
app.use("/static", express.static(path.resolve(__dirname, "./server/files")));
```

Per poder servir les peticions de la API del servidor, i si la petició no existeix retorna un error:

```
// Mount your existing apiRouter below at the '/api' path.
const apiRouter = require("./server/api");
app.use("/api", apiRouter);

app.get("*", function (req, res) {
    res.status(404).send("Not existing url!");
});
```

El següent mòdul és l'apiRouter, el qual en funció del path de la petició utilitza un subrouter o un altre.

```
const express = require('express');
const apiRouter = express.Router();

const usersRouter = require('./routes/users');
apiRouter.use('/users',usersRouter);
const categoriesRouter = require('./routes/categories');
apiRouter.use('/categories',categoriesRouter);
const ingredientsRouter = require('./routes/ingredients');
apiRouter.use('/ingredients',ingredientsRouter);
const kitchenwareRouter = require('./routes/kitchenware');
apiRouter.use('/kitchenware',kitchenwareRouter);
const unitsRouter = require('./routes/units');
apiRouter.use('/units',unitsRouter);
const recipesRouter = require('./routes/recipes');
apiRouter.use('/recipes',recipesRouter);
const planRouter = require('./routes/plan');
apiRouter.use('/plan',planRouter);
const purchaseListRouter = require('./routes/purchaseList');
apiRouter.use('/purchase-list',purchaseListRouter);
const notificationsRouter = require('./routes/notifications');
apiRouter.use('/notifications',notificationsRouter);
const mediaRouter = require('./routes/media');
apiRouter.use('/media',mediaRouter);
```

A partir d'aquí, cada subrouter processa les peticions definides dins els fragments de codi com el següent, en aquest cas el codi agafa les dades dels usuaris de la base de dades i les envia:

```
usersRouter.get("/", (req, res, next) => {
    dbCont
        .select("user", {}, ["username", "email", "first_name", "last_name",
"profile_picture", "bio", "role"])
        .then((rows) => res.send(rows))
        .catch((err) => console.error(err));
});
```

Com hem vist en l'anterior exemple on agafem les dades de la base de dades per enviar-les al client, veiem que el servidor està connectat a la base de dades i per fer-ho té un mòdul que n'implementa la lògica. Aquest mòdul implementa el patró de disseny Singleton per assegurar que només hi hagi un controlador de la base de dades.

```
class Singleton {
    constructor() {
        if (!Singleton.instance) {
            Singleton.instance = new DBController();
        }
    }

    getInstance() {
        return Singleton.instance;
    }
}
```

A més aquest controlador proporciona una interfície simplificada per fer peticions a la base de dades. Per començar, el controlador crea la connexió amb la base de dades al instanciar-lo, un aspecte important és que abans de penjar al servidor de la web el programa la funció utilitzada era mysql.createConnection(), però provocava que el socket quedés obert si no hi havien peticions i el servidor tancava els sockets connectats que no rebien peticions. Per tal d'evitar aquest error vaig canviar la funció per mysql.*createPool*().

```
class DBController {
    constructor() {
        this.db = mysql.createPool({
            connectionLimit: 10,
            host: config.host,
            user: config.user,
            password: config.password,
            port: config.port,
            database: config.database,
        });
        this.db.getConnection((err, connection) => {
            if (err) {
             ...
            }
            if (connection) connection.release();
            return;
        });
    }
```

El controlador permet definir les taules de la base de dades amb les seves columnes i restriccions i en cas de que no existeixi la taula a la base de dades al iniciar el servidor es creen les taules. Per poder fer això es defineixen totes les taules i el seu contingut en un

fitxer JSON que serà llegit i processat per la funció de createDB(), la qual utilitza la següent notació per parsejar les restriccions:

0:Camp ordinari,
1:Clau primària,
2:Clau forana,
3:Clau primària i forana,
4:Camp únic

```
createDB() {
        return new Promise((resolve, reject) => {
            fs.readFile(baseDir + "/server/db/tables.json", (err, data) => {
                if (err) reject(err);
                const tables = JSON.parse(data);
                const promises = [];
                //TABLES
                for (let key in tables) {
                    promises.push(
                        new Promise((resolve, reject) => {
                            if (tables.hasOwnProperty(key)) {
                                const table = tables[key];
                                const tableName = table["name"];
                                const tableFields = table["fields"];
                                // FIELD FOR THE TABLE
                                let primaries = [];
                                let foreigns = [];
                                let createFields = " ( ";
                                for (let keyField in tableFields) {
                                    if (tableFields.hasOwnProperty(keyField))
{
                                        const field = tableFields[keyField];
                                        // NAME
                                        createFields += "`" +
field["field_name"] + "` ";
                                        //TYPE
                                        createFields +=
field["field_type"]["type"].toUpperCase();
                                        if (field["field_type"]["type"] ==
"varchar") {
                                            if
(field["field_type"].hasOwnProperty("length")) {
```

```
                                                createFields += "(" +
field["field_type"]["length"] + ")";
                                            } else {
                                                createFields += "(150)";
                                            }
                                        } else if (field["field_type"]["type"]
== "int") {
                                            if
(field["field_type"].hasOwnProperty("params")) {
                                                createFields += " " +
field["field_type"]["params"].toUpperCase() + " UNIQUE";
                                            }
                                        } else if (field["field_type"]["type"]
== "float") {
                                            if
(field["field_type"].hasOwnProperty("params")) {
                                                createFields += "(" +
field["field_type"]["params"][0] + "," + field["field_type"]["params"][1] +
")";
                                            }
                                        }
                                        createFields += " NOT NULL ";
                                        if (field["field_type"]["type"] ==
"varchar") {
                                            createFields += "DEFAULT ''";
                                        } else if (field["field_type"]["type"]
== "int" || field["field_type"]["type"] == "boolean") {
                                            if
(!field["field_type"].hasOwnProperty("params")) {
                                                createFields += "DEFAULT 0";
                                            }
                                        } else if (field["field_type"]["type"]
== "date" || field["field_type"]["type"] == "datetime") {
                                            createFields += "DEFAULT NOW()";
                                        } else if (field["field_type"]["type"]
== "float") {
                                            createFields += "DEFAULT 0.0";
                                        }
                                        if
(field.hasOwnProperty("field_constraint")) {
```

```
                                        const fieldConstraint =
field["field_constraint"];
                                        if (fieldConstraint == 1) {

primaries.push(field["field_name"]);
                                        } else if (fieldConstraint == 2) {
                                            foreigns.push({
                                                field_name:
field["field_name"],
                                                relation_table:
field["field_relation"]["table"],
                                                relation_field:
field["field_relation"]["field"],
                                            });
                                        } else if (fieldConstraint == 3) {

primaries.push(field["field_name"]);
                                            foreigns.push({
                                                field_name:
field["field_name"],
                                                relation_table:
field["field_relation"]["table"],
                                                relation_field:
field["field_relation"]["field"],
                                            });
                                        } else if (fieldConstraint == 4) {
                                            createFields += " UNIQUE ";
                                        }
                                    }
                                    createFields += ",";
                                }
                            }
                            //CONSTRAINTS
                            let constraints = "PRIMARY KEY (";
                            primaries.forEach((element) => {
                                constraints += element + ",";
                            });
                            constraints = constraints.substring(0,
constraints.length - 1); //Remove last separator
                            constraints += ")";
```

```
                                if (foreigns.length > 0) {
                                    constraints += ", ";
                                    foreigns.forEach((element) => {
                                        constraints +=
                                            "FOREIGN KEY (" +
                                            element["field_name"] +
                                            ") REFERENCES " +
                                            element["relation_table"] +
                                            "(" +
                                            element["relation_field"] +
                                            "), ";
                                    });
                                    constraints = constraints.substring(0,
constraints.length - 2);
                                }
                                // CREATE TABLE
                                this.db.query(
                                    "CREATE TABLE IF NOT EXISTS " + tableName
+ createFields + constraints + ")ENGINE=InnoDB DEFAULT CHARSET=utf8 ;",
                                    function (error, results, fields) {
                                        if (error) reject(error);
                                        resolve(results);
                                    }
                                );
                            }
                        })
                    );
                }
                Promise.all(promises)
                    .then(() => resolve())
                    .catch((err) => reject(err));
            });
        });
    }
```

Un cop creada la base de dades, les consultes que rebrà la base de dades seran SELECT, INSERT, UPDATE, DELETE, i per simplificar una mica la generació de consultes a la base de dades he creat 4 mètodes per encapsular la complexitat del processament de les peticions, per exemple una petició SELECT utilitzaria el següent mètode, com es pot veure a partir dels paràmetres, aquests es transformen a cadenes de text per poder afegir a les consultes.:

```
select(table, where = {}, columns = "*", limit = "", offset = "", order = {})
{
        const columnsString = Array.isArray(columns) ? columns.join(",") :
columns;
        const values = [];
        for (let key in where) {
            values.push(where[key]["field"]);
        }
        const whereString = processWhere(table, where);
        const orderString = processOrder(table, order);
        const limitString = limit ? " LIMIT " + limit : "";
        const offsetString = offset ? " OFFSET " + offset : "";
        return new Promise((resolve, reject) => {
            this.db.query(
                "SELECT " + columnsString + " FROM " + table + whereString +
orderString + limitString + offsetString + ";",
                values,
                (err, results, fields) => {
                    if (err) {
                        reject(err);
                    }
                    resolve(results);
                }
            );
        });
    }
```

Cridant aquesta funció d'aquesta forma, fet que estalvia escriure paraules com "SELECT" o "FROM":

```
dbCont.select("recipe", where, ["recipe_id", "title", "thumbnail", "time"],
limit, offset, { publication_date: { direction: "DESC" } })
            .then((recipes) => {
                res.send(recipes);
            })
            .catch((err) => res.status(500).send(err));
```

Tot i que aquestes funcions són vàlides en molts casos, a l'hora de fer consultes amb JOINS o alguna consulta amb condicions complexes pot ser que no ens serveixin aquests

mètodes i per això també vaig crear el mètode següent que permet escriure una query SQL per poder-la executar directament.

```
rawQuery(query, values) {
        return new Promise((resolve, reject) => {
            if (values) {
                this.db.query(query, values, (err, results, fields) => {
                    if (err) reject(err);
                    resolve(results);
                });
            } else {
                this.db.query(query, (err, results, fields) => {
                    if (err) reject(err);
                    resolve(results);
                });
            }
        });
    }
```

La resta de mètodes segueixen un procés similar al *select()*.

Un aspecte a considerar del servidor ha estat que calia que permetés pujar imatges i les guardés. Per fer-ho el mediaRouter s'encarrega de les peticions relacionades amb fitxers, el qual al fer una petició post amb la URL '/media' espera un fitxer, que guarda a un directori temporal i retorna com a resposta la ruta del fitxer penjat. El mòdul encarregat de gestionar que es pengin fitxers s'anomena "multer", i només especificant la ruta a guardar els fitxers i a quines peticions s'esperen fitxers i quants ja es guarden automàticament.

```
const upload = multer({ dest: filesTempPath });

mediaRouter.post("/", upload.single('media'), (req, res, next) => {
    const ext =
req.file.originalname.substring(req.file.originalname.lastIndexOf(".") + 1,
req.file.originalname.length) || req.file.originalname;
    const oldpath = req.file.path;
    const newpath = oldpath + "." + ext;
    fs.rename(oldpath, newpath, function (err) {
        if (err) res.status(500).send();
        let pathRelative = path.relative(filesPath, newpath);
        pathRelative = pathRelative.replace("\\", "/");
        res.send({ media: pathRelative });
    });
});
```

Com he esmentat anteriorment al penjar un fitxer es guarda en una carpeta temporal, per exemple al guardar una recepta es mou aquest fitxer en una carpeta on es guarden els fitxers definitius.

Una de les peticions més complexes a servir és la de planificacions, ja que cal fer diverses consultes a la base de dades. Per passos:

- Comprovar autenticació usuari
- Processar les dates inicials i finals i buscar totes les dates creades entre les dos dates
- Obtenir totes les planificacions de l'usuari que estiguin dins l'interval
- Per cada planificació obtenir informació de la recepta planficada
- Ordenar els resultats per hores inicials.

```
planRouter.get("/", (req, res, next) => {
    if (req.query.user && req.query.token) {
        lib.checkUserCredentials(req.query.user, req.query.token)
            .then((isAuthenticated) => {
                if (isAuthenticated) {
                    let plannings = {};
                    let intBegin = moment(req.query.dateI).clone();
                    const intEnd = moment(req.query.dateF).clone();
                    while (intEnd.isAfter(intBegin)) {
                        plannings[intBegin.format("YYYY-MM-DD")] = [];
                        intBegin.add(1, "days");
                    }
                    dbCont
                        .rawQuery("SELECT * FROM date WHERE timestamp >= ? AND
timestamp < ? ORDER BY timestamp", [req.query.dateI, req.query.dateF])
                        .then((dates) => {
                            const dateMap = {};
                            dates.forEach((date) => {
                                dateMap[date.date_id] = date.timestamp;
                            });
                            const dateIds = dates.map((val) => val.date_id);
                            if (dateIds.length > 0) {
                                dbCont
                                    .rawQuery("SELECT * FROM recipes_plans
WHERE username = ? AND(date_ini IN (?) OR date_final IN (?))", [
                                        req.query.user,
                                        dateIds,
```

```
                                    dateIds,
                                ])
                                .then((results) => {
                                    const promises = [];
                                    results.forEach((result) => {
                                        promises.push(
                                            new Promise((resolve, reject)
=> {
                                                const date_ini =
moment(dateMap[result.date_ini]).clone();
                                                const date_fi =
moment(dateMap[result.date_final]).clone();
                                                const dateString =
date_ini.format("YYYY-MM-DD");
                                                dbCont
                                                    .select("recipe", {
recipe_id: { field: result.recipe_id } }, [
                                                        "title",
                                                        "author",
                                                        "visibility",
                                                    ])
                                                    .then((recipe) => {
                                                        if (recipe.length
> 0) {
                                                            recipe =
recipe[0];
                                                            if
(recipe.visibility === 1 || recipe.author === req.query.user) {
                                                                const
newPlan = {
recipe_id: result.recipe_id,
portions: result.portions,
list:result.list,
title: recipe.title,
date_ini: date_ini.format("YYYY-MM-DD HH:mm:ss"),
```

```
date_final: date_fi.format("YYYY-MM-DD HH:mm:ss"),
                                                                    };

plannings[dateString].push(newPlan);
                                                                }
                                                                resolve();
                                                            } else {
                                                                reject();
                                                            }
                                                        })
                                                        .catch((err) =>
reject(err));
                                                })
                                        );
                                });
                                Promise.all(promises)
                                        .then(() => {
                                                for (let key in plannings) {
                                                        const day =
plannings[key];

                                                        day.sort((a, b) => {
                                                                const date1 =
moment(a.date_ini).clone();

                                                                const date2 =
moment(b.date_ini).clone();

                                                                return
date1.isBefore(date2) ? -1 : date1.isSame(date2) ? 0 : 1;
                                                        });
                                                }
                                                res.send(plannings);
                                        })
                                        .catch((err) =>
res.status(500).send(err));
                                })
                                .catch((err) => console.error(err));
                        } else {
                                res.send(plannings);
                        }
                });
```

```
                } else {
                    res.status(401).send("Failed to authenticate user");
                }
            })
            .catch((err) => res.status(500).send("Error checking user's
authentication"));
    } else {
        res.status(401).send("Missing user parameters");
    }
});
```

Un aspecte important que he utilitzat força ha estat la funció Promise.all(), la qual serveix per esperar que totes les Promises acabin abans d'executar el codi del la funció then(), això ha estat força útil per no aniuar diverses consultes a la base de dades, així com també a l'hora de fer consultes per cada element d'un Array;en el codi anterior n'hi ha un exemple.

Per últim, cal comentar que el servidor també té certa lògica respecte els WebSockets, on es defineixen les peticions que arribaran pel socket i la lògica corresponent.

```
var io = app.get("socketio");
io.on("connection", (socket) => {
    socket.on("disconnect", () => {
        const sessionID = socket.id;
        dbCont
            .delete("users_sockets", {socket_id:{field:sessionID}})
            .then((rows) => {})
            .catch((err) => console.error(err));
    });
    socket.on("register-user", function (user) {
        const sessionID = socket.id;
        dbCont
                        .insert("users_sockets",  ["username",  "socket_id"],
[user.username, sessionID])
            .then((rows) => {})
            .catch((err) => console.error(err));
    });
});
```

9.3. Connexió entre frontend i backend

La connexió entre les dos parts es fa mitjançant peticions HTTP que són generades per les crides fetch() les quals implementen crides XHR que permeten interactuar amb el servidor sense haver de recarregar la pàgina, i servides a través dels routers d'Express.

Per la connexió dels WebSockets s'utilitza la llibreria Socket.io que permet la implementació tant de la part del client com de la part del servidor. El servidor manté una taula relacionant els usuaris amb els sockets, sabent així a quin socket enviar informació i a quin no.

9.4. Testeig de l'aplicació

La majoria dels testos realitzats per la part de frontend han estat de forma manual interactuant amb les interfícies i comprovant que l'acció resultant era l'esperada. La majoria de testos realitzats han sigut clics a butons, la introducció del text en un camp… per exemple a la pàgina d'edició de receptes, els passos es van creant a mesura que es necessiten mitjançant el botó d'afegir, per tant, per comprovar que s'afegeixen passos, el test a realitzar ha estat fer clic al botó i comprovar que s'ha afegit un nou pas a la llista. Un altre exemple és amb els desplegables, per provar el correcte funcionament, el test a realitzar ha estat clicar sobre el botó del desplegable per obrir-lo, i un cop obert clicar en una regió fora del desplegable esperant que s'amagués.

Per la part de backend com que la majoria de funcions són rutes d'una api la forma de provar-ho ha estat mitjançant un client (Postman) que fes peticions contra el servidor i proporcionant uns paràmetres, aleshores el servidor retorna una resposta al client que es compara amb la resposta esperada.

La majoria de testos han estat per provar el mòdul de connexió de la base de dades ja que d'aquest mòdul en depenien la resta i han estat utilitzant el programa Postman que permet fer peticions al servidor. L'objectiu dels testos d'aquest mòdul era assegurar que el mòdul permet fer les consultes a la base de dades correctament a partir dels paràmetres esperats.

La resta de mòduls s'han provat utilitzant també Postman així com també la pròpia web com a client. Alguns dels exemples de peticions per testejar aquest mòdul han estat:

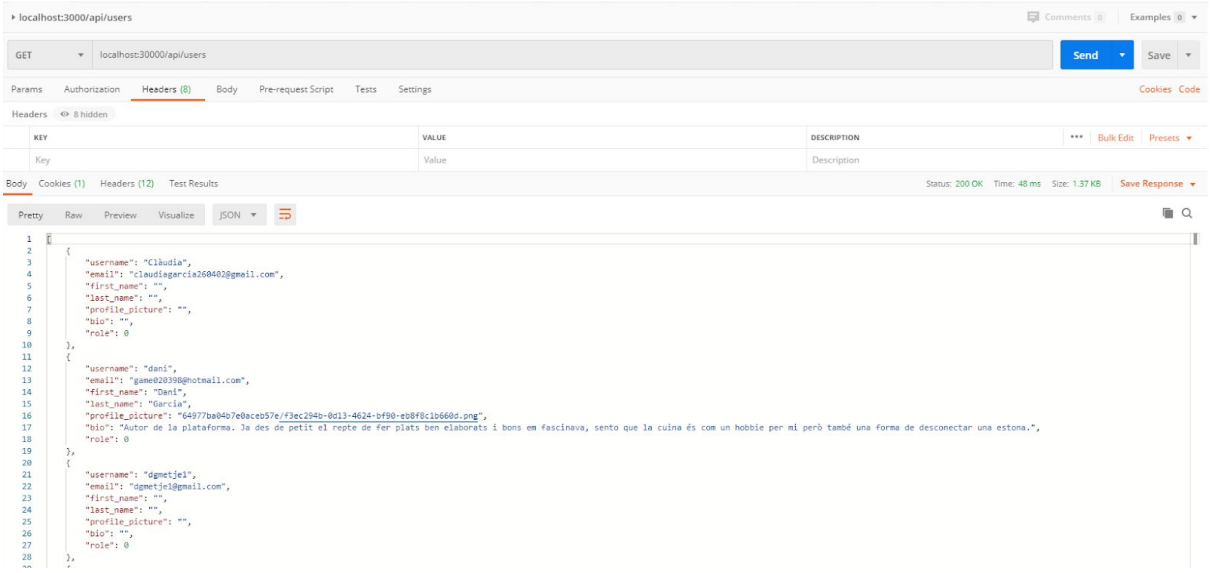


Figura 68- Captura de pantalla d'una petició amb Postman

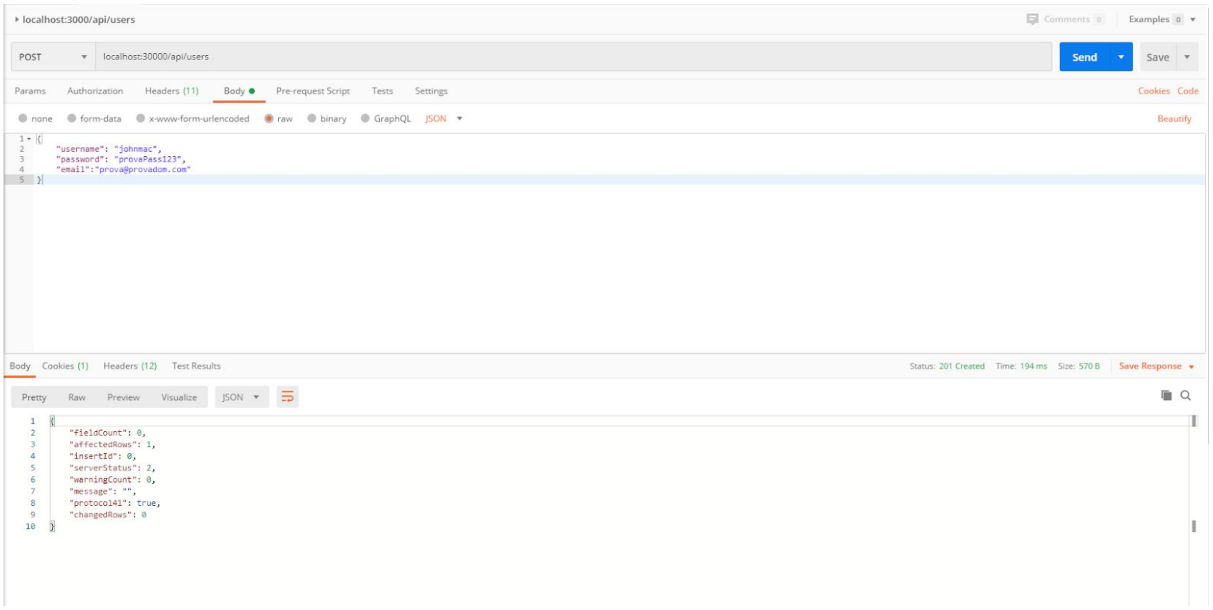


Figura 69- Captura de pantalla d'una petició amb Postman

Figura 70- Captura de pantalla d'una petició amb Postman

Per comprovar que la connexió entre les dues parts, és a dir, si s'enviava correctament la petició i la resposta tenia el format esperat vaig utilitzar les Chrome DevTools on es poden veure totes les peticions que es fan al servidor a l'apartat "Network". Aquesta eina ha servit

bastant per debugar les peticions i respostes entre les dues parts ja que en un sol lloc es pot consultar les dues parts. Un exemple és la petició de la informació d'una recepta i la resposta del servidor, tal i com es mostra a la següent figura:

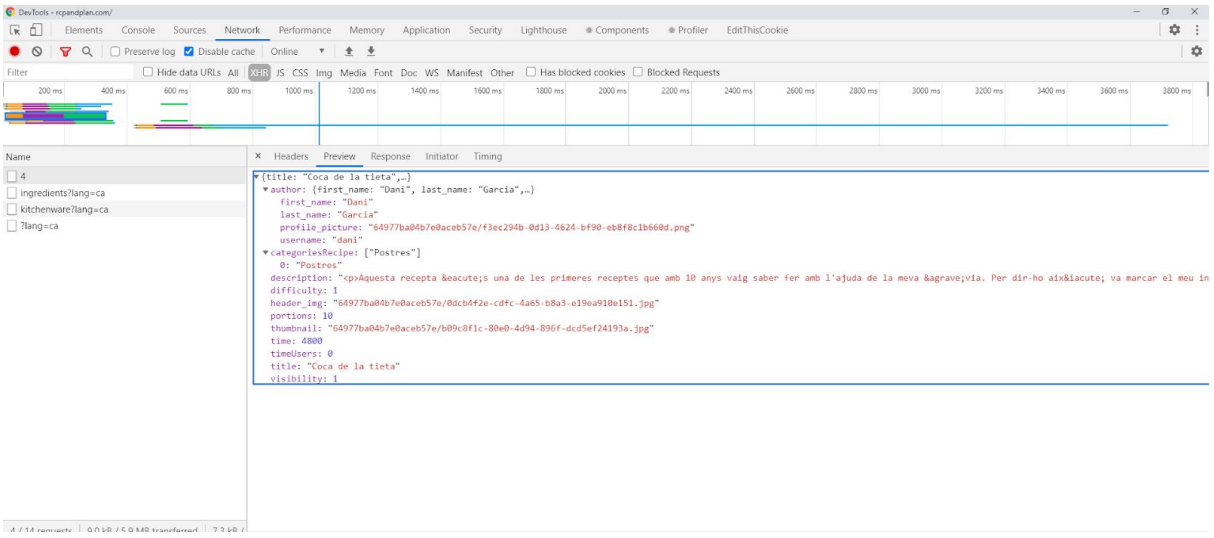


Figura 71- Captura de pantalla d'una petició al servidor registrada amb les Chrome DevTools

10. Implantació i resultats

Executant l'aplicació des del meu ordinador era possible accedir-hi a través de la ip, ja sigui de forma local a través de la ip privada del meu ordinador, especificant el port de l'aplicació a la URL.

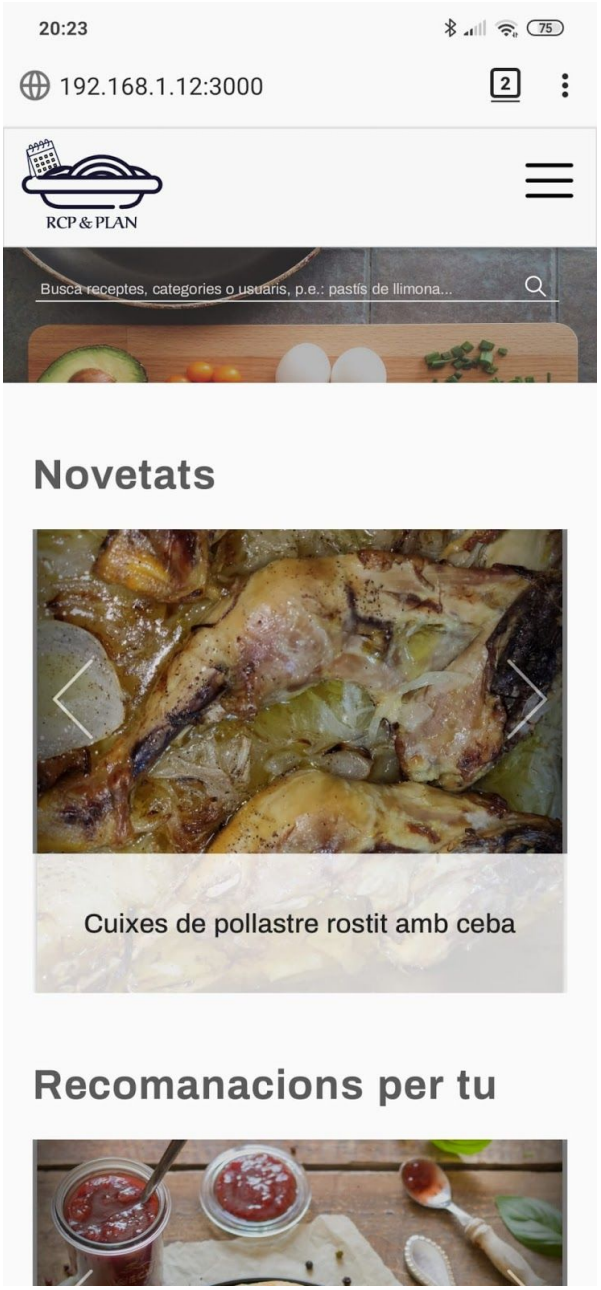


Figura 72- Captura de pantalla de la web accedint des del mòbil

I per fer-ho accessible des de fora la meva xarxa local, vaig haver d'obrir un port del router mitjançant la pàgina de configuració del router. Concretament, vaig fer que el port 80 del router redirigís al port 3000 del meu ordinador, fent accessible la web des de fora la xarxa a través de la ip pública del meu router.

Però per poder accedir a la web calia que el meu ordinador estigués encès i tingués els processos del client i servidor engegats, per tant, vaig decidir adquirir un domini i un servei de hosting web per tal de fer accessible la web sense necessitat de conèixer la meva direcció IP, i que estigués funcionant les 24 hores.

Desplegament de la web a un servidor

Tant el servei de hosting com el domini han estat adquirits al proveïdor A2 Hosting. Un cop comprats aquests serveis, vaig poder accedir al panell d'administració del servidor cPanel.

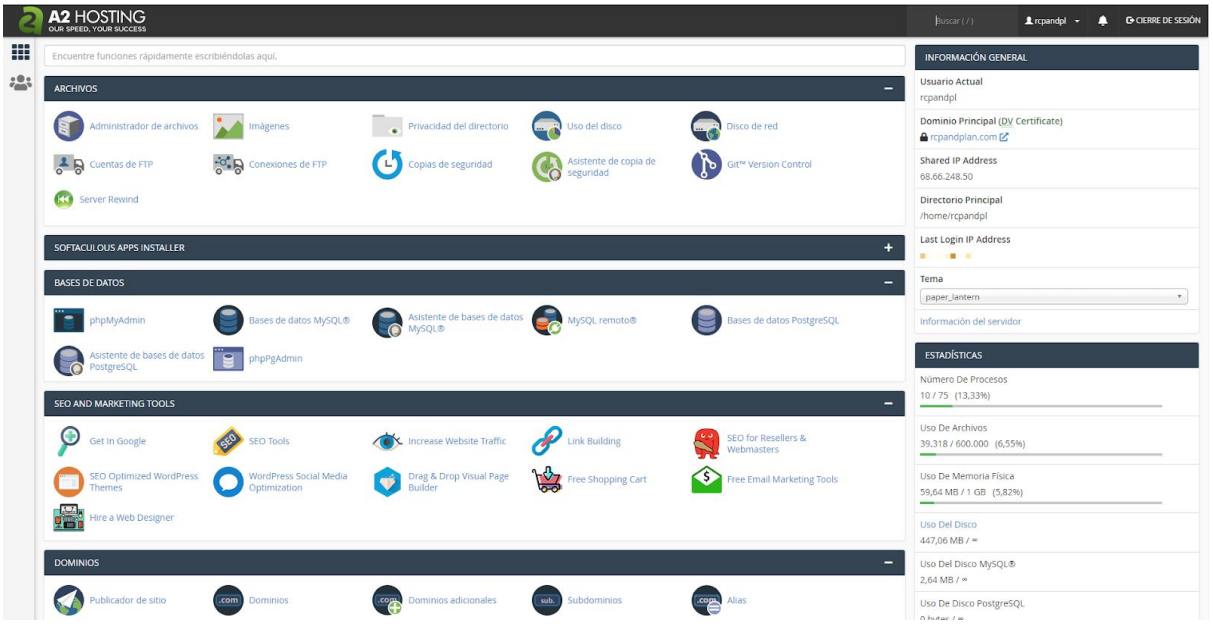


Figura 73- Captura de pantalla del panell de control del servidor cPanel

El primer pas va ser obtenir un usuari FTP per poder accedir mitjançant el programa Filezilla per poder pujar i descarregar els fitxers del servidor.

Figura 74- Captura de pantalla de l'apartat de creació de contes FTP del cPanel

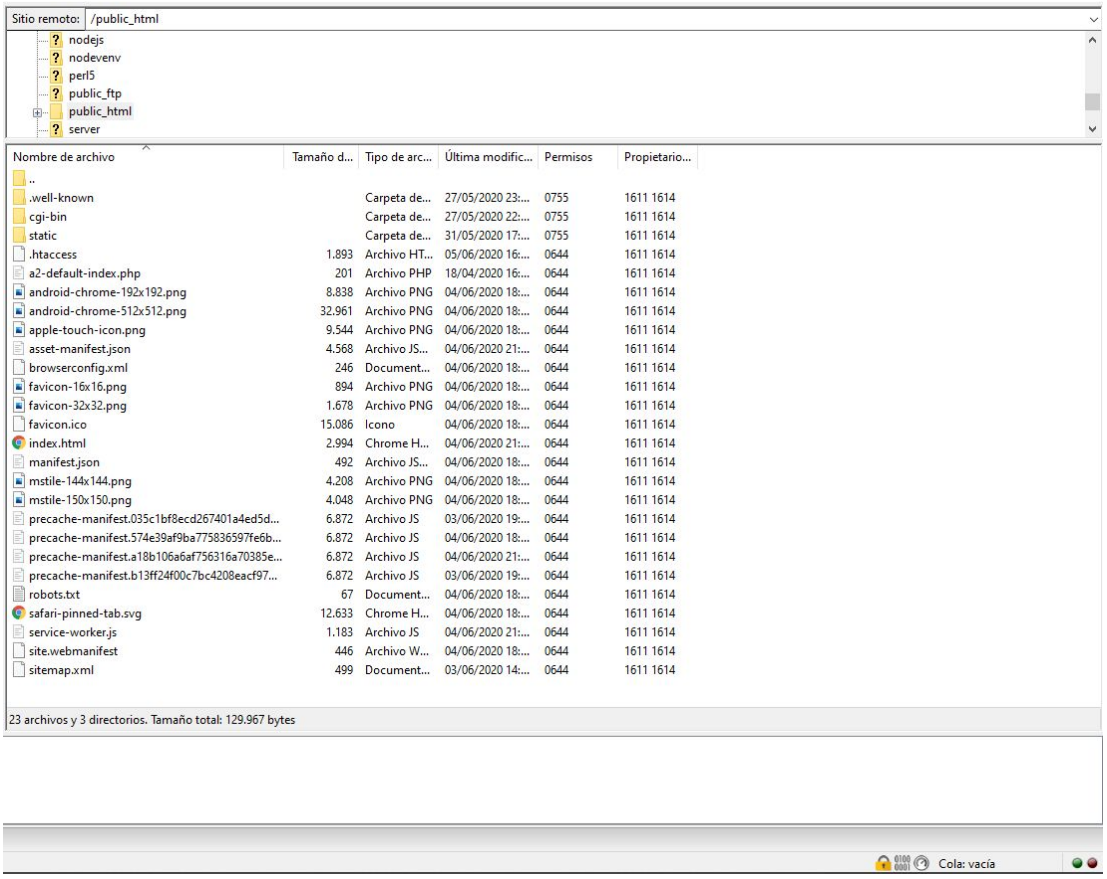


Figura 75- Directori del servidor accedint a través del Filezilla

Un cop fet això també necessitava accés a la consola, per fer-ho vaig generar les claus SSH corresponents per poder accedir-hi mitjançant el programa Putty.

Figura 76- Directori del servidor accedint a través del Filezilla

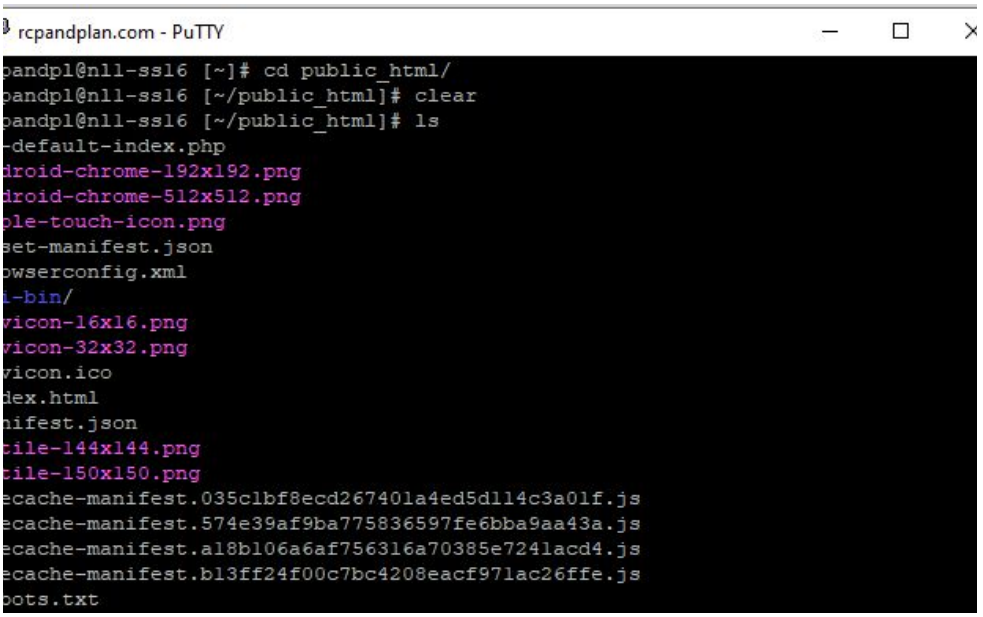


Figura 77- Accés al shell del servidor mitjançant Putty

Un cop dins el servidor vaig procedir a instal·lar el Node.js seguint la documentació proporcionada per el proveïdor de hosting, seguint els passos de la següent pàgina: https://www.a2hosting.com/kb/installable-applications/manual-installations/installing-node-js-on-managed-hosting-accounts

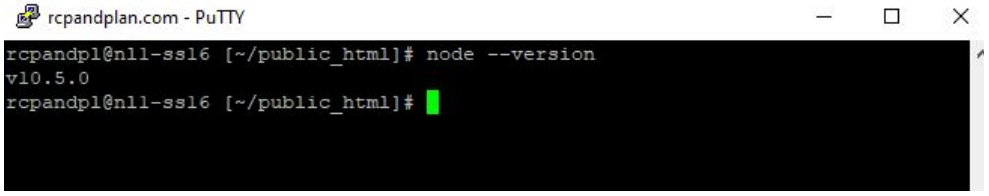


Figura 78- Comprovació d'instal·lació de Node.js

Un cop instal·lat Node.js, ja podia afegir el meu codi al servidor per poder-lo executar, però abans calia migrar la base de dades que tenia en local al servidor, per fer-ho vaig utilitzar el programa MySQL WorkBench per fer una exportació de les dades, i a través del phpMyAdmin de cPanel vaig importar-les al servidor. Per instal·lar el codi del servidor vaig utilitzar el link del repositori del codi del servidor per descarregar-lo a través de la consola.
D'altra banda, el codi del client, el vaig instal·lar executant en local la comanda per generar la versió de producció del codi, la qual genera una carpeta build amb tot el contingut que vaig pujar dins la carpeta de public_html a través del Filezilla.

Ara la web ja era accessible però el servidor encara no funcionava, primer va caldre configurar les dades del servidor perquè es connectés a la base de dades del servidor i utilitzés el servidor de correu electrònic del servidor. Un cop fet, per engegar el servidor vaig executar la següent comanda per executar el servidor en segon pla:

```
nohup node app.js &
```

En aquest punt la plataforma ja funcionava correctament però el protocol de connexió de la web i del servidor eren http, per tant per fer-ho més segur va caldre activar el protocol https,

instal·lar els certificats pertinents i activar la redirecció https per tal que totes les peticions http fossin redirigides al protocol https.

Figura 79- Comprovació d'instal·lació de Node.js

Un problema al fer la migració al servidor és que el procés servidor que serveix el contingut no és el servidor executat per servir les pàgines de React si no que és Apache, i al introduir una URL diferent de l'arrel de la pàgina donava error perquè no hi havia cap fitxer per redirigir. Davant aquest problema vaig decidir redirigir a través del fitxer .htaccess tota URL diferent de la de l'arrel que no fos cap al servidor(filtrat per port) cap a la pàgina inicial. Això va provocar que el canvi d'idioma per exemple et redirigeix a la pàgina inicial i vaig haver de canviar algunes parts del codi per tal de fer una petició al servidor per actualitzar la informació enlloc de tornar a carregar la pàgina.

Un cop desplegada la web al servidor al ser pública per tota la xarxa i tenir un domini propi, els usuaris poden accedir a la web mitjançant la url, però si busquen en un motor de cerca, com per exemple google, no apareixeria la web com a resultat. Per fer que aparegui, vaig haver de generar el fitxer sitemap.xml, el qual permet als robots de Google obtenir l'estructura de la web.

Un cop generat i posat dins el directori de la web, mitjançant la plataforma de Google per indexar pàgines web, vaig sol·licitar la indexació. Els resultats obtinguts fins ara han estat alguna impressió i inclús algun accés a la pàgina web a partir de google. Com més impressions i cerques sobre el contingut de la web, la posició de la web a l'hora de cercar conceptes claus com receptes o planificacions pot ser cada vegada més propera a l'inici. La plataforma es mostra a la Figura a continuació:

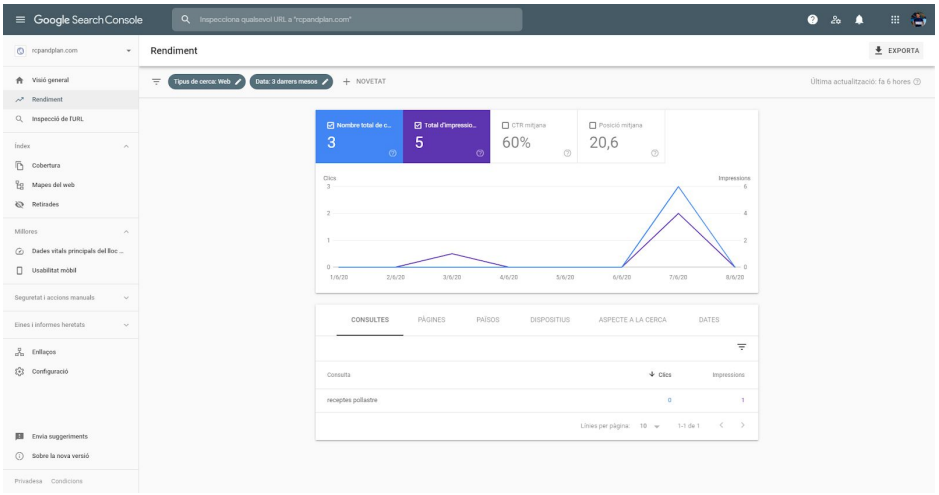


Figura 80- Google Search Console (administració de la web a Google)

Resultats

En aquest apartat es detallen els resultats obtinguts un cop finalitzat el desenvolupament del projecte. El projecte té per objectiu crear una solució al problema plantejat anteriorment a partir de:

- La creació d'una plataforma que permeti als usuaris compartir receptes(publicar les seves pròpies, consultar les receptes d'altres usuaris, valorar-les).
- Afegir a la plataforma la funcionalitat de planificació de receptes en un període per organitzar l'alimentació de l'usuari.
- Afegir la possibilitat d'obtenir una llista de tots els ingredients necessaris per dur a terme la planificació prevista.
- Al ser una plataforma on qualsevol usuari pot penjar-hi contingut, afegir la capacitat de moderar el contingut dels usuaris per part d'alguns usuaris específics.

El primer punt s'ha assolit completament, i a continuació es mostren alguns exemples de les funcionalitats possibles de l'aplicació,ordenant les funcionalitats pel rol de l'usuari, per tant primer el que pot fer un usuari no registrat és:

Primer de tot, començant per la pàgina principal, tal i com es mostra a la Figura 81 es mostra la pàgina principal a través de la qual els usuaris poden veure les receptes que hi ha a la plataforma. Els usuaris tenen l'opció d'utilitzar el buscador per buscar receptes, categories o usuaris escrivint el seu nom.

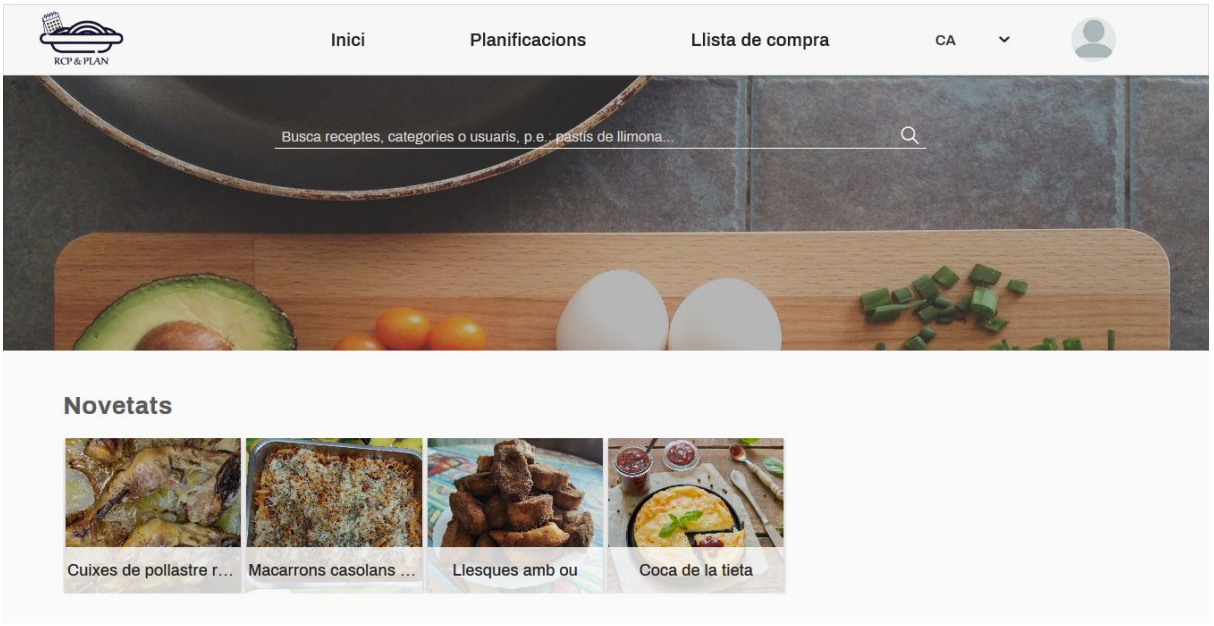


Figura 81- Pàgina principal de la web

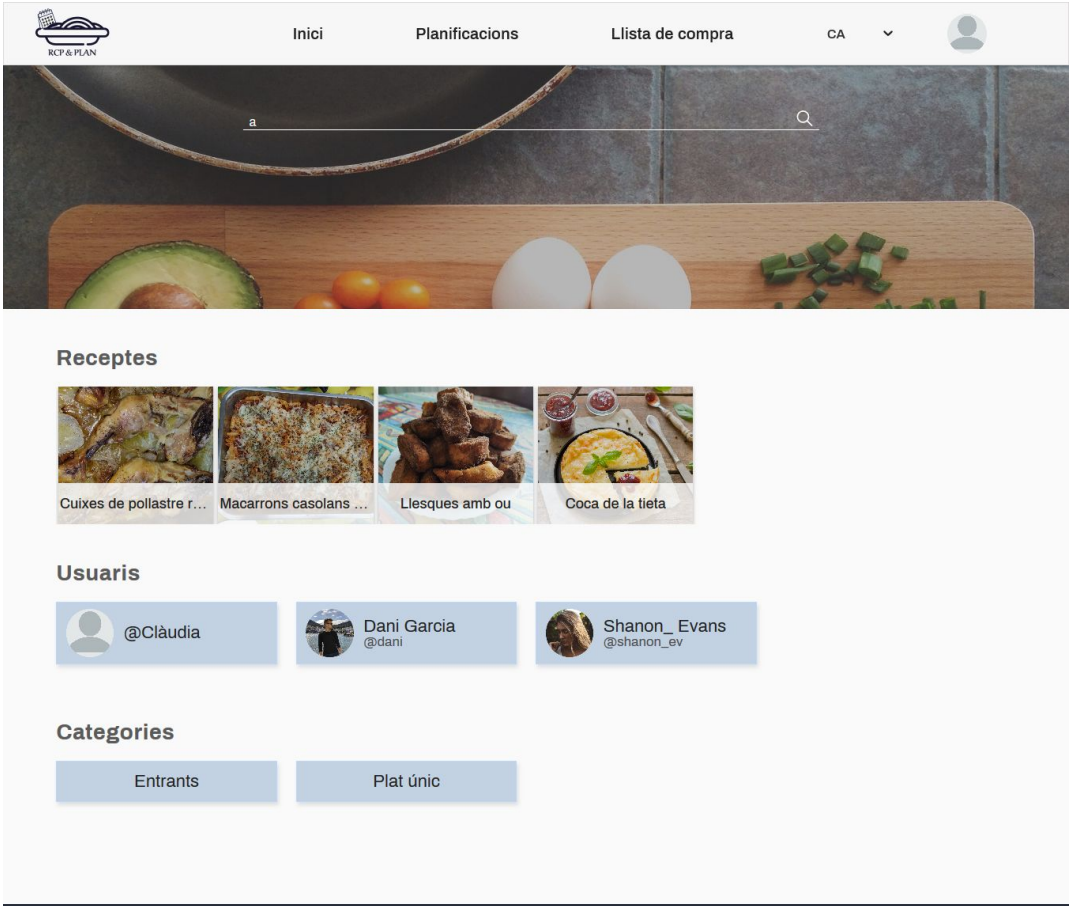


Figura 82- Pàgina principal de la web cercador

Al fer clic a en una targeta d'una recepta, es mostra la fitxa detallada de la recepta, la qual conté tota la informació de la recepta, des dels ingredients, fins als passos a seguir. Hi ha una secció on hi ha la informació general de la recepta, per exemple l'autor, el temps necessari per fer la recepta, les porcions que poden ajustar-se a les necessitats de cada usuari, recalculant així les quantitats d'ingredients necessàries.

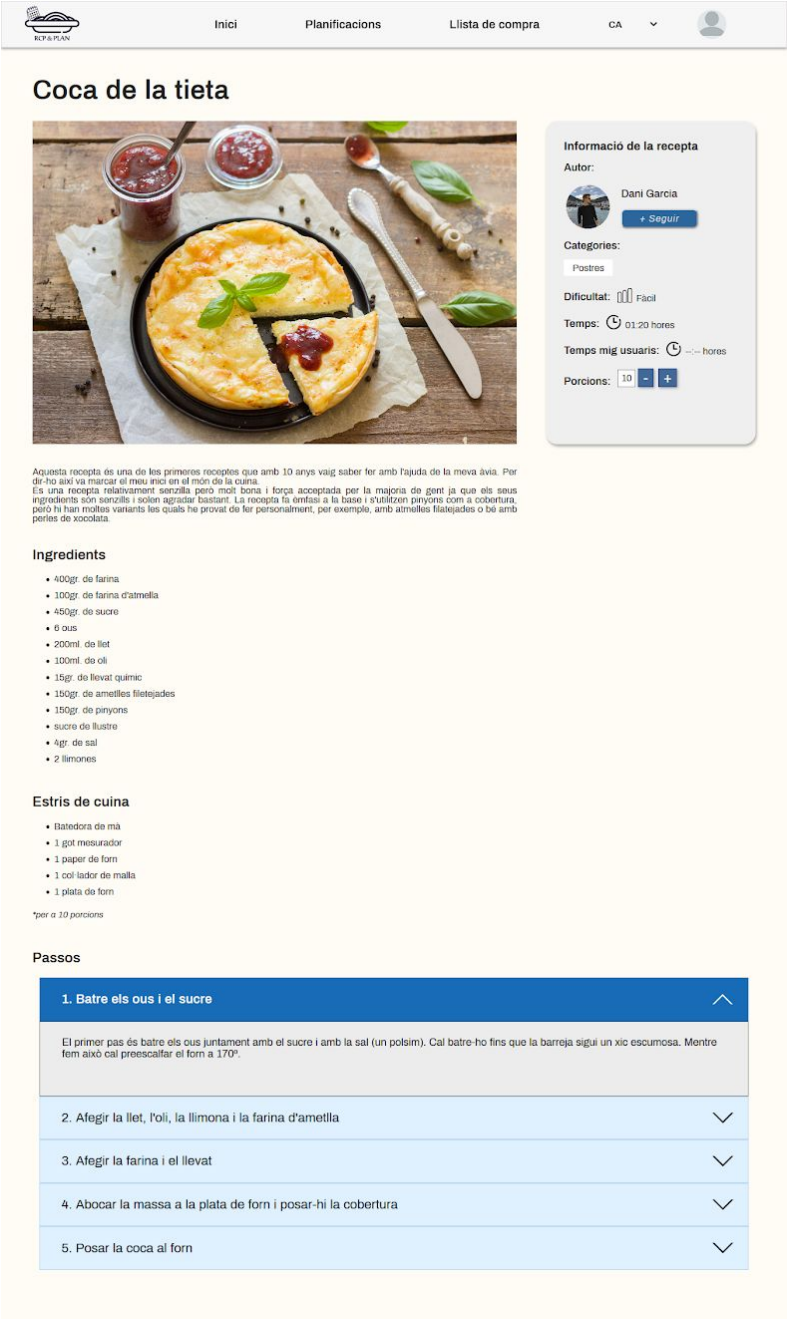


Figura 83- Pàgina de la fitxa d'una recepta

Al fer clic sobre l'autor de la recepta, o bé al cercar un usuari i fer clic sobre la targeta de l'usuari cercat, es mostrarà la pàgina del perfil de l'usuari, la qual mostra la informació personal de l'usuari i les seves receptes. La Figura 84 correspon a la pàgina d'un usuari de la plataforma. Com que els usuaris registrats poden afegir a altres usuaris registrats com a contactes, en aquesta pàgina hi ha un botó a sota el nom de l'usuari que permet a un usuari seguir-ne a un altre per així rebre notificacions de noves receptes que pengi l'usuari. En cas de ja seguir-lo el botó permet eliminar a l'usuari com a contacte, i si l'usuari no està registrat, aleshores el botó redirigeix a la pàgina d'inici de sessió.

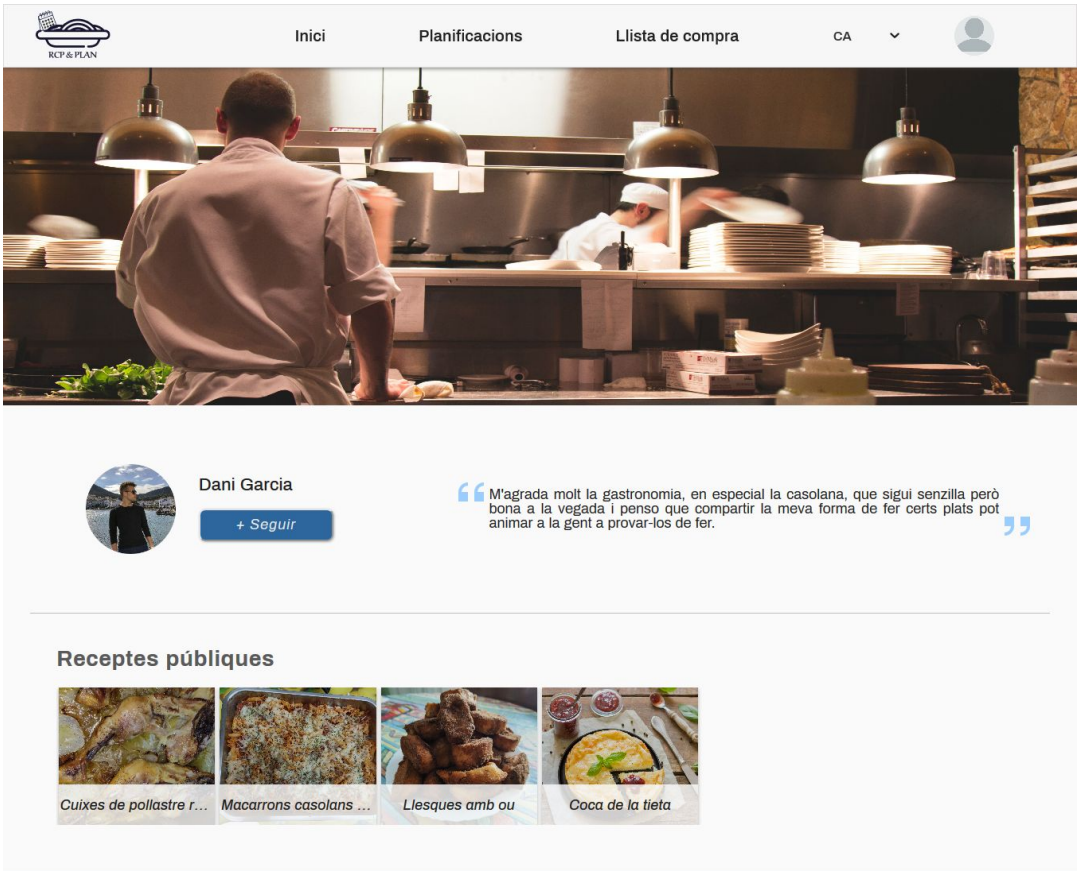


Figura 84- Pàgina de la fitxa d'una recepta

La pàgina conté també seccions que es poden consultar sobre els termes legals de la web, la protecció de les dades personals i la política de galetes utilitzada per la pàgina. Un exemple n'és la Figura 85.

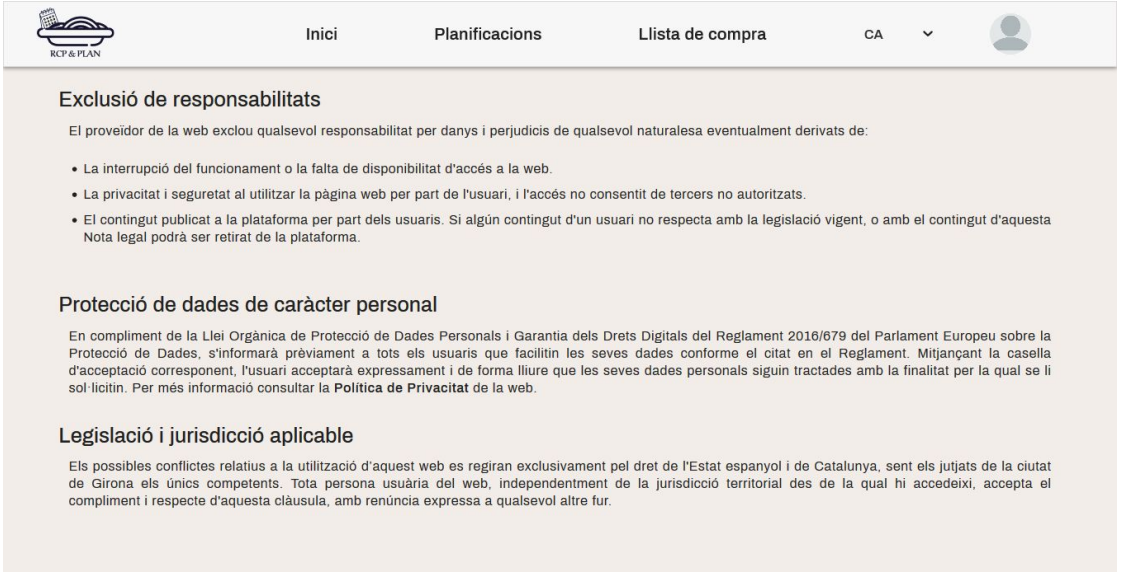


Figura 85- Pàgina dels temes legals de la web

Com a usuari no registrat, pots consultar les receptes d'altres usuaris però en canvi no pots publicar cap recepta a la pàgina ni pots fer moltes altres coses, com valorar les receptes,

rebre notificacions, crear planificacions i generar la llista de la compra d'aquestes planificacions. Per fer-ho cal donar-se d'alta al sistema a través de la pàgina de registre mostrada a la Figura 86.

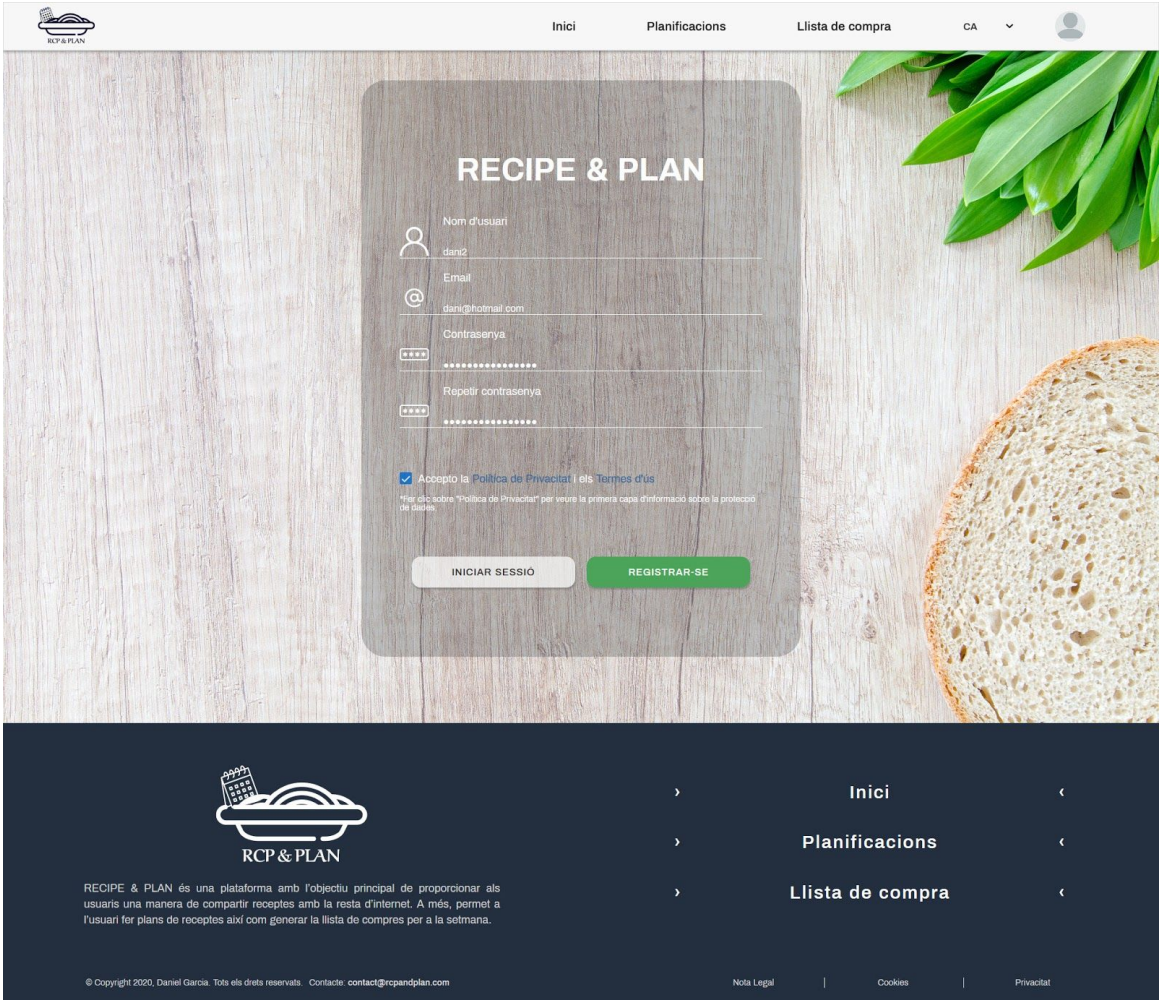


Figura 86- Pàgina de registre per a nous usuaris

Un cop enviat el formulari de registre l'aplicació enviarà al mail de l'usuari registrat un correu electrònic per tal de verificar la identitat de l'usuari. Al fer clic al link del missatge, l'usuari ja podrà accedir a la plataforma iniciant sessió a la pàgina d'inici de sessió que es mostra a la següent figura.

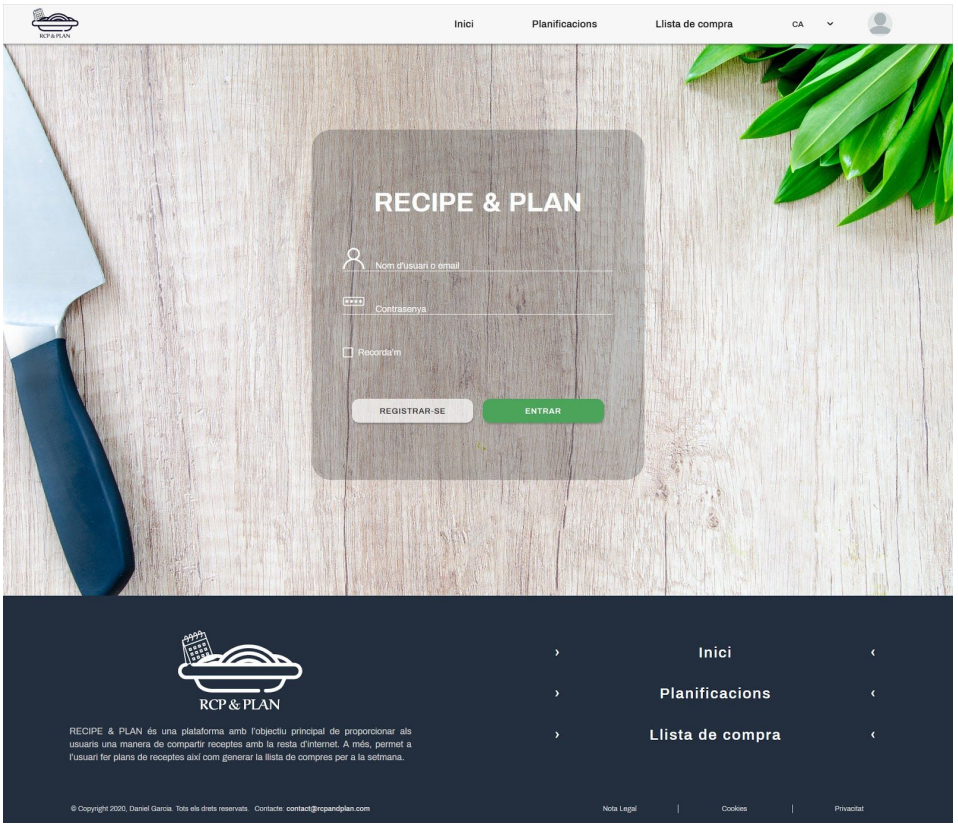


Figura 87- Pàgina d'inici de sessió pels usuaris ja registrats

Un cop l'usuari s'ha identificat al sistema, la seva imatge de perfil es mostra a la part dreta i ja podrà utilitzar les funcionalitats que no estaven disponibles per la resta d'usuaris.
Una pàgina que un usuari no registrat no té és la pàgina del seu propi usuari, en la qual es mostra el contingut que veuen tots els usuaris, i a més les receptes privades de l'usuari, i el botó que permetia afegir a l'usuari de la pàgina com a contacte, ara serveix per editar la informació de l'usuari.

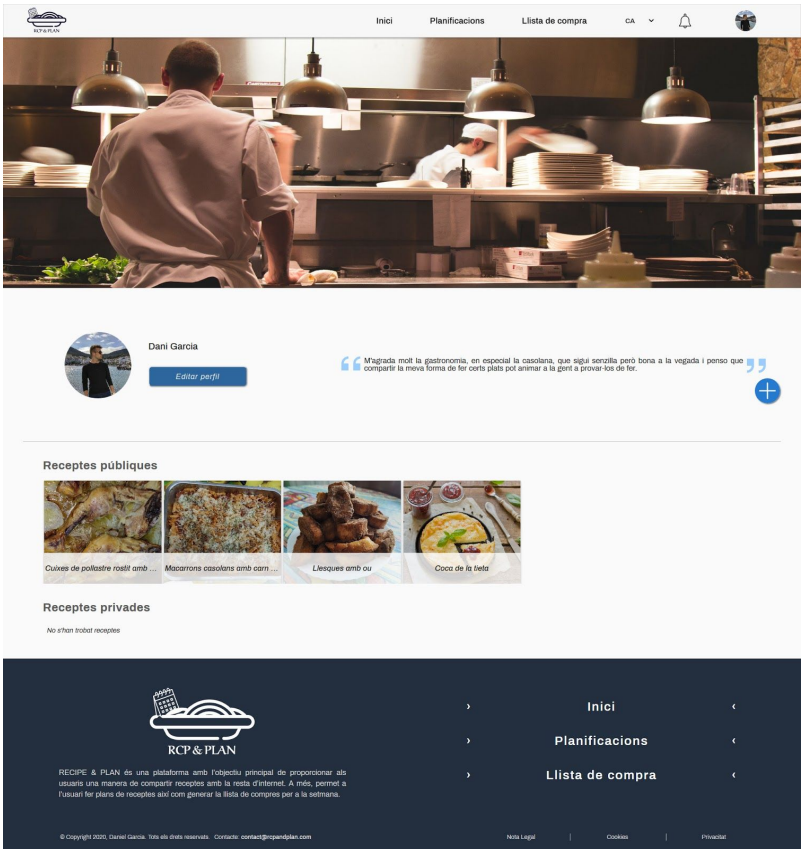


Figura 88- Pàgina del perfil de l'usuari de la sessió

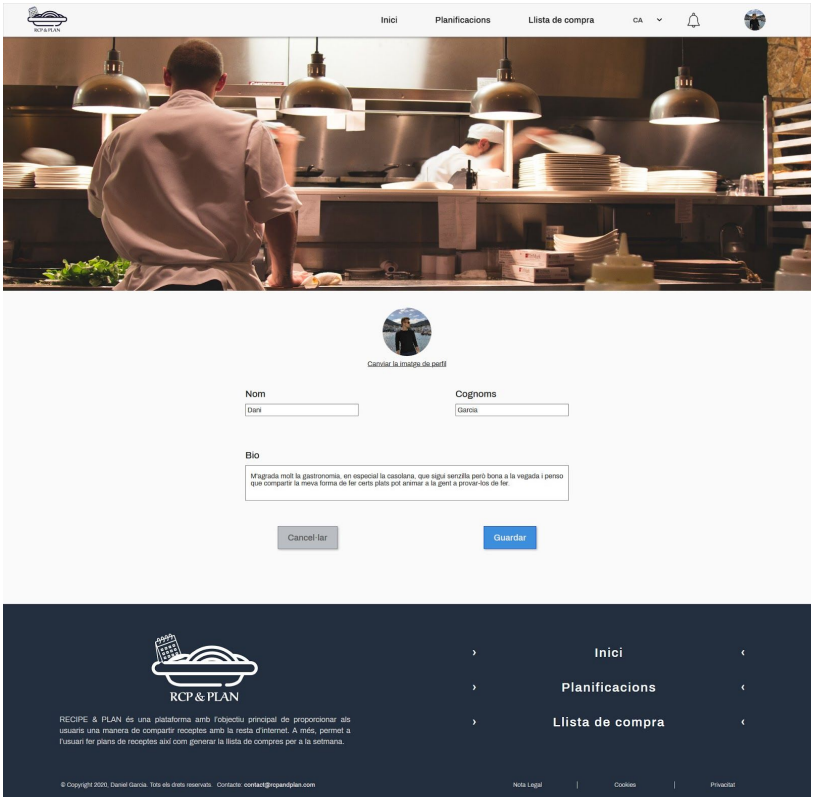


Figura 89- Pàgina d'edició de la informació de l'usuari

Una altra pàgina que canvia una mica és la fitxa d'una recepta, l'usuari al estar registrat ara pot valorar les receptes clicant els botons a la secció de valoracions, i a més també pot registrar el seu temps de preparació de la recepta, d'aquesta forma contribuir mostrar una mitjana del temps que realment es passen els usuaris fent la recepta. A més a la Figura 90 es pot veure el menú desplegable de la recepta que permet eliminar la recepta, o bé canviar-li la visibilitat o editar-la. Això és degut a que l'usuari registrat és l'autor i per tant, pot fer certes accions a la recepta que els altres usuaris no poden fer.

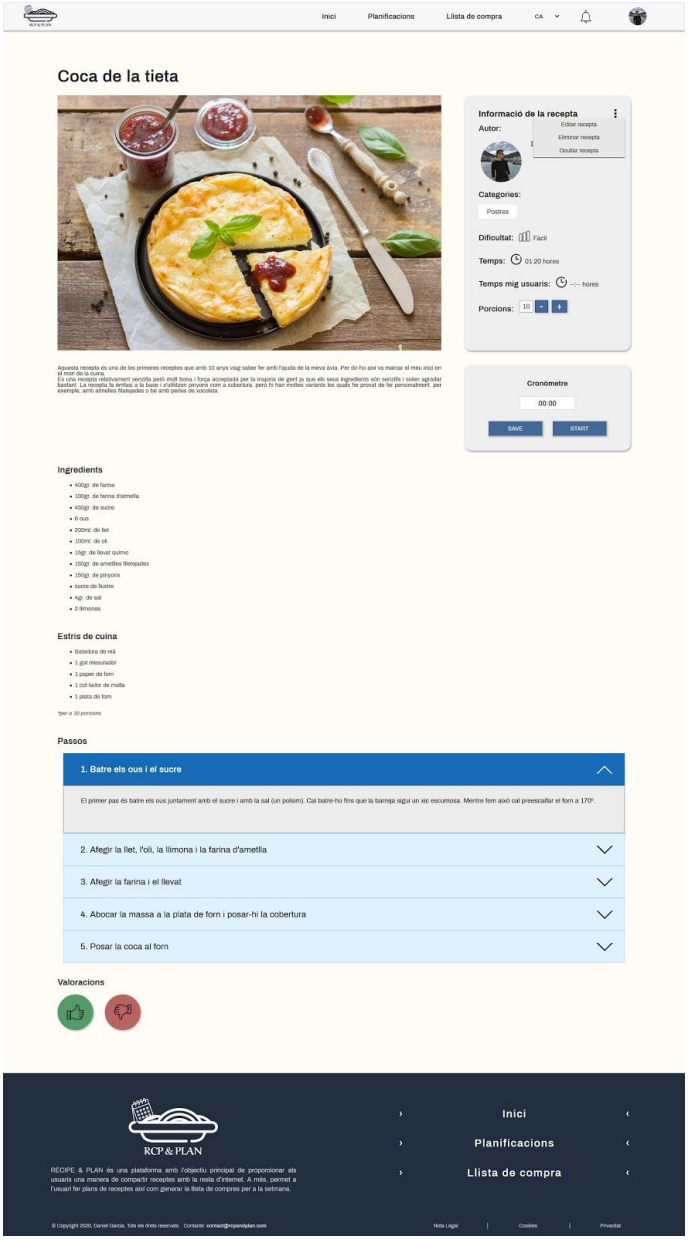


Figura 90- Pàgina de la fitxa d'una recepta utilitzant l'usuari autor de la recepta

Dins el menú d'opcions hi ha l'opció d'editar la recepta, la qual ens porta a la pàgina de creació i edició de receptes, allà s'hi introdueixen totes les dades de la recepta, des de les categories a les que pertany la recepta, el títol, les imatges de la recepta, una descripció opcional, la llista d'ingredients i estris, i els passos als quals es pot afegir alguna imatge per

ajudar l'explicació amb elements gràfics. A la Figura 91(separada en dos parts per la llargada) es pot veure la estructura completa d'aquesta pàgina.

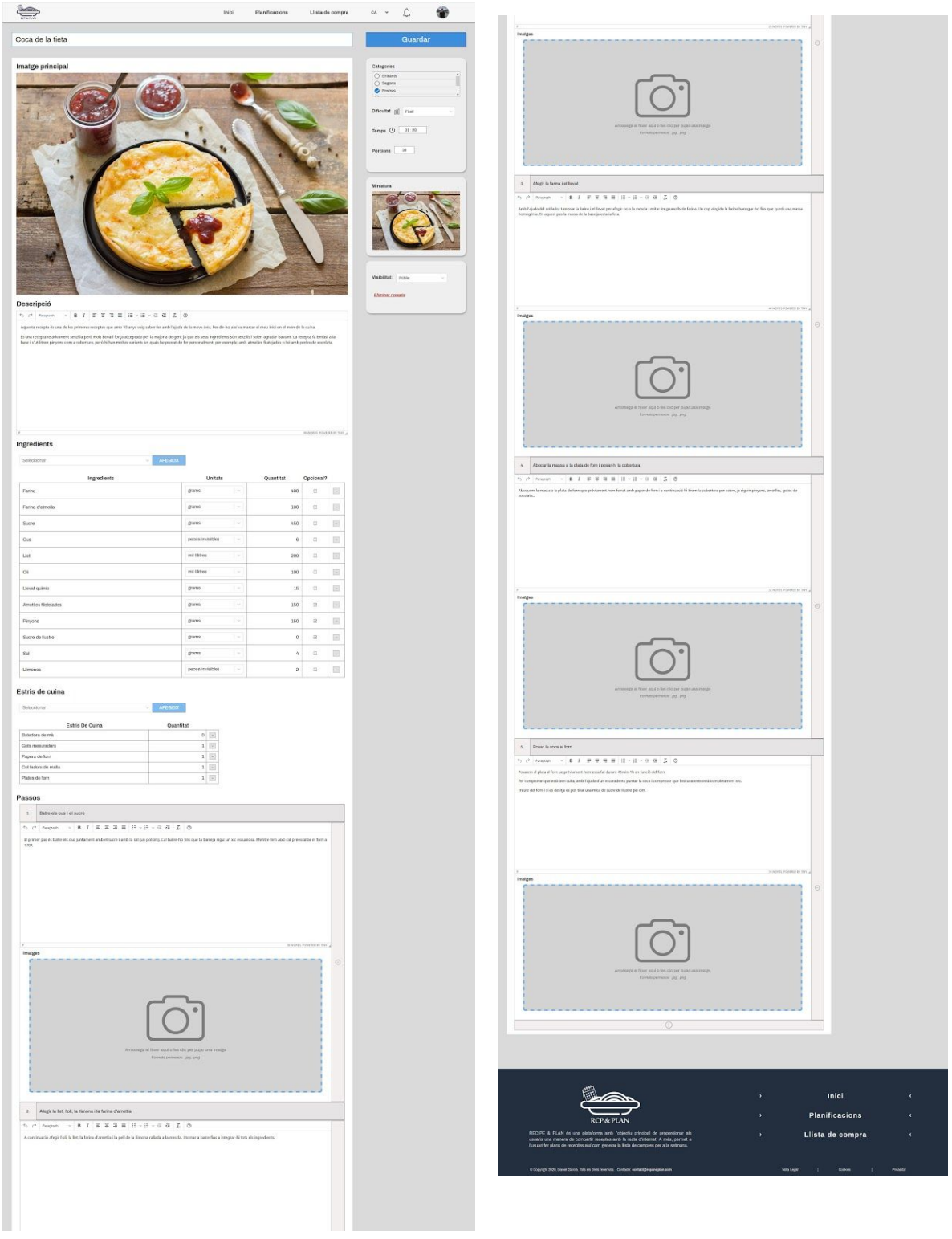


Figura 91- Pàgina d'edició d'una recepta

D'aquesta pàgina una secció que crec que és important comentar com funciona, és la dels ingredients, la qual et permet seleccionar un ingredient o bé crear-ne un de nou escrivint l'ingredient al selector i seleccionant l'opció de crear, llavors apareixerà un diàleg on introduir el nom de l'ingredient en singular per poder-lo crear, i finalment aquest ingredient

s'afegirà a la base de dades, i a la taula d'ingredients de la recepta; les Figura 92 i 93 reflexen aquest procés.

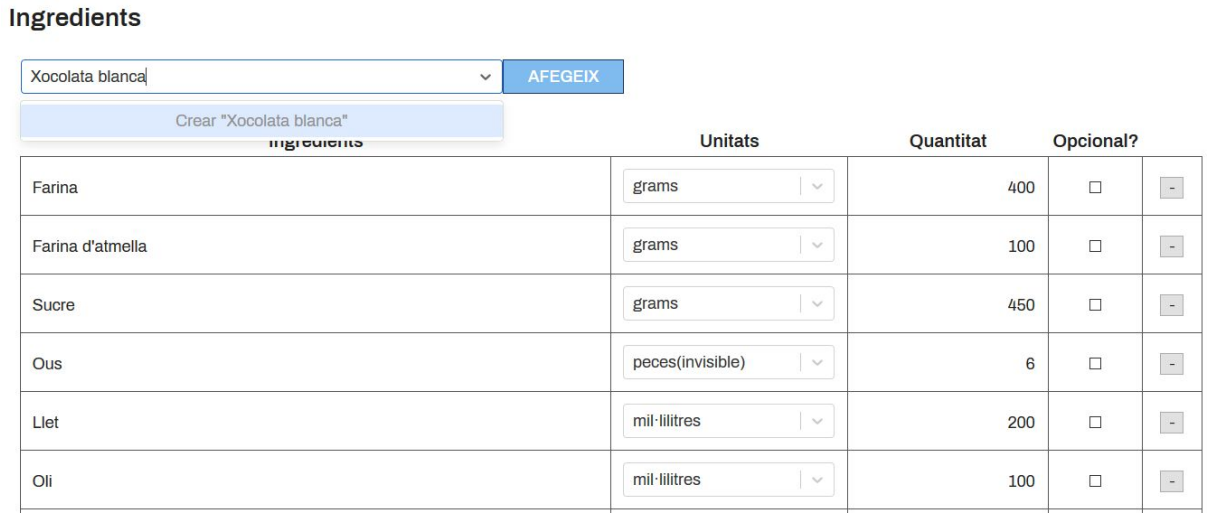


Figura 92- Pàgina d'edició d'una recepta (secció ingredients)

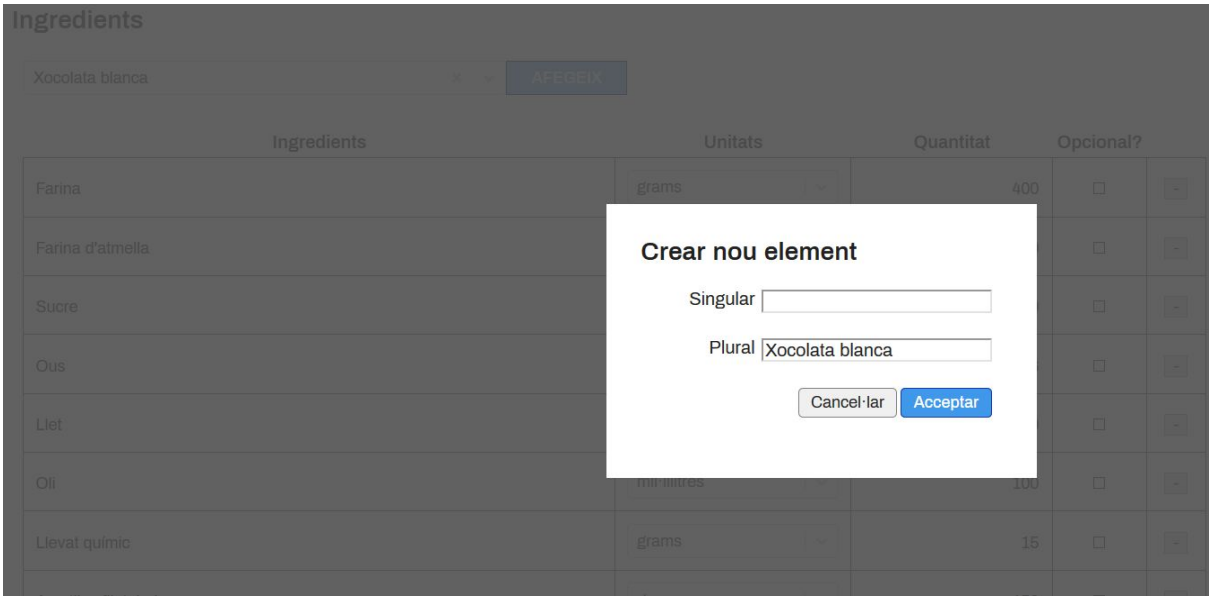


Figura 93- Pàgina d'edició d'una recepta (diàleg de creació d'ingredients)

Un cop assolit el primer punt, la plataforma ja permet publicar i consultar receptes, per tant, un cop això fet, el següent pas ha estat implementar la funcionalitat de planificació de les receptes al llarg d'un període. Per fer-ho hi ha una pàgina que mostra les planificacions de l'usuari en format de calendari, que es pot anar consultant els propers períodes o bé els anteriors mitjançant els botons de navegació.

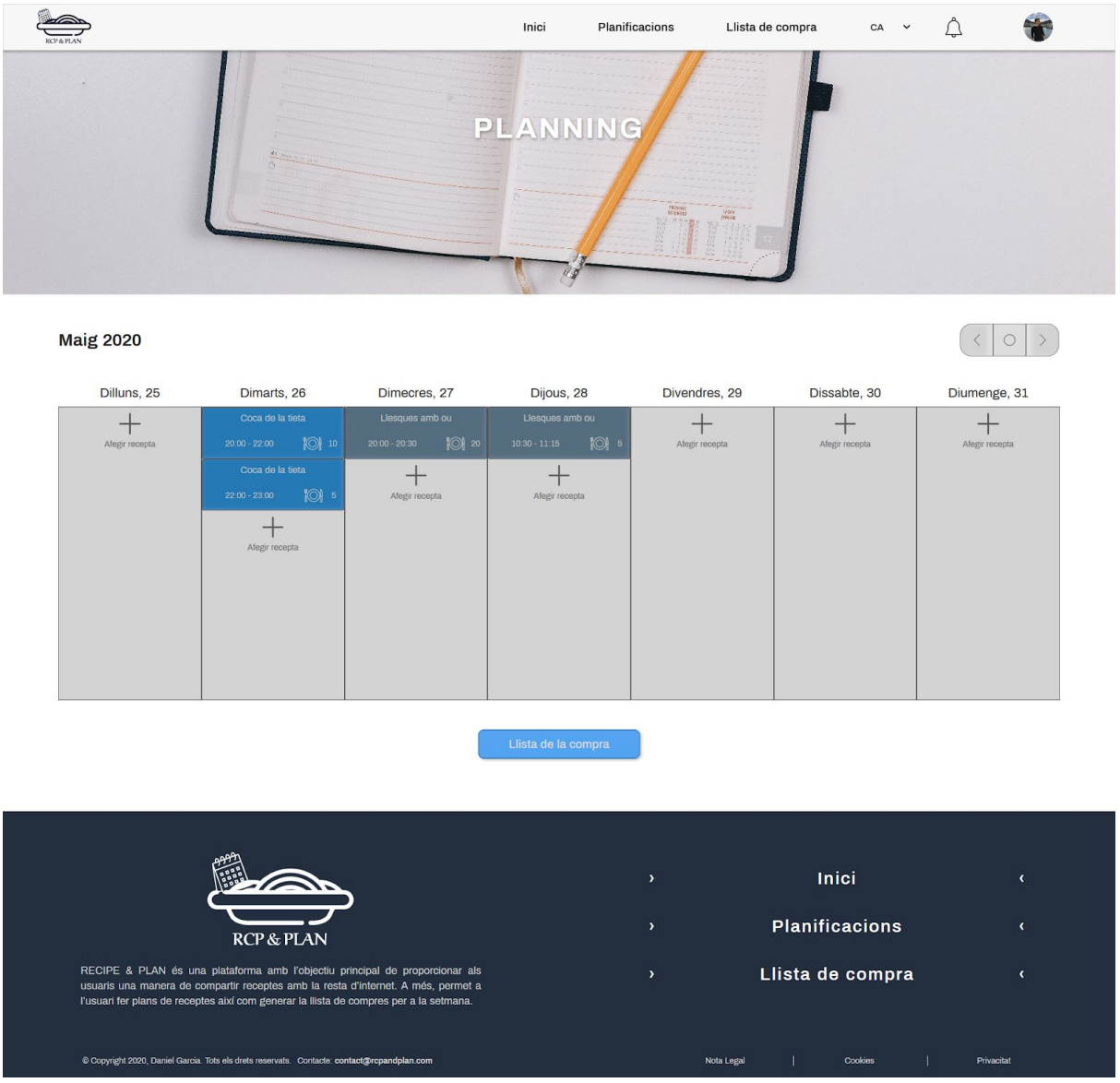


Figura 94- Pàgina de planificacions

Cada dia té la opció per poder afegir una nova planificació d'una recepta, per fer-ho s'obre un diàleg que permet cercar receptes i al seleccionar la recepta a planificar, llavors s'especifica a quina hora es vol començar i a quina hora es pretén acabar, a més l'usuari ha d'anotar quantes porcions té plantejat fer. Si es selecciona una targeta d'una planificació no bloquejada(blau més clar) aleshores apareix una opció a sota del mes i any que permet eliminar aquella planificació.

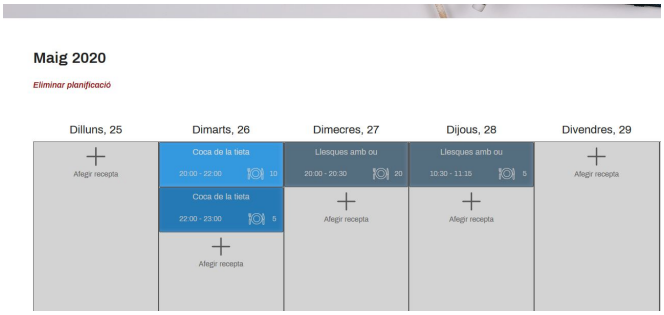


Figura 95- Pàgina de planificacions (opció eliminar mostrada)

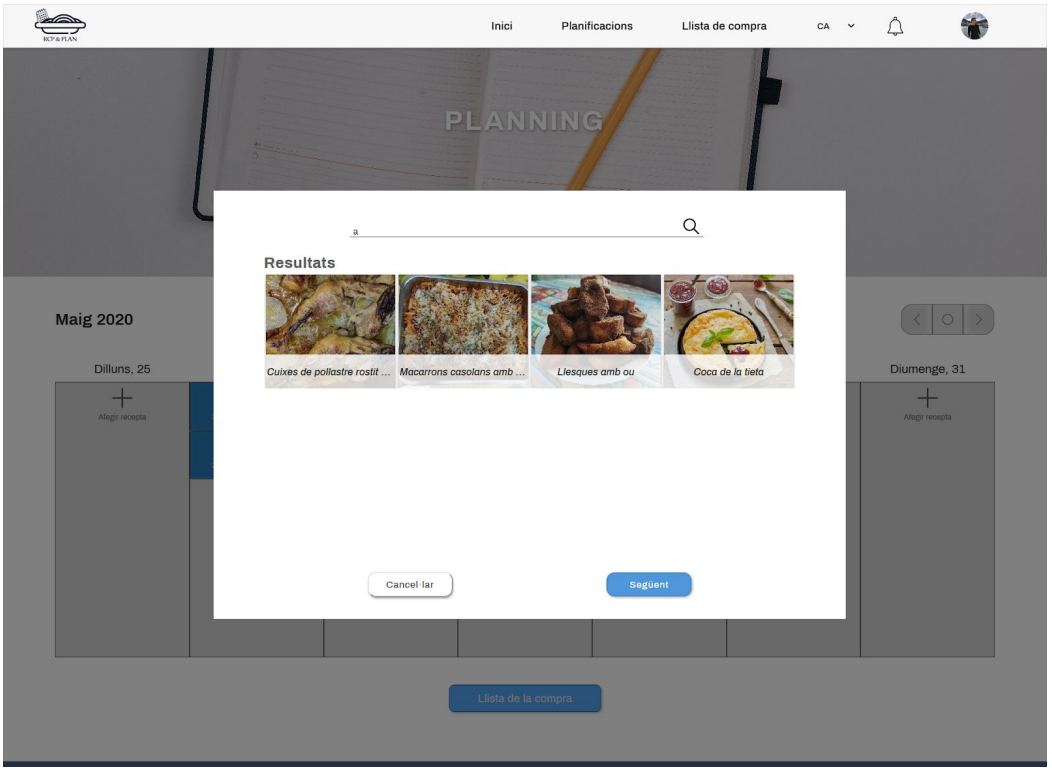


Figura 96- Pàgina de planificacions(diàleg afegir planificació)

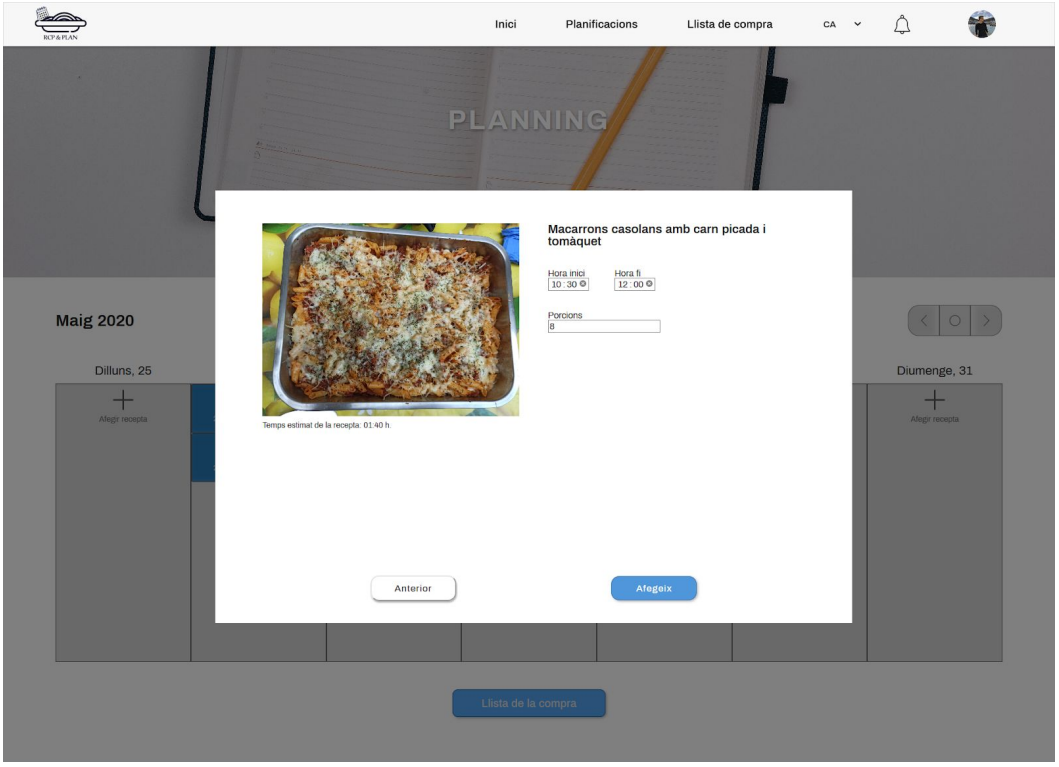


Figura 97- Pàgina de planificacions(diàleg afegir planificació introduir informació)

Un cop els usuaris tenen la capacitat de poder generar les seves planificacions, el següent punt dels objectius del projecte era proporcionar als usuaris la opció de generar la llista de la compra per la planificació de receptes que tenen. Per fer-ho a la pàgina de planificacions hi ha un botó a la part inferior per generar la llista de la compra per un interval concret.

Cada llista de la compra és determinada per la seva data inicial i final, per tant, al crear una llista cal tenir en compte que si es tria unes dates coincidents amb les d'una altra llista, aleshores mostrarà aquella llista, sense crear-ne una de nova. D'altra banda, en cas de que les dates escollides es solapin amb les d'una altra llista, es mostrarà un missatge d'error. Per contra quan una llista ja està completada no afecta la creació d'una altra llista amb les dates sol·lapades.

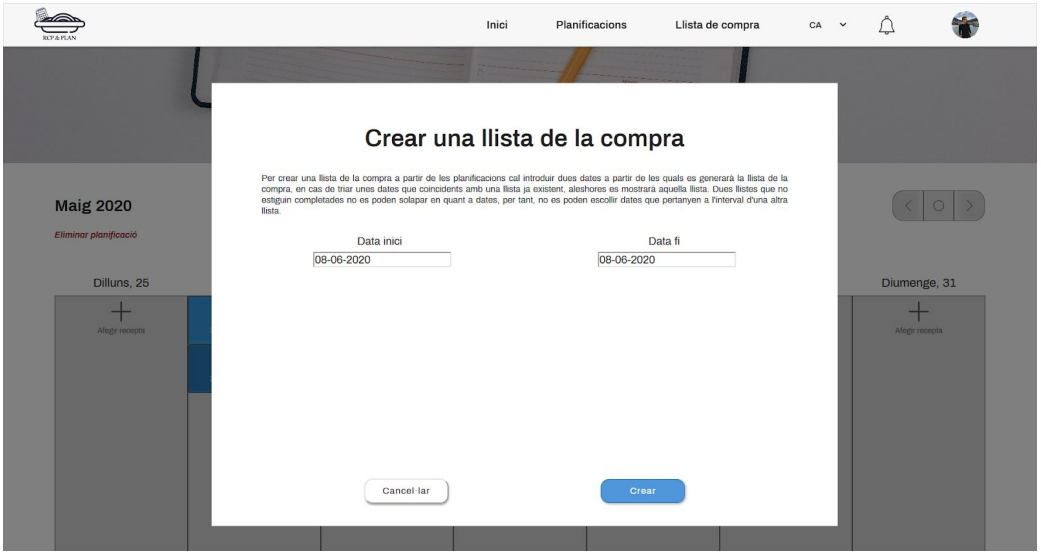


Figura 98- Pàgina de planificacions(diàleg crear llista compra)

Un cop creada la llista, o bé a través del menú de la capçalera o del peu podem accedir a la pàgina de la llista de la compra, la qual ens mostra la llista ja creada, o bé hi ha una opció per triar les llistes de la compra disponibles per poder-la mostrar. Si la llista no està completada al fer clic sobre un ingredient de la llista es canviarà l'estat de l'element marcant-lo com a comprat o desmarcant-lo en funció del seu estat anterior. A la part inferior de la pàgina hi ha un botó que permet eliminar la llista i així desbloquejar les planificacions que contenia la llista o bé en cas de que tots els elements estiguin marcats permet completar la llista.

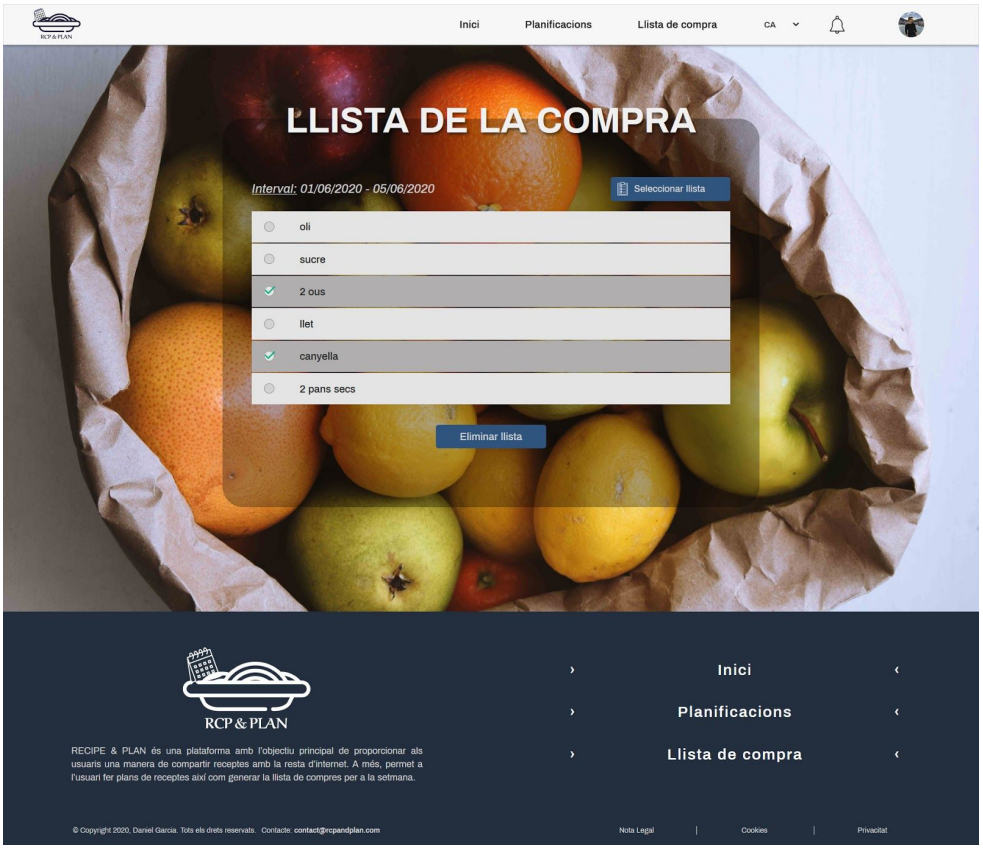


Figura 99- Pàgina de llista de la compra

Ja per acabar el repàs dels punts, caldria comentar que el darrer punt degut al temps no ha estat possible d'implementar a més de ser un punt que hagués representat un volum de feina important.

D'altra banda, tal com he esmentat en l'apartat de viabilitat legal i contractual, aquesta web s'ha d'ajustar a la Llei orgànica de protecció de dades personals i garantia dels drets digitals i al Reglament General de la Protecció de dades. Per fer-ho s'han près les mesures de seguretat i informació esmentades al document de l'aplicació Facilita de l'Agència Espanyola de Protecció de Dades, incloent, tant la primera capa d'informació en el formulari de registre, així com també la segona capa d'informació a la pàgina de privacitat de la web. A més també s'informa als usuaris sobre l'ús de galetes a través d'un missatge resumit i també a través de la pàgina de política de cookies.

Finalment, per una web els resultats solen ser els visitants i el temps que es passen els usuaris dins la web. Per obtenir aquesta informació el servei de hosting facilita eines per l'anàlisi del tràfic a la web. Un exemple són les estadístiques de AwStats que tenen un format com el de la següent figura. Cal destacar que les mesures són el nombre diferent de visitants web,el nombre pàgines carregades, el nombre de peticions fetes… Hi ha diferents gràfics i taules en funció de la granularitat temporal, és a dir, si està agrupat per mesos, o bé per dies. Inclús hi ha una taula amb la classificació de les ips que més visiten la web

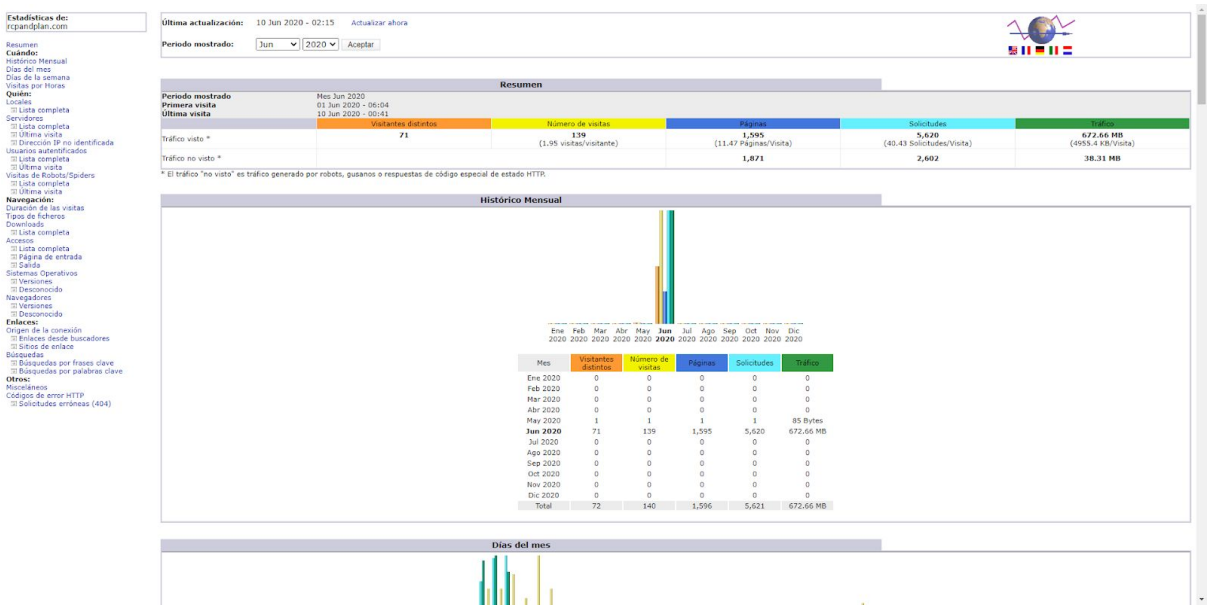


Figura 100- Eina d'estadístiques de la web

11. Conclusions

El projecte té per objectiu crear una solució al problema plantejat anteriorment a partir de:

- La creació d'una plataforma que permeti als usuaris compartir receptes(publicar les seves pròpies, consultar les receptes d'altres usuaris, valorar-les).
- Afegir a la plataforma la funcionalitat de planificació de receptes en un període per organitzar l'alimentació de l'usuari.
- Afegir la possibilitat d'obtenir una llista de tots els ingredients necessaris per dur a terme la planificació prevista.
- Al ser una plataforma on qualsevol usuari pot penjar-hi contingut, afegir la capacitat de moderar el contingut dels usuaris per part d'alguns usuaris específics.

Un cop havent desenvolupat el projecte puc concloure que el producte final és un producte vàlid tenint en compte que el projecte tenia una planificació ajustada que ha provocat que, degut a certs factors, no s'ajustés del tot a la realitat. Tot i això valoro positivament la feina feta ja que excepte el darrer punt que hauria permès la moderació de la plataforma, la resta sí que s'ha pogut aconseguir. També valoro positivament el projecte en quant a aprenentatge que penso que tot el procés del projecte m'ha fet posar en pràctica els coneixements de la carrera així com també m'ha fet aprendre noves coses.

Per començar la gestió del projecte ha fet que aprengui la metodologia de gestió de projectes del PMBOK ja explicat. A més al haver de dissenyar una web he hagut d'adquirir unes nocions mínimes en quant a disseny de pàgines web i també la utilització d'un programa per crear maquetes com el Lunacy.

De cares al desenvolupament al utilitzar un llenguatge el qual al començar el projecte en tenia un nivell força baix, els problemes i dificultats en el desenvolupament del projecte han fet que adquirís una mica més de nivell del llenguatge. A més al utilitzar frameworks, el projecte m'ha servit per entendre més profundament com funcionen i com fer-los servir. També m'ha servit a nivell de disseny i gestió de bases de dades per posar en pràctica els coneixements impartits a la carrera.

Fins aquí el desenvolupament ja m'ha aportat una experiència força enriquidora, ara bé, al voler anar més enllà i desplegar la web al servidor, ha requerit per exemple alguns dels coneixements de xarxes impartits a la carrera. La experiència de penjar la web al servidor m'ha servit per aprendre com implementar el protocol https a la web, com gestionar les connexions al servidor mitjançant l'ús de protocols FTP o SSH.

Havent dit tot això penso que aquest projecte ha sigut una bona forma de culminar i aplicar tota l'etapa d'aprenentatge del grau.

12. Treball futur

Per determinar el treball futur per aquest projecte, primer de tot, podríem tenir en compte l'objectiu que no he fet com a punt de partida, ja que coincideix amb els requisits que són més prioritaris.

Dels requisits funcionals del projecte els que han quedat sense satisfer són:

RF6. Tot usuari registrat pot compartir receptes amb els seus contactes *(Baixa)*

RF9. Tot usuari registrat es pot guardar receptes com a preferides. *(Baixa)*

RF10. Tot usuari registrat pot valorar receptes i fer-hi comentaris. *(Mitjana)*

RF11. Tot usuari registrat que hagi creat una recepta pot crear la seva respectiva traducció. *(Mitjana)*

Moderador

RF13. Els moderadors poden revisar receptes publicades pels usuaris, ocultar-les al públic o eliminar-les. *(Mitjana)*

RF14. Els moderadors poden revisar comentaris publicats pels usuaris, ocultar-les al públic o eliminar-les. *(Mitjana)*

RF15. Els moderadors poden notificar als usuaris sobre una recepta o comentari eliminat o moderat. *(Mitjana)*

RF16. Els moderadors poden eliminar ingredients i estris de cuina entrats. (Baixa)

Administrador

RF17. Tot administrador pot gestionar els usuaris i els seus rols. *(Mitjana)*

RF18. Tot administrador pot consultar les estadístiques generals de la plataforma, des de nombre de visualitzacions de receptes, a nombre d'altes d'usuaris, nombre de valoracions,.... *(Baixa)*

RF19. Tot administrador pot crear, modificar i eliminar les categories de receptes (*Mitjana*)

Per tant, per a un treball futur seguiria un ordre determinat per prioritat i per rol. És a dir, jo primer implementaria les funcionalitats de comentar receptes i crear les traduccions, i després implementaria les funcionalitats per moderar receptes, comentaris i ingredients/estris del sistema.

Una altre possibilitat de treball futur seria el desenvolupament d'una aplicació en Android i iOS que funcionés amb el mateix servidor que la web. Aquesta possibilitat és la que m'ha fet mantenir el servidor i el client completament separats.

Com a treball futur també es podria plantejar un sistema per utilitzar receptes per promocionar un o diversos productes com a ingredients. Diferenciant les receptes promocionades de les que no, i donant certa prioritat per mostrar les promocionals dins les categories.

13. Bibliografia

- Informació sobre els principals llenguatges web

https://www.piensasolutions.com/blog/principales-lenguajes-programacion-web/

- Informació sobre tendències de bases de dades

https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

- Informació sobre la guia de projectes PMBOK

Project Management Institute, PMI (2004) *PMBOK Project Management Base Of Knowledge*. PMI .6ª edición español

https://www.ealde.es/metodologia-direccion-de-proyectos-pmi/

- Informació respecte guardar contrasenyes a la base de dades

https://security.stackexchange.com/questions/211/how-to-securely-hash-passwords

- Informació sobre com implementar un web amb múltiples pàgines en React

https://medium.com/better-programming/how-to-pass-multiple-route-parameters-in-a-react-url-path-4b919de0abbe

https://programmingwithmosh.com/react/react-router-add-the-power-of-navigation/

- Informació sobre llocs web multiidioma en React

https://dev.to/halilcanozcelik/create-a-multi-language-website-with-react-context-api-4i27

- Documentació sobre el mòdul de carousels utilitzat en React

https://brainhubeu.github.io/react-carousel/docs/api/carousel

- Exemple de com implementar el mòdul de tallar imatges en React

https://codesandbox.io/s/croppie-to-crop-uploaded-base64-image-vtjnq?file=/src/index.js

- Informació sobre com detectar clics fora d'un element amb React

https://stackoverflow.com/questions/32553158/detect-click-outside-react-component

- Informació sobre com implementar un sistema de notificacions a temps real amb Socket.io

https://stackoverflow.com/questions/36931311/implement-notification-system-using-node-js-express-socket

- Documentació de la llibreria Socket.io

https://socket.io/docs/emit-cheatsheet/

- Documentació del mòdul de selectors de React

https://react-select.com/home

- Documentació sobre la instal·lació de Node.js al servidor

https://www.a2hosting.es/kb/installable-applications/manual-installations/installing-node-js-on-managed-hosting-accounts

14. Annexos

14.1. Annex 1

A continuació es mostraran les diferents activitats recollides a l'hora de planificar el projecte. Cadascuna conté 4 paràmetres que són:

- Estimació optimista(tO): És l'estimació de temps que es trigaria a realitzar l'activitat si tot anés bé a la primera.
- Estimació probable(tM): És l'estimació de temps que més probablement es trigaria a realitzar l'activitat. És un valor situat entre l'estimació optimista i l'estimació pessimista.
- Estimació pessimista(tP): És l'estimació de temps que com a molt es trigaria per realitzar l'activitat. Sol contemplar possibles contratemps i problemes que fessin retardar la duració de l'activitat.
- Durada esperada(tE): És un valor normalitzat producte de les estimacions anteriors. El càlcul es fa a partir de:

$$tE = \frac{(tO + 4 \cdot tM + tP)}{6}$$

Les activitats recollides són:

A1. Plantejar el problema a solucionar

tO: 1 hora
tM: 1,5 hores
tP: 3 hores
tE: 1,667 hores
Requisit completat: -

A2. Analitzar les necessitats plantejades al problema

tO: 0,5 hora
tM: 1 hores
tP: 1,5 hores
tE: 1 hora
Requisit completat: -

A3. Fer recerca sobre les solucions existents
tO: 0,75 hores
tM: 1,75 hores
tP: 2 hores
tE: 1,625 hores
Requisit completat: -

A4. Proposar una solució al problema plantejat
tO: 0,75 hores
tM: 1 hora
tP: 1,5 hores
tE: 1,04 hores
Requisit completat: -

A5. Definir l'abast de la solució
tO: 1 hora
tM: 1,25 hores
tP: 1,5 hores
tE: 1,25 hores
Requisit completat: -

A6. Documentar els requeriments funcionals de la pàgina web
tO: 2 hores
tM: 3,25 hores
tP: 4,75 hores
tE: 3,29 hores
Requisit completat: -

A7. Documentar els requeriments no funcionals de la web
tO: 1,5 hores
tM: 2 hores
tP: 3,5 hores
tE: 2,16 hores
Requisit completat: -

A8. Establir prioritats entre els requeriments

tO: 0,5 hores
tM: 0,75 hores
tP: 0,9 hores
tE: 0,73 hores
Requisit completat: -

A9. Generar la matriu de dependències entre requeriments

tO: 0,5 hores
tM: 0,6 hores
tP: 0,9 hores
tE: 0,63 hores
Requisit completat: -

A10. Generar els paquets de treball a partir dels requisits

tO: 1,25 hora
tM: 1,6 hores
tP: 2,4 hores
tE: 1,675 hores
Requisit completat: -

A11. Crear les activitats i construir-ne el diagrama de xarxa

tO: 2 hores
tM: 3 hores
tP: 3,5 hores
tE: 2,916 hores
Requisit completat: -

A12. Construir el diagrama de Gantt a partir de les activitats i el temps previst

tO: 1,5 hores
tM: 3,5 hores
tP: 4,5 hores
tE: 3,33 hores
Requisit completat: -

A13. Dissenyar el diagrama de casos d'ús UML

tO: 4 hores
tM: 6 hores
tP: 7 hores
tE: 5,83 hores
Requisit completat: -

A14. Documentar les fitxes dels diferents casos d'ús

tO: 6 hores
tM: 9 hores
tP: 11 hores
tE: 8,83 hores
Requisit completat: -

A15. Dissenyar els diversos diagrames d'activitats respecte els casos d'ús UML

tO: 7 hores
tM: 8,5 hores
tP: 12 hores
tE: 8,83 hores
Requisit completat: -

A16. Dissenyar els diagrames de seqüència dels casos d'ús UML

tO: 9 hores
tM: 11 hores
tP: 11,5 hores
tE: 10,75 hores
Requisit completat: -

A17. Investigar sobre les possibles tecnologies a utilitzar i els seus beneficis

tO: 0,75 hores
tM: 1,25 hores
tP: 1,5 hores
tE: 1,208 hores
Requisit completat: -

A18. Escollir les diferents tecnologies necessàries per realitzar cada tasca

tO: 0,2 hora
tM: 0,3 hores
tP: 0,5 hores
tE: 0,3167 hores
Requisit completat: -

A19. Preparar l'entorn de treball amb les tecnologies escollides

tO: 1,25 hora
tM: 2 hores
tP: 2,25 hores
tE: 1,9167 hores
Requisit completat: RNF6

A20. Construir el diagrama EER que reflexi tota la informació a guardar

tO: 7 hores
tM: 8 hores
tP: 8,25 hores
tE: 7,875 hores
Requisit completat: -

A21. Determinar per cada entitat i relació els camps que han de contenir

tO: 2 hores
tM: 3 hores
tP: 3,5 hores
tE: 2,916 hores
Requisit completat: -

A22. Transformar el model lògic a model físic

tO: 0,5 hores
tM: 0,75 hores
tP: 1 hores
tE: 0,75 hores
Requisit completat: -

A23. Normalitzar les taules si s'escau

tO: 0 hores
tM: 0,2 hores
tP: 0,3 hores
tE: 0,183 hores
Requisit completat: -

A24. Disseny de la pàgina Home

tO: 9 hores
tM: 12 hores
tP: 13 hores
tE: 11,67 hores
Requisit completat: -

A25. Disseny de la pàgina de Planificació

tO: 8 hores
tM: 10 hores
tP: 12 hores
tE: 10 hores
Requisit completat: -

A26. Disseny de la pàgina Nota Legal, Política de Cookies, Política de Privacitat

tO: 5 hores
tM: 6 hores
tP: 6,5 hores
tE: 5,9167 hores
Requisit completat: -

A27. Disseny de la fitxa d'una recepta

tO: 9 hores
tM: 12 hores
tP: 13 hores
tE: 11,67 hores
Requisit completat: -

A28. Disseny de la capçalera i el peu de pàgina de la web

tO: 4 hores
tM: 5 hores
tP: 6 hores
tE: 5 hores
Requisit completat: -

A29. Disseny de la pàgina de Login i de Registre

tO: 7 hores
tM: 9 hores
tP: 10 hores
tE: 8,83 hores
Requisit completat: -

A30. Disseny de la pàgina de perfil d'usuari

tO: 5 hores
tM: 8 hores
tP: 9,5 hores
tE: 7,75 hores
Requisit completat: -

A31. Disseny de la pàgina de creació/edició de receptes

tO: 5 hores
tM: 6.5 hores
tP: 9 hores
tE: 6,67 hores
Requisit completat: -

A32. Disseny de la pàgina del Panell General

tO: 3 hores
tM: 4 hores
tP: 4,5 hores
tE: 3,9167 hores
Requisit completat: -

A33. Disseny de la pàgina d'administració d'usuaris
tO: 2,5 hora
tM: 3,5 hores
tP: 4,5 hores
tE: 3,5 hores
Requisit completat: -

A34. Disseny de la pàgina d'administració de receptes
tO: 2 hores
tM: 3,5 hores
tP: 4,5 hores
tE: 3,4167 hores
Requisit completat: -

A35. Disseny de la pàgina d'administració d'ingredients
tO: 2 hora
tM: 3,5 hores
tP: 4,5 hores
tE: 3,4167 hores
Requisit completat: -

A36. Disseny de la pàgina d'administració d'estris de cuina
tO: 2 hora
tM: 3,5 hores
tP: 4,5 hores
tE: 3,4167 hores
Requisit completat: -

A37. Disseny de la pàgina d'administració de categories
tO: 2 hora
tM: 3,5 hores
tP: 4,5 hores
tE: 3,4167 hores
Requisit completat: -

A38. Disseny de la pàgina d'edició del perfil d'usuari

tO: 2,5 hora
tM: 3,5 hores
tP: 4,5 hores
tE: 3,5 hores
Requisit completat: -

A39. Integrar la maqueta de la Home

tO: 8 hores
tM: 12 hores
tP: 15 hores
tE: 11,83 hores
Requisit completat:RNF1

A40. Integrar capçalera i peu de pàgina de la web

tO: 0,75 hores
tM: 2 hores
tP: 3,5 hores
tE: 2,042 hores
Requisits completats:RNF1,RNF2

A41. Integrar pàgina de Planificació

tO: 12 hores
tM: 14 hores
tP: 15 hores
tE: 13,83 hores
Requisits completats:RNF1,RNF2

A42. Integrar pàgina de Llista de la Compra

tO: 8 hores
tM: 10 hores
tP: 11 hores
tE: 9,83 hores
Requisits completats: RNF1,RNF2

A43. Integrar pàgina de Nota Legal

tO: 1,25 hores
tM: 1,5 hores
tP: 2 hores
tE: 1,542 hores
Requisits completats: RNF1,RNF2

A44. Integrar pàgina de Política de Cookies

tO: 1,25 hores
tM: 1,5 hores
tP: 2 hores
tE: 1,542 hores
Requisits completats:RNF1,RNF2

A45. Integrar pàgina de Política de Privacitat

tO: 1,25 hores
tM: 1,5 hores
tP: 2 hores
tE: 1,542 hores
Requisits completats:RNF1,RNF2

A46. Integrar fitxa d'una Recepta

tO: 10 hores
tM: 13 hores
tP: 15 hores
tE: 12,83 hores
Requisits completats:RNF1,RNF2

A47. Integrar pàgina de Login

tO: 5 hores
tM: 8 hores
tP: 9 hores
tE: 7,67 hores
Requisits completats:RNF1,RNF2

A48. Integrar pàgina de Registre

tO: 5 hores
tM: 8 hores
tP: 9 hores
tE: 7,67 hores
Requisits completats: RNF1,RNF2

A49. Integrar pàgina de Perfil d'usuari

tO: 10 hores
tM: 12 hores
tP: 14 hores
tE: 12 hores
Requisits completats: RNF1,RNF2

A50. Integrar pàgina d'edició de receptes

tO: 15 hores
tM: 18 hores
tP: 22 hores
tE: 18,167 hores
Requisits completats: RNF1,RNF2

A51. Integrar pàgina del Panell general

tO: 12 hores
tM: 13 hores
tP: 13,5 hores
tE: 12,917 hores
Requisits completats: RNF1,RNF2

A52. Integrar pàgina d'administració d'usuaris

tO: 8 hores
tM: 12 hores
tP: 13 hores
tE: 11,5 hores
Requisits completats:RNF1,RNF2

A53. Integrar pàgina d'administració de receptes

tO: 5 hores
tM: 7 hores
tP: 9,5 hores
tE: 7,08 hores
Requisits completats: RNF1,RNF2

A54. Integrar pàgina d'administració d'ingredients

tO: 5 hores
tM: 7 hores
tP: 9 hores
tE: 7 hores
Requisits completats: RNF1,RNF2

A55. Integrar pàgina d'administració d'estris de cuina

tO: 5 hores
tM: 7 hores
tP: 9 hores
tE: 7 hores
Requisits completats: RNF1,RNF2

A56. Integrar pàgina d'administració de categories

tO: 4 hores
tM: 6 hores
tP: 7 hores
tE: 5,83 hores
Requisit completat:RNF1,RNF2

A57. Integrar pàgina d'edició del perfil d'usuari

tO: 6 hores
tM: 6,5 hores
tP: 7,5 hores
tE: 6,58 hores
Requisits completats:RNF1,RNF2

A58. Desenvolupar el mòdul de connexió amb la base de dades

tO: 8 hores
tM: 12 hores
tP: 15 hores
tE: 11,83 hores
Requisits completats: RNF3,RNF4

A59. Desenvolupar el mòdul/mòduls per gestionar la lògica dels usuaris

tO: 13 hores
tM: 15 hores
tP: 17 hores
tE: 15 hores
Requisits completats: RF2,RF5,RF17

A60. Desenvolupar el mòdul/mòduls per gestionar la lògica de les receptes

tO: 12 hores
tM: 13 hores
tP: 15 hores
tE: 13,167 hores
Requisits completats: RF1,RF3,RF4,RF9,RF10,RF11,RNF7

A61. Desenvolupar el mòdul/mòduls per gestionar la lògica de les categories

tO: 5 hores
tM: 8 hores
tP: 9 hores
tE: 7,67 hores
Requisits completats: RF19,RNF7

A62. Desenvolupar el mòdul/mòduls per gestionar la lògica de les planificacions

tO: 10 hores
tM: 11 hores
tP: 13 hores
tE: 11,17 hores
Requisit completat: RF7

A63. Desenvolupar el mòdul/mòduls per gestionar la lògica dels les llistes de la compra

tO: 6 hores
tM: 8 hores
tP: 9 hores
tE: 7,83 hores
Requisit completat: RF8

A64. Desenvolupar el mòdul/mòduls per gestionar la lògica dels ingredients

tO: 5 hores
tM: 6,5 hores
tP: 7,5 hores
tE: 6,42 hores
Requisits completats: RF12,RF16,RNF7

A65. Desenvolupar el mòdul/mòduls per gestionar les unitats de mesura

tO: 3 hores
tM: 4,5 hores
tP: 5,5 hores
tE: 4,42 hores
Requisits completats: RF12,RNF7

A66. Desenvolupar el mòdul/mòduls per gestionar la lògica dels estris de cuina

tO: 5 hores
tM: 6,5 hores
tP: 7,5 hores
tE: 6,42 hores
Requisits completats: RF12,RF16,RNF7

A67. Desenvolupar el mòdul/mòduls per gestionar la lògica del Panell general

tO: 6 hores
tM: 9 hores
tP: 12 hores
tE: 9 hores
Requisits completats: RF13,RF18

A68. Desenvolupar el mòdul/mòduls per gestionar la lògica dels comentaris

tO: 3 hores
tM: 4,5 hores
tP: 7 hores
tE: 4,33 hores
Requisit completat: RF15

A69. Desenvolupar el mòdul/mòduls per gestionar la lògica de les receptes compartides

tO: 7 hores
tM: 11 hores
tP: 15 hores
tE: 11 hores
Requisit completat: RF6

A70. Crear els testos respectius a les funcions que ho requereixin

tO: 10 hores
tM: 28 hores
tP: 35 hores
tE: 26,17 hores
Requisit completat: -

A71. Comprovar el correcte funcionament de la plataforma

tO: 15 hores
tM: 25 hores
tP: 40 hores
tE: 25,83 hores
Requisit completat: -

A72. Obtenir hosting on poder penjar la web

tO: 1,5 hores
tM: 2,5 hores
tP: 4 hores
tE: 2,58 hores
Requisit completat: -

A73. Penjar la web a un servidor

tO: 10 hores
tM: 14 hores
tP: 17 hores
tE: 13,83 hores
Requisit completat: -

A74. Comprovar el correcte funcionament de la plataforma dins el servidor

tO: 2 hores
tM: 4 hores
tP: 6 hores
tE: 4 hores
Requisit completat: -

A75. Comprovar la satisfacció dels usuaris de la web

tO: 2 hores
tM: 3,5 hores
tP: 4 hores
tE: 3,33 hores
Requisit completat: -

14.2. Annex 2

- Consultar informació legal i sobre privacitat

CAS D'ÚS:	Consultar informació legal i sobre privacitat (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Consultar les pàgines de nota legal, privacitat, i cookies
Actors	Usuari no registrat
Funcionalitat	Llegir informació legal, de privacitat i cookies
Precondició	Recepta creada i visible públicament
Flux principal	1. Escollir la pàgina a consultar (legal, cookies, privacitat) 2. Mostrar pàgina amb el contingut escollit
Postcondició	Pàgina mostrada

- Llegir receptes

CAS D'ÚS:	Llegir receptes (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Llegir recepta pública
Actors	Usuari no registrat
Funcionalitat	Llegeix una recepta
Precondició	Recepta creada i visible públicament
Flux principal	1. Seleccionar Receptes amb visibilitat pública 2. Escollir Recepta 3. Seleccionar Ingredients de la Recepta 4. Seleccionar Estris de Cuina de la Recepta 5. Seleccionar Passos de la Recepta 6. Seleccionar Comentaris públics de la Recepta 7. Mostrar Recepta, Ingredients, Estris de cuina, Passos i Comentaris seleccionats
Postcondició	Recepta mostrada

CAS D'ÚS:	Llegir receptes (versió anàlisi)
Descripció	Escenari secundari del cas d'ús: Llegir recepta privada
Actors	Usuari registrat, moderador, administrador
Funcionalitat	Llegeix una recepta privada
Precondició	Recepta creada i no visible
Flux principal	1. Seleccionar Receptes amb visibilitat privada 2. Escollir Recepta 3. Si l'usuari n'és l'autor o és un moderador o administrador 3.1. Seleccionar Ingredients de la Recepta 3.2. Seleccionar Estris de Cuina de la Recepta 3.3. Seleccionar Passos de la Recepta 3.4. Seleccionar Comentaris públics de la Recepta 3.5. Si l'usuari és l'autor del Comentari o és moderador o administrador 3.5.1. Seleccionar Comentaris privats de la Recepta 3.6. Mostrar Recepta, Ingredients, Estris de cuina, Passos i Comentaris seleccionats. 4. Altrament 4.1. Redirigir a la pàgina principal 5. FSi
Postcondició	Recepta mostrada o usuari redirigit.

- Entrar com a usuari registrat

CAS D'ÚS:	**Entrar com a usuari registrat (versió anàlisi)**
Descripció	Escenari principal del cas d'ús: Entrar com a usuari registrat
Actors	Usuari no registrat
Funcionalitat	Entra les dades per identificar-se en el sistema
Precondició	Cert
Flux principal	1. Entrar nom d'usuari i contrasenya de l'Usuari 2. Buscar Usuari per nom d'usuari 3. Si existeix un Usuari activat amb el nom d'usuari entrat 3.1. Si la contrasenya entrada coincideix amb la de l'Usuari 3.1.1. Redirigir a la pàgina principal 3.2. Altrament 3.2.1. Mostrar missatge d'error 3.3. FSi 4. Altrament 4.1. Mostrar missatge d'error 5. FSi
Postcondició	L'usuari està identificat en el sistema o bé s'ha mostrat un missatge d'error.

- Recuperar contrasenya usuari registrat

CAS D'ÚS:	**Recuperar contrasenya usuari registrat (versió anàlisi)**
Descripció	Escenari principal del cas d'ús: Recuperar contrasenya d'un usuari registrat
Actors	Usuari no registrat
Funcionalitat	Recuperar la contrasenya d'un usuari
Precondició	Cert
Flux principal	1. Entrar email de l'Usuari 2. Buscar Usuari per email 3. Si existeix un Usuari amb l'email entrat 3.1. Enviar email amb link de recuperació 3.2. Accedir al link de recuperació 3.3. Entrar nova contrasenya 3.4. Mostrar missatge de confirmació 4. Altrament 4.1. Mostrar missatge d'error 5. FSi
Postcondició	La contrasenya de l'usuari ha canviat o bé s'ha mostrat un missatge d'error.

- Registrar-se

CAS D'ÚS:	Registrar-se (versió anàlisi)
Descripció	Escenari principal del cas d'ús: registrar nou usuari
Actors	Usuari no registrat
Funcionalitat	Crear un nou usuari en el sistema
Precondició	Cert
Flux principal	1. Entrar nom d'usuari, email i contrasenya 2. Si existeix un Usuari amb l'email o amb el nom d'usuari entrat 2.1. Crear Usuari 2.2. Enviar email de confirmació 2.3. Afegir Usuari a RelacioUsuaris 2.4. Redirigir a la pàgina de login 3. Altrament 3.1. Mostrar missatge d'error 4. FSi
Postcondició	Usuari creat o bé s'ha mostrat un missatge d'error.

- Consultar informació usuaris

CAS D'ÚS:	Consultar informació usuari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: consultar la informació usuari
Actors	Usuari no registrat
Funcionalitat	Visualitzar la informació pública de l'usuari
Precondició	Cert
Flux principal	1. Entrar nom d'usuari 2. Buscar Usuari per nom d'usuari 3. Seleccionar Receptes públiques de l'Usuari 4. Mostrar Usuari i Receptes seleccionades
Postcondició	Informació de l'usuari mostrada

CAS D'ÚS:	Consultar informació usuari (versió anàlisi)
Descripció	Escenari secundari del cas d'ús: consultar la informació usuari
Actors	Usuari registrat, moderador, administrador
Funcionalitat	Visualitzar la informació pública i privada de l'usuari
Precondició	Cert
Flux principal	1. Entrar nom d'usuari 2. Buscar Usuari per nom d'usuari 3. Si l'usuari és l'Usuari trobat, o bé l'usuari és moderador o administrador 3.1. Seleccionar totes les Receptes de l'Usuari 4. Altrament 4.1. Seleccionar Receptes públiques de l'Usuari 5. Fsi 6. Mostrar Usuari i Receptes seleccionades
Postcondició	Informació de l'usuari mostrada

- Mantenir receptes pròpies

CAS D'ÚS:	Baixa recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Donar de baixa una recepta
Actors	Usuari registrat
Funcionalitat	Eliminar una recepta pròpia
Precondició	Recepta creada per l'usuari actual
Flux principal	1. Seleccionar Receptes de l'Usuari 2. Escollir Recepta 3. Escollir l'opció d'eliminar la Recepta 4. Eliminar Recepta 5. Eliminar planificacions de la recepta
Postcondició	Recepta i tot el que té relacionat eliminats

CAS D'ÚS:	Eliminar notificacions de la recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar les notificacions dels usuaris sobre la recepta
Actors	Usuari registrat
Funcionalitat	Eliminar les notificacions dels usuaris que estiguin relacionades amb la recepta a eliminar.
Precondició	Recepta serà eliminada
Flux principal	1. Seleccionar totes les notificacions de la Recepta 2. Eliminar Notificacions seleccionades 3. Actualitzar les seccions de notificacions dels usuaris
Postcondició	Notificacions actualitzades

CAS D'ÚS:	Notificar contactes (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Notificar usuaris contactes de l'autor de la recepta de que una nova recepta està disponible.
Actors	Usuari registrat
Funcionalitat	Generar les notificacions als seguidors de l'autor de la recepta
Precondició	Recepta creada o modificada amb visibilitat pública
Flux principal	1. Obtenir autor de la Recepta 2. Seleccionar tots els usuaris seguidors de l'Autor 3. Per cada usuari seleccionat a. Crear una Notificació amb l'autor, l'usuari seleccionat i la Recepta b. Guardar la Notificació a la llista de Notificacions de l'usuari seleccionat 4. Fper 5. Actualitzar les seccions de notificacions dels usuaris
Postcondició	Usuaris seguidors de l'autor notificats

CAS D'ÚS:	Modificar recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Modificar una recepta
Actors	Usuari registrat
Funcionalitat	Modificar una recepta pròpia
Precondició	Recepta creada per l'usuari actual
Flux principal	1. Seleccionar Receptes de l'Usuari 2. Escollir Recepta 3. Modificar les dades de la Recepta(informació de la recepta, ingredients, estris, passos) 4. Assignar informació de la recepta entrada a la Recepta 5. Per cada Ingredient eliminat de la recepta 5.1. Eliminar ingredient de la llista d'Ingredients de la Recepta 6. FPer 7. Per cada Estri eliminat de la recepta 7.1. Eliminar Estri de la llista d'Estris de la Recepta 8. FPer 9. Per cada Pas eliminat de la recepta 9.1. Eliminar Pas de la llista de Passos de la Recepta 10. FPer 11. Per cada Ingredient modificat de la recepta 11.1. Actualitzar ingredient de la llista d'Ingredients de la Recepta 12. FPer 13. Per cada Pas modificat de la recepta 13.1. Actualitzar pas de la llista de Passos de la Recepta 14. FPer 15. 16. Per cada Ingredient afegit 16.1. Si no existeix 16.1.1. Donar alta ingredient 16.2. Fsi 16.3. Afegir Ingredient dins la llista d'Ingredients de la Recepta 17. FPer 18. Per cada Estri afegit 18.1. Si no existeix 18.1.1. Donar alta estri de cuina 18.2. Fsi 18.3. Afegir Estri dins la llista d'Estris de la Recepta 19. FPer 20. Per cada Pas afegit 20.1. Crear Pas 20.2. Afegir Pas dins la llista de Passos de la Recepta 21. FPer
Postcondició	Recepta modificada

CAS D'ÚS:	Alta recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Crear una recepta
Actors	Usuari registrat
Funcionalitat	Crear una recepta
Precondició	Cert
Flux principal	1. Entrar totes les dades relacionades amb la recepta(informació de la recepta, ingredients, estris, passos) 2. Crear llista Ingredients buida 3. Crear llista Estris buida 4. Crear llista Passos buida 5. Assignar informació de la recepta entrada a la Recepta 6. Per cada Ingredient entrat 6.1. Si no existeix 6.1.1. Donar alta ingredient 6.2. Fsi 6.3. Afegir Ingredient dins la llista d'Ingredients de la Recepta 7. FPer 8. Per cada Estri entrat 8.1. Si no existeix 8.1.1. Donar alta estri de cuina 8.2. Fsi 8.3. Afegir Estri dins la llista d'Estris de la Recepta 9. FPer 10. Per cada Pas entrat 10.1. Crear Pas 10.2. Afegir Pas dins la llista de Passos de la Recepta 11. FPer 12. Afegir Recepta a la llista de receptes d'Usuari 13. Afegir Recepta a RelacioReceptes
Postcondició	Recepta creada

- Guardar receptes

CAS D'ÚS:	Guardar recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: guardar una recepta a la llista
Actors	Usuari registrat
Funcionalitat	Guarda una recepta pública a la llista de l'usuari
Precondició	Cert
Flux principal	1. Seleccionar Receptes públiques 2. Escollir Recepta 3. Escollir l'opció de guardar la recepta 4. Afegir Recepta a la llista de Receptes guardades de l'Usuari
Postcondició	Recepta guardada

- Mantenir informació usuari

CAS D'ÚS:	Donar baixa Usuari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Donar-se de baixa com a usuari
Actors	Usuari registrat
Funcionalitat	L'usuari s'elimina del sistema
Precondició	L'usuari a eliminar és l'actual
Flux principal	1. Escollir l'opció d'eliminar usuari 2. Seleccionar les Receptes de l'usuari 3. Per cada Recepta seleccionada 3.1. Eliminar Recepta de la llista de Receptes de l'Usuari 3.2. Eliminar Recepta de la llista de Receptes Guardades dels Usuaris 3.3. Eliminar Recepta de la RelacioReceptes 4. FPer 5. Eliminar Usuari de la llista de contactes dels Usuaris 6. Eliminar Notificacions que contenen l'usuari a eliminar de les llistes dels usuaris. 7. Eliminar Usuari de la RelacioUsuaris
Postcondició	Usuari eliminat

CAS D'ÚS:	Modificar Usuari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Modificar la informació de l'usuari
Actors	Usuari registrat
Funcionalitat	Les dades de l'usuari es modifiquen
Precondició	L'usuari a modificar és l'actual
Flux principal	1. Escollir l'opció de modificar l'Usuari 2. Obtenir dades de l'Usuari 3. Mostrar les dades de l'Usuari 4. Entrar dades a modificar de l'Usuari 5. Assignar les dades entrades a l'Usuari
Postcondició	Usuari modificat

- Mantenir planificacions receptes

CAS D'ÚS:	Alta planificació Recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Crear una planificació d'una recepta
Actors	Usuari registrat
Funcionalitat	Crear una planificació d'una recepta per l'usuari actual
Precondició	La recepta a planificar és visible per l'Usuari
Flux principal	1. Seleccionem les Receptes visibles per l'Usuari 2. Escollir Recepta a planificar 3. Entrar DataIni, DataFi, Porcions 4. Crear Planificació amb la Recepta i les dades entrades 5. Afegir Planificació a la llista de Planificacions de l'Usuari
Postcondició	Planificació creada

CAS D'ÚS:	Baixa planificació Recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar planificació d'una recepta
Actors	Usuari registrat
Funcionalitat	Eliminar una planificació d'una recepta per l'usuari actual
Precondició	La planificació ha estat creada, no hi ha cap llista de la compra bloquejant la planificació
Flux principal	1. Seleccionem les Planificacions de l'Usuari 2. Escollir Planificació a eliminar 3. Eliminar Planificació escollida
Postcondició	Planificació eliminada

- Registrar temps de preparació d'una recepta

CAS D'ÚS:	Registrar temps de preparació d'una recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: L'usuari registra el temps per preparar la recepta
Actors	Usuari registrat
Funcionalitat	Guardar el temps emprat en dur a terme la recepta
Precondició	Recepta ja escollida
Flux principal	1. Iniciar el cronòmetre al començar la recepta 2. Parar el cronòmetre al acabar-la 3. Escollir la opció de guardar el temps 4. Afegir temps enviat a la llista de preparacions de la Recepta
Postcondició	Temps guardat

- Mantenir llistes de compra

CAS D'ÚS:	**Alta Llista Compra (versió anàlisi)**
Descripció	Escenari principal del cas d'ús: Crear llista compra
Actors	Usuari registrat
Funcionalitat	Crear una llista de la compra per l'usuari actual
Precondició	Cert
Flux principal	1. Crear llistaCompra buida 2. Entrar dataIni, dataFi 3. Seleccionar llistesCompra de l'usuari 4. Si existeix una llista no completada entre les dates 4.1. Mostrar missatge d'error 5. Altrament 5.1. Seleccionar Planificacions entre dataIni i dataFi 5.2. Per cada Planificacio seleccionada 5.2.1. Obtenir Recepta de la Planificacio seleccionada 5.2.2. Seleccionar ingredients de la Recepta 5.2.3. Per cada Ingredient seleccionat 5.2.3.1. Si l'Ingredient existeix a la llistaCompra 5.2.3.1.1. Sumar quantitat de l'Ingredient dins la quantitat de la llistaCompra 5.2.3.2. Altrament 5.2.3.2.1. Afegir Ingredient dins la llistaCompra 5.2.3.3. Fsi 5.2.4. FPer 5.2.5. Bloquejar Planificació 5.3. FPer 5.4. Mostrar llistaCompra 6. Fsi
Postcondició	Llista creada

CAS D'ÚS:	Baixa Llista Compra (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar llista compra
Actors	Usuari registrat
Funcionalitat	Eliminar una llista de la compra per l'usuari actual
Precondició	Llista creada
Flux principal	1. Seleccionar llistesCompra de l'Usuari 2. Escollir llistaCompra 3. Desbloquejar Planificacions de la llistaCompra 4. Eliminar llistaCompra
Postcondició	Llista eliminada

CAS D'ÚS:	Marcar Ingredient com a comprat (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Marcar ingredient com a comprat
Actors	Usuari registrat
Funcionalitat	Marcar un Ingredient de la llistaCompra com a comprat
Precondició	Llista creada i amb algun ingredient no comprat
Flux principal	1. Seleccionar llistesCompra de l'Usuari 2. Escollir llistaCompra 3. Seleccionar ingredients de llistaCompra 4. Mostrar Ingredients seleccionats 5. Escollir Ingredient 6. Assignar comprat a l'Ingredient escollit 7. Si tots els ingredients seleccionats estan comprats 7.1. Mostrar opció de completar llista 8. Altrament 8.1. Mostrar opció d'eliminar llista 9. FSi
Postcondició	Ingredient marcat com a comprat

CAS D'ÚS:	Completar llista (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Marcar llista com a completada
Actors	Usuari registrat
Funcionalitat	Marcar una llista com a completada
Precondició	Cert
Flux principal	10. Seleccionar llistesCompra de l'Usuari 11. Escollir llistaCompra 12. Seleccionar ingredients de llistaCompra 13. Si tots els ingredients seleccionats estan marcats com a comprats 13.1. Mostrar opció Completar llista 13.2. Escollir opció Completar llista 13.3. Marcar la llista com a completada 14. FSi
Postcondició	Llista marcada com a completada

- Compartir receptes

CAS D'ÚS:	Compartir receptes (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Compartir recepta
Actors	Usuari registrat
Funcionalitat	Compartir recepta amb un contacte
Precondició	Recepta creada i visible i l'usuari té almenys un contacte
Flux principal	1. Seleccionar Receptes públiques 2. Escollir Recepta 3. Seleccionar Contactes de l'Usuari 4. Escollir Contacte 5. Entrar Missatge 6. Crear ReceptaCompartida amb l'usuari Actual, el Contacte escollit, la Recepta escollida i el missatge entrat 7. Notificar al Contacte
Postcondició	Recepta compartida i usuari notificat

- Valorar receptes

CAS D'ÚS:	**Valorar receptes (versió anàlisi)**
Descripció	Escenari principal del cas d'ús: Valorar una recepta
Actors	Usuari registrat
Funcionalitat	Valorar una recepta
Precondició	Recepta creada
Flux principal	1. Seleccionar Receptes públiques 2. Escollir Recepta 3. Mostrar Recepta 4. Escollir opció per valorar 5. Si existia una Valoració de l'usuari 5.1. Obtenir Valoració existent 5.2. Si la Valoració existent es positiva i l'escollida és negativa o bé la existent es negativa i l'escollida és positiva 5.2.1. Eliminar Valoració existent 5.2.2. Afegir Valoració escollida 5.3. FSi 6. Altrament 6.1. Afegir Valoració escollida 7. FSi 8. Notificar a l'Autor de la Recepta
Postcondició	Recepta valorada i autor notificat

- Comentar receptes

CAS D'ÚS:	Comentar receptes (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Comentar una recepta
Actors	Usuari registrat
Funcionalitat	Comentar una recepta
Precondició	Recepta creada
Flux principal	1. Seleccionar Receptes públiques 2. Escollir Recepta 3. Mostrar Recepta 4. Entrar text 5. Crear Comentari a partir de Recepta, Usuari i text entrat 6. Afegir Comentari a llistaComentaris de Recepta 7. Notificar a l'Autor de la Recepta
Postcondició	Recepta comentada i autor notificat

- Mantenir contactes

CAS D'ÚS:	Alta contacte (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Crear un contacte
Actors	Usuari registrat
Funcionalitat	Crear un contacte
Precondició	Usuari creat
Flux principal	1. Entrar nom d'usuari o escollir una Recepta de l'usuari desitjat 2. Escollir opció d'afegir contacte 3. Crear Contacte amb l'usuari desitjat 4. Afegir Contacte a la llista de Contactes de l'usuari 5. Notificar l'usuari desitjat
Postcondició	Usuari afegit com a contacte i notificat

CAS D'ÚS:	Baixa contacte (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar un contacte
Actors	Usuari registrat
Funcionalitat	Eliminar un contacte
Precondició	Contacte afegit
Flux principal	1. Entrar nom d'usuari o escollir una Recepta de l'usuari desitjat 2. Escollir l'opció d'eliminar contacte 3. Eliminar Contacte a la llista de Contactes de l'usuari
Postcondició	Usuari eliminat com a contacte

- Denunciar receptes

CAS D'ÚS:	Denunciar receptes (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Denunciar una recepta
Actors	Usuari registrat
Funcionalitat	Denunciar una recepta
Precondició	Recepta creada
Flux principal	1. Seleccionar Receptes públiques 2. Escollir Recepta 3. Mostrar Recepta 4. Escollir opció per denunciar Recepta 5. Crear Denuncia a partir de Recepta i Usuari 6. Notificar als moderadors i administradors del sistema
Postcondició	Recepta denunciada i moderadors notificats

- Moderar comentaris

CAS D'ÚS:	Amagar comentari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Oculta un comentari
Actors	Moderador
Funcionalitat	Oculta un comentari
Precondició	Comentari creat i visible
Flux principal	1. Seleccionar Receptes visibles 2. Escollir Recepta 3. Seleccionar Ingredients, Estris, Passos, Comentaris de la Recepta 4. Mostrar Recepta, Ingredients, Estris, Passos, Comentaris seleccionats 5. Escollir Comentari 6. Escollir opció d'amagar el comentari 7. Assignar visibilitat a fals al Comentari escollit 8. Notificar a l'autor del Comentari
Postcondició	Comentari ocultat

CAS D'ÚS:	Eliminar comentari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar un comentari
Actors	Moderador
Funcionalitat	Eliminar un comentari
Precondició	Comentari creat
Flux principal	1. Seleccionar Receptes visibles 2. Escollir Recepta 3. Seleccionar Ingredients, Estris, Passos, Comentaris de la Recepta 4. Mostrar Recepta, Ingredients, Estris, Passos, Comentaris seleccionats 5. Escollir Comentari 6. Escollir l'opció d'eliminar el comentari 7. Eliminar Comentari de la llista de Comentaris de la Recepta 8. Notificar a l'autor del Comentari 9. Eliminar Notificacions del comentari de l'autor de la recepta
Postcondició	Comentari eliminat

- Moderar receptes

CAS D'ÚS:	Ocultar recepta(versió anàlisi)
Descripció	Escenari principal del cas d'ús: Oculta una recepta
Actors	Moderador
Funcionalitat	Oculta una recepta
Precondició	Recepta creada i visible
Flux principal	1. Seleccionar Receptes visibles 2. Escollir Recepta 3. Escollir opció d'amagar la recepta 4. Assignar visibilitat a fals a la recepta escollida 5. Seleccionar Usuaris que tinguin la Recepta dins la llista de Receptes planejades 6. Per cada Usuari seleccionat a. Eliminar Recepta de la llista de Receptes planejades de l'Usuari 7. FPer 8. Seleccionar Usuaris que tinguin la Recepta dins la llista de Receptes compartides 9. Per cada Usuari seleccionat a. Eliminar Recepta de la llista de Receptes compartides de l'Usuari 10. FPer 11. Seleccionar Usuaris que tinguin la Recepta dins la llista de Receptes guardades 12. Per cada Usuari seleccionat a. Eliminar Recepta de la llista de Receptes guardades de l'Usuari 13. FPer 14. 15. Notificar a l'autor de la Recepta
Postcondició	Recepta oculta

CAS D'ÚS:	Eliminar recepta (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar una recepta
Actors	Moderador
Funcionalitat	Eliminar una recepta
Precondició	Recepta creada
Flux principal	1. Seleccionar Receptes visibles 2. Escollir Recepta 3. Escollir l'opció d'eliminar la Recepta 4. Eliminar Recepta de la llista de Receptes de l'Usuari 5. Seleccionar Usuaris que tinguin la Recepta dins la llista de Receptes planejades 6. Per cada Usuari seleccionat 6.1. Eliminar Recepta de la llista de Receptes planejades de l'Usuari 7. FPer 8. Seleccionar Usuaris que tinguin la Recepta dins la llista de Receptes compartides 9. Per cada Usuari seleccionat 9.1. Eliminar Recepta de la llista de Receptes compartides de l'Usuari 10. FPer 11. Seleccionar Usuaris que tinguin la Recepta dins la llista de Receptes guardades 12. Per cada Usuari seleccionat 12.1. Eliminar Recepta de la llista de Receptes guardades de l'Usuari 13. FPer 14. Eliminar Notificacions sobre la recepta dels usuaris 15. Eliminar Recepta de la RelacioReceptes 16. Notificar a l'autor de la Recepta
Postcondició	Recepta eliminada

- Mantenir ingredients

CAS D'ÚS:	Donar Baixa ingredient (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar un ingredient
Actors	Moderador
Funcionalitat	Eliminar un ingredient
Precondició	Ingredient creat
Flux principal	1. Seleccionar Ingredients 2. Escollir Ingredient 3. Escollir l'opció d'eliminar l'Ingredient 4. Seleccionar Receptes que tinguin l'Ingredient dins la llista de d'Ingredients 5. Per cada Recepta seleccionat 5.1. Eliminar Ingredient de la llista d'Ingredients de la Recepta 6. FPer 7. Seleccionar Usuaris amb llistes Compra que tinguin l'Ingredient 8. Per cada Usuari seleccionat 8.1. Obtenir la llista de Compra 8.2. Eliminar Ingredient de la llista de Compra 9. FPer 10. Eliminar Ingredient de la RelacioIngredients
Postcondició	Ingredient eliminat

CAS D'ÚS:	Modificar ingredient (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Modificar la informació d'un ingredient
Actors	Moderador
Funcionalitat	Modificar un ingredient
Precondició	Ingredient creat
Flux principal	1. Seleccionar Ingredients 2. Escollir Ingredient 3. Escollir l'opció de modificar l'Ingredient 4. Entrar informació a actualitzar 5. Assignar informació entrada a l'Ingredient
Postcondició	Ingredient actualitzat

CAS D'ÚS:	Ajuntar ingredients (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Ajuntar diversos ingredients degut a la seva similitud
Actors	Moderador
Funcionalitat	Ajuntar diversos ingredients
Precondició	Ingredients creats
Flux principal	1. Seleccionar Ingredients 2. Escollir Ingredients 3. Escollir l'opció d'ajuntar Ingredients 4. Escollir l'Ingredient a mantenir 5. Seleccionar Receptes que tinguin algun dels Ingredients que no es mantingui dins la llista d'Ingredients 6. Per cada Recepta seleccionat 6.1. Si l'Ingredient a mantenir és a la llista 6.1.1. Sumar les quantitats dels Ingredients a eliminar a l'Ingredient a mantenir 6.2. Altrament 6.2.1. Crear nou Ingredient amb les quantitats de la resta d'ingredients sumades 6.2.2. Afegir a la llista d'Ingredients de la Recepta 6.3. FSi 6.4. Eliminar Ingredients de la llista d'Ingredients de la Recepta 7. FPer 8. Seleccionar Usuaris amb llistes Compra que tinguin algun dels ingredients que no es mantinguin 9. Per cada Usuari seleccionat 9.1. Obtenir la llista de Compra 9.2. Si l'Ingredient a mantenir és a la llista 9.2.1. Sumar les quantitats dels Ingredients a eliminar a l'Ingredient a mantenir 9.3. Altrament 9.3.1. Crear nou Ingredient amb les quantitats de la resta d'ingredients sumades 9.3.2. Afegir a la llista d'Ingredients de Compra 9.4. FSi 9.5. Eliminar Ingredients de la llista d'Ingredients de la llista de Compra 9.6. 10. FPer 11. Eliminar Ingredients de la RelacioIngredients
Postcondició	Ingredients ajuntats

CAS D'ÚS:	Donar alta unitat per un ingredient (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Afegir unitats a un ingredient
Actors	Moderador
Funcionalitat	Afegir unitats a un ingredient
Precondició	Ingredient creat i unitats no afegides a l'ingredient
Flux principal	1. Seleccionar Ingredients 2. Escollir Ingredient 3. Escollir l'opció de modificar l'Ingredient 4. Seleccionar Unitats 5. Mostrar Unitats seleccionades 6. Escollir les Unitats a afegir 7. Afegir les Unitats escollides a la llista d'Unitats de l'Ingredient
Postcondició	Unitats afegides a l'Ingredient

CAS D'ÚS:	Donar baixa unitat per un ingredient (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar unitats d'un ingredient
Actors	Moderador
Funcionalitat	Eliminar unitats a un ingredient
Precondició	Ingredient creat i unitats afegides a l'ingredient
Flux principal	1. Seleccionar Ingredients 2. Escollir Ingredient 3. Escollir l'opció de modificar l'Ingredient 4. Seleccionar Unitats 5. Mostrar Unitats seleccionades 6. Escollir les Unitats a eliminar 7. Eliminar les Unitats escollides a la llista d'Unitats de l'Ingredient 8. Eliminar de les Receptes aquells ingredients amb la Unitat a eliminar
Postcondició	Unitats eliminades a l'Ingredient

- Mantenir estris de cuina
  Pels estris de cuina, els casos d'ús només es diferencien respecte els dels ingredients amb les unitats i les llistes de compra.

CAS D'ÚS:	**Ajuntar estris de cuina (versió anàlisi)**
Descripció	Escenari principal del cas d'ús: Ajuntar diversos estris de cuina degut a la seva similitud
Actors	Moderador
Funcionalitat	Ajuntar diversos estris de cuina
Precondició	Estris creats
Flux principal	1. Seleccionar Estris de cuina 2. Escollir Estris de cuina 3. Escollir l'opció d'ajuntar Estris de cuina 4. Escollir l'Estri de cuina a mantenir 5. Seleccionar Receptes que tinguin algun dels Estris de cuina que no es mantingui dins la llista d'Ingredients 6. Per cada Recepta seleccionat 6.1. Si l'Estri de cuina a mantenir no és a la llista 6.1.1. Afegir Estri de cuina a mantenir a la llista d'Estris de cuina de la Recepta 6.2. FSi 6.3. Eliminar Estris de cuina de la llista d'Estris de cuina de la Recepta 7. FPer 8. Eliminar Ingredients de la RelacioEstris
Postcondició	Estris ajuntats

- Mantenir categories

CAS D'ÚS:	Donar Alta categoria (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Crear una nova Categoria
Actors	Administrador
Funcionalitat	Crea una nova categoria
Precondició	Cert
Flux principal	1. Entrar nom de la Categoria 2. Crear llista buida de Receptes 3. Crear Categoria amb el nom entrat i la llista de Receptes 4. Afegir Categoria a RelacioCategories
Postcondició	Categoria creada

CAS D'ÚS:	Donar baixa categoria (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar una Categoria
Actors	Administrador
Funcionalitat	Eliminar una categoria
Precondició	Categoria creada
Flux principal	1. Seleccionar Categories 2. Escollir Categoria 3. Escollir opció d'eliminar Categoria 4. Eliminar Categoria de RelacioCategories
Postcondició	Categoria eliminada

CAS D'ÚS:	Modificar categoria (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Modificar una Categoria
Actors	Administrador
Funcionalitat	Modificar una categoria
Precondició	Categoria creada
Flux principal	1. Seleccionar Categories 2. Escollir Categoria 3. Escollir opció de modificar Categoria 4. Entrar informació a actualitzar 5. Assignar informació entrada a la Categoria
Postcondició	Categoria modificada

- Consultar estadístiques generals de la plataforma

CAS D'ÚS:	Consultar estadístiques generals de la plataforma (versió anàlisi)
Descripció	Escenari principal del cas d'ús: TODO
Actors	Administrador
Funcionalitat	Mostra les estadístiques d'ús generals de la plataforma
Precondició	Cert
Flux principal	1. Seleccionar totes les Receptes 2. Agrupar Receptes seleccionades per mes de publicació 3. Mostrar gràfic de les receptes agrupades 4. Contar les visites a Receptes d'avui 5. Mostrar nombre de visites de receptes 6. Obtenir la categoria que conté més visites a partir de les visites de les seves receptes 7. Mostrar la categoria obtinguda 8. Contar plans creats avui 9. Mostrar nombre de plans creats 10. Agrupar les Receptes seleccionades per visibilitat 11. Mostrar gràfic de les receptes agrupades per visibilitat 12. Seleccionar tots els Usuaris 13. Contar els usuaris registrats per mes 14. Mostrar gràfic dels usuaris registrats per mes
Postcondició	Informació mostrada

- Mantenir usuaris

CAS D'ÚS:	Donar baixa Usuari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Eliminar usuari del sistema
Actors	Administrador
Funcionalitat	Elimina un usuari del sistema
Precondició	usuari creat
Flux principal	1. Seleccionar Usuaris 2. Escollir Usuari 3. Escollir l'opció d'eliminar Usuari 4. Seleccionar les Receptes de l'usuari 5. Per cada Recepta seleccionada 5.1. Eliminar Recepta de la llista de Receptes de l'Usuari 5.2. Eliminar Recepta de la llista de Receptes Guardades dels Usuaris 5.3. Eliminar Recepta de la RelacioReceptes 6. FPer 7. Eliminar Usuari de la llista de contactes dels Usuaris 8. Eliminar Usuari de la RelacioUsuaris
Postcondició	Usuari eliminat

CAS D'ÚS:	Modificar Usuari (versió anàlisi)
Descripció	Escenari principal del cas d'ús: Modificar informació d'un usuari
Actors	Administrador
Funcionalitat	Modifica la informació d'un usuari
Precondició	usuari creat
Flux principal	9. Seleccionar Usuaris 10. Escollir Usuari 11. Escollir l'opció d'editar Usuari 12. Entrar informació actualitzada 13. Si el rol de l'usuari no era administrador 13.1. Assignar rol entrat a l'Usuari 14. Fsi 15. Assignar informació actualitzada a l'Usuari excepte el rol
Postcondició	Usuari modificat