Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Robledo Deiros, Alvaro; Benítez Fernández, Yolanda, dir. Desenvolupament d'un Back Office per a Mango. 2021. (958 Enginyeria Informàtica)

This version is available at https://ddd.uab.cat/record/248525

under the terms of the  license

Desenvolupament d'un Back Office per a Mango

Álvaro Robledo

Resum– El Treball de Final de Grau següent consisteix en la creació d'un *Back Office* per a l'empresa de Mango Fa. El desenvolupament d'aquest projecte és educatiu, a més a més, professional perquè és per l'ambit empresarial. Té com a objectiu principal ser escalable, ja que en un futur s'incorporaran noves funcionalitats. Per això, s'utilitzen tècniques de *Clean Code*, com és l'exemple de l'arquitectura hexagonal, i una metodologia de desenvolupament com *Scrum*. L'IDE de desenvolupament és IntelliJ amb un codi font en Reactjs i SpringBoot (Java).

Paraules clau– *Back-end*, *Back Office*, *Clean Code*, desenvolupament de software, *framework*, *Front-end*, gestor, Java, Mango, *marketplace*, metodologia àgil, Reactjs, *Scrum*, SpringBoot.

Abstract– The following Final Degree Project consists of the creation of a Back Office for the company Mango Fa. The development of this project is educational, moreover, professional because it is for the business environment. Its main objective is to be scalable, as new functionalities will be added in the future. For this reason, Clean Code techniques are used, such as the hexagonal architecture, and a development methodology such as Scrum. The development IDE is IntelliJ with a source code in Reactjs and SpringBoot (Java).

Keywords– Agile methodology, Back-end, Back Office, Clean Code, framework, Front-end, Java, manager, Mango, marketplace, Reactjs, Scrum, software development, SpringBoot.

1 Introducció

MANGO FA és una empresa que es dedica a la venta de productes de roba. Compta amb diverses tendes físiques, i diversos convenis amb altres *marketplaces* per poder vendre els seus productes a la web d'aquests, i així poder augmentar el número de ventes i beneficis.

Per poder publicar els productes de Mango en aquests *marketplaces*, el que es fa és adaptar la informació d'aquests, per a que aquestes empreses els arribi la informació com desitgen i els productes es publiquin amb èxit a la pàgina web corresponent.

Però, la informació que s'envia a aquests, necessita un manteniment, ja que pot estar errònia o en un format no desitjat un cop enviada.

El flux per corregir aquests errors és el que s'indica a la *Figura 1*. Hi ha un equip anomenat "Business", i un altre "Marketplaces", amb una part destinada al suport i manteniment d'incidències.

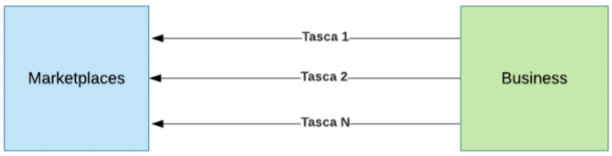

Fig. 1: Flux de treball actual per corregir errors

Molta de la informació que s'envia a aquests *marketplaces*, està emmagatzemada a la base de dades, on tota la informació està adaptada per poder enviar correctament aquesta a través de diversos fitxers JSON.

Hi ha molts casos d'incidències a l'hora de solucionar els errors que apareixen, ja sigui modificant la base de dades o directament fen-t'ho des del codi. Actualment, aquestes incidències que es resolen a través de la base de dades, es fa llençant les consultes SQL corresponents de manera manual.

La solució que es proposa per a que aquest procediment sigui més ràpid, es basa en el desenvolupament d'un Back Office des de cero, per solucionar aquestes problemàtiques que sorgeixen a l'hora de poder corregir informació que s'envia a aquests *marketplaces*. A més a més, el flux de treball canviarà, fet que l'equip de "Marketplaces" deixarà de rebre incidències per part de "Business", i per tant, també

• E-mail de contacte: alvaro.robledo@mango.com
• Menció realitzada: Enginyeria del Software
• Treball tutoritzat per: Yolanda Benítez (Ciències de la Computació)
• Curs 2020/21

serà més independent i no dependrà de "Marketplaces" a l'hora de fer la correcció.

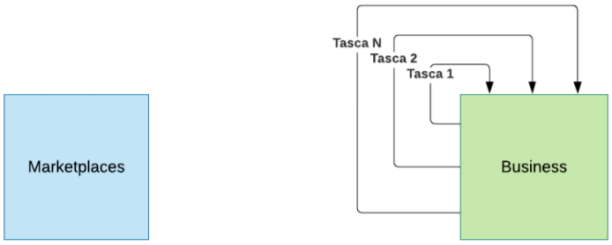

Fig. 2: Flux de treball futur per corregir errors

A part de desenvolupar la base que ocupa aquest Back Office, també es vol incorporar un gestor que es basa en la tasca de poder mapejar/corregir productes, a la mateixa base de dades, quan hi hagi una manca dinformació o sigui errònia. Es dona la casuística de que si aquesta infomació no és correcta, aquests productes no seran publicats en el seu respectiu *marketplace*.

2 Estat de l'art

El projecte que es desenvolupa per a l'empresa Mango, té una funcionalitat bastant específica, i per tant, és un projecte intern i molt concret per resoldre una necessitat pel departament Online. En el mercat avui dia no hi ha cap altre BackOffice que serveixi per solucionar problemes interns a Mango, però si que hi ha d'altres amb funcionalitats semblants amb altre tipus d'objectius.

2.1 SAP

Systems, Applications, Products in Data Processing, o més conegut com SAP, funciona pràcticament en tots els àmbits de l'administració empresarial. És un sistema informàtic que fa que les empreses puguin administrar correctament els seus recursos, per tant, es podria relacionar amb un sistema ERP, fet que gestiona les diferents accions d'una empresa.

SAP té tres característiques fonamentals que el diferencien d'altres ERPs, com per exemple, es pot definir com un sistema a mesura, on satisfà les necessitats del client d'una forma més positiva. També es pot considerar un sistema encapsulat, on engloba els sistemes dins de les empreses que desenvolupen software i soluciona les necessitats del mercat actual a un nivell més ampli. Finalment, és un sistema de codi obert, on facilita una major rapidesa i que el client pugui modificar el codi donant a l'empresa bons beneficis.[1]

2.2 Oracle Netsuite

Netsuite és una empresa fundada a l'any 1988, on oferia software de comptabilitat emmagatzemat en la web. Avui dia, aquest software està dissenyat per racionalitzar els processos crítics, on permet a les empreses centrar-se en el que fan millor i reaccionar a les noves oportunitats del mercat amb rapidesa i confiança. Netsuite ERP, ofereix a les empreses una clara visibilitat i control del seu negoci, començant des de la cadena de subministrament, i passant per la facturació. [2]

Algunes de les característiques més potenciades d'Oracle Netsuite són:

- La gestió financera: Es combina una gestió financera sòlida amb una intel·ligència empresarial integrada per impulsar la presa de decisions.
- La planificació financera: Potencia el procés de planificació amb una solució intuïtiva de planificació, pressupost i previsió.
- La gestió de comandes: Accelera el procés de comanda a efectiu, vinculant vendes, finançament i compliment a preus.
- La direcció de producció: Aprofita la visibilitat en temps real dels processos de gestió de la producció.
- La gestió de la cadena de subministrament: Dona suport als plans de la gestió de la cadena de subministrament des d'una única plataforma col·laborativa.
- El magatzem i compliment: Gestiona l'inventari d'extrem a extrem i la logística d'entrada i sortida en temps real.
- La contractació: Millora la precisió dels processos de compra de pagaments i seguretat.

2.3 Sage

Sage és la companyia tecnològica de software més gran del Regne Unit, i el tercer proveïdor més gran del món de software de planificació de recursos empresarials darrere de SAP i Oracle Netsuite. [3]

Aquest ERP serveix tant per empreses petites com per grans corporacions. Sage potencia el creixement i desenvolupament des de l'assessorament expert fins processos més eficients. [4] Alguna de les característiques que incorpora són les següents:

- Optimitzar recursos essencials: S'unificaran operacions quotidianes per maximitzar l'eficiència i reduir costos.
- Simplificar els processos essencials: Facilita el control dels aspectes més importants com la gestió del capital humà, els sistemes CRM i els actius fixes.
- Expansió a nivell nacional e internacional: És una eina ràpida, fàcil i flexible, que té com a objectiu fer créixer el negoci a tot el món.
- Consell d'experts: Ofereix suport i informació sobre el sector per poder planificar i implantar les solucions que millor s'adaptin al negoci.

2.4 Microsoft Dynamics

Microsoft Dynamic és una aplicació de software de planificació de recursos empresarials o ERP, i gestió de relacions amb els clients o CRM. Aquest sistema permet la possibilitat d'analitzar les dades provinents de les interaccions dels clients amb l'equip de màrqueting i ventes, i el departament d'atenció al client, conjuntament amb una gran varietat de funcionalitats. A més a més, és un sistema que està totalment integrat amb Outlook i Microsoft Office.

També ofereix poder identificar les oportunitats, potenciar els punts forts de l'empresa i una presa de decisions més

acurada per poder incrementar la rendibilitat. Altra de les característiques important que ofereix aquesta plataforma, és la utilització de les xarxes socials per ampliar els canals de comunicació i poder arribar a un públic més específic, mitjançant campanyes de segmentació augmentant la tasa de conversió.[5]

2.5 TRYTON

Tryton és una plataforma informàtica ERP desenvolupada amb el llenguatge de programació Python conjuntament amb Javascript, amb una arquitectura d'alt nivell en tres capes; el client, el servidor i la seva base de dades en PostgreSQL i SQLite. El seu propòsit és desenvolupar solucions per negocis a través dels seus mòduls. La primera versió va ser llançada al novembre de l'any 2008. [6]

Els mòduls oficials que la plataforma incorpora són més de 130. Aquest ERP serveix tant per l'administració ventes, compres, inventari, calendari i projectes, entre d'altres. Tant el client i el servidor de l'aplicació estan disponibles en Linux, MacOS i Windows. [7]

3 OBJECTIUS

La intencionalitat de realitzar un projecte de software, és degut a l'interès que suposa millorar una funcionalitat i/o per facilitar i millorar una part de la feina a persones. En aquest cas, es vol aconseguir fer ambdues, fet que és una plataforma que s'utilitzarà com a mínim en els pròxims deu anys.

3.1 PRINCIPALS

Els objectius principals són els més importants i prioritaris. Es consideren essencials per poder avançar tant personalment com professionalment al treball.

- Objectiu nº 1: Millorar el procés de realitzar les diverses tasques principals i rutinàries que l'equip "Marketplaces" realitza. És important remarcar que cada marketplace treballa d'una manera diferent, i pot ser que una tasca no es realitzi amb el mateix procediment.
- Objectiu nº 2: Aprendre i donar èmfasi a la part de *front-end* amb Reactjs. Aquest objectiu és un dels més importants. L'equip està format amb gent que es dedica exclusivament al *back-end*, i totes les webapps que s'utilitzen actualment a l'empresa, estan desenvolupades amb Reactjs.
- Objectiu nº 3: Aprendre i donar èmfasi a la gestió i construcció de projectes amb Gradle. Aquest objectiu també es considera un dels més importants, i està molt relacionat amb l'objectiu nº 3.
- Objectiu nº 4: La possibilitat de desenvolupar un projecte que sigui escalable. Com bé s'ha mencionat, en els propers mesos, s'ha de poder seguir introduint gestors.
- Objectiu nº 5: Aprendre l'arquitectura hexagonal. En l'equip, tots els nous projectes que s'estan desenvolupant fan servir aquesta arquitectura orientada en el *clean code*.

3.2 SECUNDARIS

Els objectius secundaris no són menys importants que els principals, però tenen menys prioritat a l'hora de complir els objectius establerts.

- Objectiu nº 6: L'experiència de l'usuari ha de ser bona a l'hora de fer-ne ús de l'aplicació, és a dir, que sigui fàcil d'utilitzar i molt senzilla.
- Objectiu nº 7: El manteniment de l'aplicació ha de ser fàcil i ràpid. La intenció és que en un futur l'aplicació sigui robusta i gran, i altre de les fites és que el desenvolupament del codi es pugui mantenir degut a l'arquitectura utilitzada.
- Objectiu nº 8: L'aplicació ha de ser *responsive*. El motiu és per que a l'equip hi ha persones que realitzen guàrdies a través d'un telèfon mòbil.

4 METODOLOGIA

Existeixen diverses metodologies per el desenvolupament d'un projecte de software, que s'utilitzen per estructurar, planificar i controlar un procediment a l'hora de realitzar un desenvolupament. Dona una solució a un client, mercat o necessitat, tenint en compta els costos, la qualitat i les dificultats associades.

Per tant, la metodologia emprada en el projecte es basada en *Scrum*. L'objectiu inicial d'aquesta metodologia, es controlar i planificar projectes amb un gran volum de tasques. A més a més, s'utilitza per treballar en equip a partir d'iteracions o *sprints*, on normalment es planifica per setmanes. Al final de cada iteració, las tasques que han estat finalitzades es revisen i corregeixen si es necessari. També es prioritzen i es tornen a planificar activitats per una següent iteració. És una metodologia que també esta focalitzada en la realització de tests, on aquests es desenvolupen de manera progressiva, i no un cop finalitzat tot el desenvolupament.

S'han utilitzat tècniques d'aquesta metodologia per a que el desenvolupament estigui molt ben estructurat i organitzat, sobretot que tota la feina que s'ha realitzat quedi registrada i documentada. Les tècniques utilitzades han estat les següents:

- *Daily meetings:* La tècnica consisteix en realitzar diàriament una petita reunió de seguiment per saber com el projecte està d'avançat, sense donar gaire detalls.
- *Refinements:* La següent també és bastant semblant a un *daily meeting*, però la diferència és que aquesta reunió té un nivell més tècnic i detallada, fins a la possibilitat d'obrir debats dins de l'equip.
- *Roadmap:* Consisteix en realitzar una reunió un cop al més, on es parla sobre tots els projectes que hi ha en desenvolupament. És important saber el progrés d'aquests. També hi ha la possibilitat de realitzar replanificacions, depenent de la situació actual.
- *Board amb Jira:* Aquesta eina s'ha utilitzat per tenir una visualització de quines són les tasques a desenvolupar, quines estan en progrés, en test i finalitzades.

- *Repositori amb Bitbucket:* Una segona eina per al desenvolupament àgil pot estar la utilització d'un repositori. En aquesta ocasió, s'ha utilitzar Bitbucket per a tenir un control de versions de l'aplicació.
- *Miro:* També s'ha utilitzat altres eines com Miro, molt útil per prioritzar els gestors que es volen incorporar a Mambo.

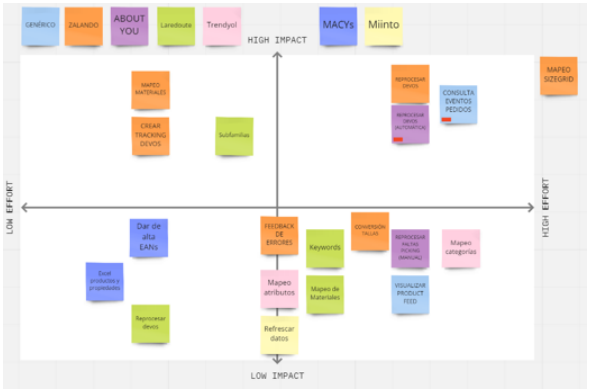

Fig. 3: Taulell de prioritats sobre els gestors

5 Planificació

En aquesta secció s'adjunta una taula de la planificació del projecte en l'apèndix. [A1]

6 Anàlisi

Per a un correcte disseny del software, cal realitzar prèviament una sèrie d'anàlisis anteriors, per poder complir amb els objectius establerts a l'inici del projecte. Altra punt molt important, és la planificació i definició del desenvolupament. En les seccions de requeriments funcionals i no funcionals, s'utilitzaran les següents assumpcions:

Prioritat	Verificació
A: Alta	R: Revisió
M: Mitjana	T: Test
B: Baixa	D: Disseny

6.1 Diagrama de casos d'ús

A l'apèndix hi ha adjunt el diagrama de casos d'ús. [A2]

Per començar, es pot observar com el diagrama té un actor principal a la part del *front-end* i un altre a la part del *back-end*.

L'actor usuari té la possibilitat de realitzar diverses tasques, com introduïr una contrasenya i un e-mail per fer *login*, i conseqüentment, fer *logout*. Pot escollir entre tornar a la pàgina principal de l'aplicació, o tornar enrere (navegació entre pàgines visitades). Per poder accedir als diferents gestors, haurà de seleccionar una opció del menú, i posteriorment del submenú. Finalment, parlant a trets del gestor desenvolupat, podrà seleccionar un fitxer i després enviar-lo.

L'actor aplicació serà l'encarregat de validar les dades i, posteriorment, enviar feedback un cop el fitxer ha estat rebut. Si aquestes dades són vàlides, les introduirà a la base de dades. Finalment, també serà l'encarregat d'enviar les dades del menú cap al *front-end*.

A continuació, es mostrarà un exemple de cas d'ús:

Propietat	Descripció
Id. Cas d'ús	UC-01
Nom o Títol	*Login*
Pre-condicions	- L'usuari ha d'entrar a través de la url a la pàgina de *login* de l'aplicació. - L'usuari no ha de tenir cap sessió iniciada o ha d'haver fet *logout*.
Post-condicions	- L'usuari ha de poder visualitzar l'aplicació un cop s'hagin validat les credencials.
Escenari principal	1. L'usuari introdueix el seu correu electrònic. 2. L'usuari introdueix la seva contrasenya. 3. L'usuari clica sobre el botó de *login*. 4A. L'usuari visualitza la pàgina principal de l'aplicació.
Escenari(s) alternatiu(s)	4B. El sistema retorna un missatge d'error, comprovant que les credencials no són correctes.
Requisits addicionals	- L'usuari ha de tenir un compte a Mango. - L'usuari ha d'estar connectat a la VPN.

6.2 Requeriments funcionals

Els requeriments funcionals que s'expliquen, són els corresponents a la base del projecte, conjuntament amb l'únic gestor desenvolupat, "Mapeig de size-grids". Aquests requeriments es troben a l'apèndix. [A3]

6.2.1 Usuaris

En aquests requeriments funcionals, es vol donar una visió de com un usuari de Mango pot interactuar amb el sistema, i realitzar les accions necessàries per finalitzar un procés qualsevol. També per a que l'experiència d'usuari sigui bona.

6.3 Requeriments no funcionals

Els requeriments no funcionals que s'expliquen, són els corresponents a la base del projecte, conjuntament amb l'únic gestor desenvolupat, "Mapeig de size-grids". Aquests requeriments es troben a l'apèndix. [A4]

6.3.1 Seguretat

Els requeriments no funcionals de la seguretat, principalment serveixen per donar la informació de com aquesta aplicació desenvolupada pot arribar a protegir-se d'usuaris que no tenen els privilegis ni permisos per utilitzar-la.

6.3.2 Rendiment

Els requeriments no funcionals del rediment de l'aplicació, són per donar un valor extra a l'experiència d'usuari, on el puguin utilitzar diverses persones alhora, i que l'aplicació compleixi amb el rang correcte dels temps de resposta.

6.3.3 Aparença/Usabilitat

En els requeriments no funcionals de la usabilitat també estan molt relacionats amb l'experiència d'usuari, on els estils de l'aplicació segueixen els de Mango.

6.3.4 Manteniment

En els requeriments no funcionals del manteniment estàn relacionats amb les bones pràctiques de programació, per a que en un futur el codi no sigui molt complicat de mantenir ni utilitzar per més desenvolupament.

6.3.5 Disponibilitat

En els requeriments no funcionals de la disponibilitat s'explica i es dona més detall de quins són aquests requisits extra per poder utilitzar l'aplicació.

6.4 Diagrama de BD

Les taules que s'utilitzen a la base de dades són tres. S'utilitzen per a una gestió i utilització correcte per a un menú dinàmic, i l'inserció o actualització de noves dades. A continuació s'explicarà cadascuna d'aquestes:

6.4.1 Zalando_Product_Mapping

Aquesta taula s'utilitza per poder inserir dades actualitzades des de el gestor "Mapeig de size-grids", i recollir també la informació existent per evitar fer updates innecessaris. Com a clau primària hi ha el *product_id*, l'identificador de producte, i com a clau forana la mida d'aquest.

Això implica que aquesta taula necessita informació d'una altra, *Zalando_Shape_Rules*, on s'emmagatzema tot tipus de mides dels productes. Però, el que interessa és el camp de *size_grid*. Un cop s'utilitzi el gestor desenvolupat, serà el lloc on anirà emmagatzemat cadascun dels *size-grids* dels productes.

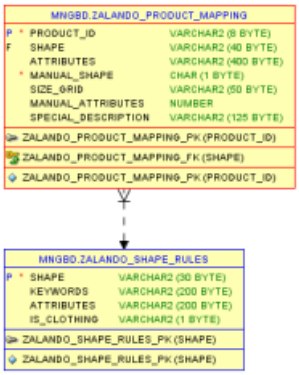

Fig. 4: Taula zalando_product_mapping

6.4.2 Mkt_Mambo_Markets

La següent taula s'utilitza per gestionar el menú dinàmic de l'aplicació, per a que no hi hagi codi *hardcoded* i es pugui gestionar el menú des de la base de dades. La taula emmagatzema un identificador auto-incremental, juntament amb el nom dels diferents *marketplaces* que tindran gestors.

6.4.3 Mkt_Mambo_Managers

Aquesta taula també s'utilitza per gestionar el menú, però a diferència de *Mkt_Mambo_Markets*, en aquesta el que es guarda és el nom de cada gestor desenvolupat, la url que aquest tindrà i un identificador com a clau forana de la taula anterior, per relacionar el gestor amb el seu *marketplace*. També s'emmagatzema un identificador auto-incremental com a clau primària.

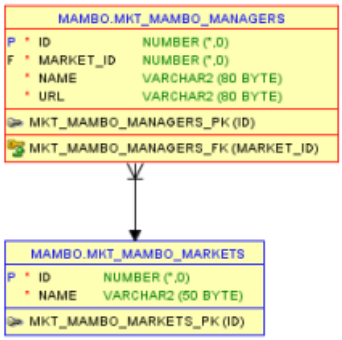

Fig. 5: Taula mkt_mambo_markets i mkt_mambo_managers

6.5 Diagrama de microserveis

Es vol donar una visió més general de les parts que composa aquest projecte orientat als microserveis. Per una part, es pot observar a la Figura 6 com el *front-end* i el *back-end* estan connectats entre si per l'intercanvi d'informació de la base de dades. A més a més, aquest *front-end*, també serà utilitzat per poder connectar-se a *Mkt-login*, el projecte de login per l'equip, que també ha estat desenvolupat per poder incorporar-ho a Mambo, i on també per a que l'intercanvi d'informació per l'autenticació d'un usuari sigui efectiva.

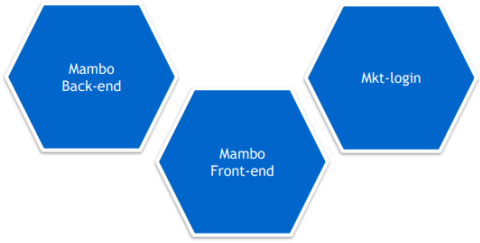

Fig. 6: Diagrama de microserveis

7 Desenvolupament

7.1 Arquitectura Front-end

El codi desenvolupat per la part de *front-end* és Reactjs. El codi utilitza tant Javascript com Html, denominat JSX. Per aquest motiu, l'arquitectura que aquest utilitza és molt semblant a un estil de Model-Vista-Controlador, però més adaptat al que és el llenguatge en sí.

Per tant, es podria afirmar que l'arquitectura es nombrada com "Arquitectura de components", fet que Reactjs es basa en components, o en un estil més vulgar, trossos de la pantalla de l'ordinador que van canviant reactivament. En realitat, el que Reactjs permet, és anar canviant dinàmicament el DOM de l'aplicació.

A la Figura 7 es mostra l'arquitectura dels directoris.

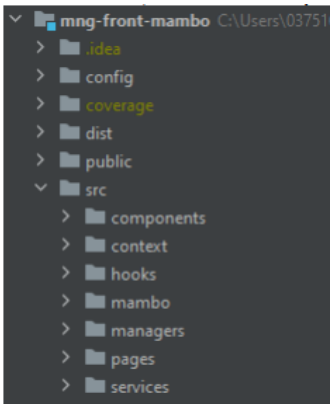

Fig. 7: Arquitectura de directoris front-end

7.1.1 Directori components

En aquest directori és on s'emmagatzemen tots aquells components que es poden reutilitzar en tota l'aplicació, com és l'exemple d'un formulari per enviar un fitxer, un botó, el menú, etc.

7.1.2 Directori context

En el següent directori és on s'emmagatzemen tots els fitxer de configuració per al context de l'aplicació. El context en Reactjs, fa referència a un o diversos estats en el que tots els components tenen accessibilitat a aquest. El funcionament de Reactjs fa que cada component gestioni el seu propi estat, però també permet que un component pugui enviar el seu estat a un altre. No és gaire òptim que, per exemple, des de l'arrel del DOM, es passi un estat a una de les seves fulles, fet que hauria de passar per cadascun dels nodes que hi ha entre mig. Per aquest motiu s'utilitza el context. Un exemple clar d'un estat genèric pot estar si un usuari està logejat o no.

7.1.3 Directori hooks

En aquest directori és on s'emmagatzemen tots els *hooks* de l'aplicació. Un *hook* realment és una API de la llibreria de Reactjs que permet tenir un estat en aquells components, que estan creats com funció, i no com classe. Això abans no era possible, fet que per poder tenir un estat abans en un component, aquest s'havia de crear com classe. També es poden crear *hooks* propis per utilitzar-los en l'aplicació.

7.1.4 Directori mambo

El directori mambo és on s'emmagatzema el fitxer d'*index.js*, conjuntament amb el de l'arrel del DOM. En altres paraules, és el component que encapsula tota l'aplicació, anomenat *App.jsx*.

7.1.5 Directori managers

Aquest directori també podria considerar-se de components, però amb la diferència de que aquests no són reutilitzables. Són components més genèrics, que estan desenvolupats amb altres components més específics. El mateix nom ho diu, i és on aniran totes les pàgines dels gestors a desenvolupar en un futur.

7.1.6 Directori pages

Igual que amb l'anterior directori, aquest també podria considerar-se de components, però a diferència de que són més diversos i no tant concrets com l'anterior, que esta destinat únicament a gestors. Alguns dels exemples podria ser la pàgina principal de l'aplicació, la del *login* o la del *logout*.

7.1.7 Directori services

El directori esta composat per un fitxer on es guarden tots els serveis de l'aplicació. Un servei s'utilitza per poder enviar i rebre informació, és a dir, consumir una API. També fa la connexió entre el *front-end* i *back-end*.

7.2 Arquitectura Back-end

7.2.1 Microservei Mambo

El codi desenvolupat per la part del *back-end* és Springboot, un *framework* de Java, i també s'utilitza el propi Java. A l'equip sorgeix la necessitat d'estructurar el codi d'una manera més òptima, fet que les APIs antigues desenvolupades dels diferents *marketplaces*, són bastant complexes de llegir i entendre. Per aquest motiu, es va anar incorporant l'anomenada "Arquitectura hexagonal", també coneguda com "Arquitectura de ports i adaptadors", però la que s'utilitza pel projecte està adaptada a l'equip.

L'arquitectura té la intenció de que una aplicació sigui utilitzada de la mateixa forma per usuaris, programes o tests, i que sigui testejada d'una forma aïllada.

La principal característica d'aquesta arquitectura és poder separar l'aplicació en tres distintes capes o mòduls, amb la seva pròpia responsabilitat cadascun. D'aquesta manera, el que s'aconsegueix es desacoblar per a que evolucionin d'una manera independent, i si cal, es puguin reutilitzar en altres projectes si és necessari.

La idea general és que cada cara de l'hexàgon representa un port. També és molt important dir que les dependències entre las capes van de dins a fora, on a continuació s'explica amb més detall.

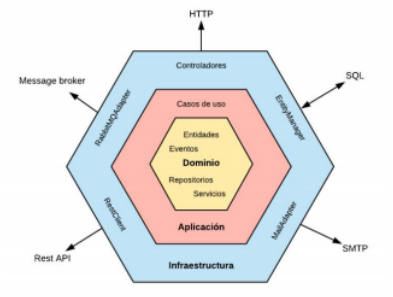

Fig. 8: Esquema de l'arquitectura hexagonal

Per tenir un aproximament al codi, a la Figura 9 es mostra l'arquitectura dels directoris, on es pot comprovar com efectivament, aquestes tres capes estan separades en tres mòduls diferents.

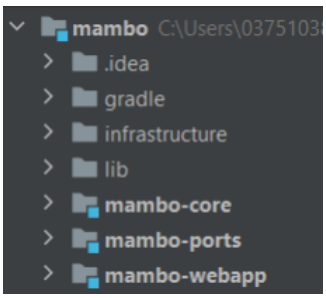

Fig. 9: Arquitectura de directoris del back-end

- **Capa Webapp:** Aquesta capa fa de pont entre una aplicació i un servei necessitat per aquesta. Actua per transformar la comunicació entre actors externs i la lògica de l'aplicació, de forma que ambdues queden independents. Els actors primaris i secundaris interactuen a través d'aquesta. Els actors primaris són aquells que utilitzen el sistema per aconseguir una fita, i els actors secundaris són aquells que el sistema utilitza per aconseguir les fites dels actors primaris.

  Un adaptador podria considerar-se un tros de codi entre l'usuari i el centre de la lògica, on és la funció de la prova unitària per el centre de la lògica. Un altre punt relacionat amb les dependències de l'aplicació, és que aquesta capa les coneix totes, tant les seves, com les de les capes de Ports i Core.

  En el que seria l'aplicació de Mambo, la Webapp és el *front-end* fet amb Reactjs, però a l'equip es crea aquest mòdul a la part de *back-end* per afegir els arxius de configuració i la entrada a l'aplicació. Per aquest motiu, s'ha comentat que aquesta arquitectura estava adaptada per a l'equip.

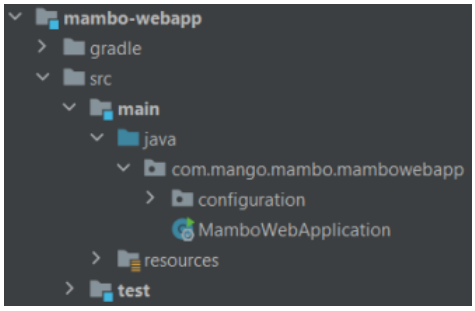

Fig. 10: Mòdul webapp

- **Capa Ports:** Un port es considerat un punt d'entrada, provinent del centre de la lògica que defineix una sèrie de funcions, i on la lògica del negoci és accessible. Majoritàriament són el que és consideren les APIs de l'aplicació, on aquestes són trucades per els adaptadors primaris que formen la part de l'usuari de l'aplicació.

  En canvi, un port secundari són les interfícies per als adaptadors secundaris, i aquests són trucats des de el centre de la lògica. Un altre punt molt important, és que aquesta capa coneix les seves pròpies dependències i les de la capa de Core.

  Com es pot observar a la Figura 11, el que conté aquesta capa són els controladors, els repositoris i els serveis, per poder realitzar aquesta connexió entre l'usuari i l'aplicació. També conté processos relacionats amb la lògica de negoci, com anteriorment s'ha mencionat, i és l'exemple de la validació del format de les dades d'un CSV, o inclús el format en el que s'envia una resposta al *front-end*, entre d'altres.

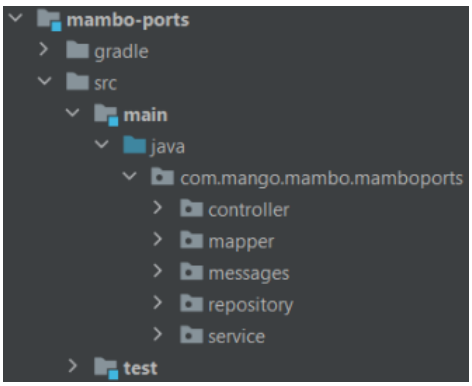

Fig. 11: Mòdul Ports

- **Capa Core:** Aquesta capa representa el model conceptual de l'aplicació, és a dir, és una representació de conceptes significatius que han de ser modelats. Aquesta inclou tota la informació que envolta el negoci i dirigeix els usos de negoci en relació a aquesta informació, on el seu codi està desenvolupat únicament i exclusivament amb Java pur, sense cap *framework*, per poder facilitar l'adaptabilitat en altres projectes.

  A la Figura 12 es veu l'arquitectura de directoris de la capa Core, on es pot observar una carpeta anomenada *model*, que és on van totes aquestes representacions esmentades amb anterioritat, conjuntament amb la carpeta *mapper*, que s'utilitza per representar un objecte amb un format determinat. També inclou un directori *validator* per validar que les dades del CSV no siguin buides, un procediment diferent al que seria la lògica de negoci. Finalment, un directori per representar les excepcions de l'aplicació.

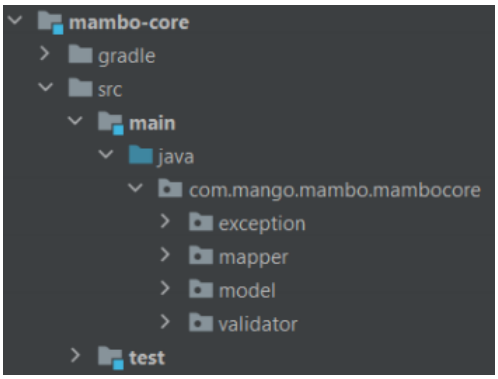

Fig. 12: Mòdul Core

7.2.2 MICROSERVEI LOGIN

Aquest microservei també esta desenvolupat amb el *framework* de programació SpringBoot, propi de Java, corresponent a la part de *back-end*. A sorgit la necessitat de realitzar aquest petit projecte, fet que encara no hi havia res per poder autenticar als integrants de l'equip. Aquest s'utilitzarà per poder accedir, tant per Mambo, com per una altra

suposada aplicació que estigui desenvolupada per l'equip en un futur, i així, poder reutilitzar-lo.

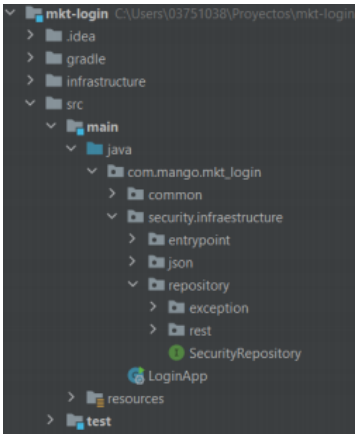

Fig. 13: Arquitectura de directoris login

Com es pot apreciar a la Figura 13, al ser un projecte tan petit, s'ha decidit no seguir cap estàndard d'arquitectura, únicament organitzar els paquets per funcionalitat.

Finalment, com es pot apreciar al seu diagrama de classes, a l'apèndix [A.4], es pot destacar el *LoginController*, l'*endpoint* per rebre la informació de l'usuari que vol entrar a l'aplicació, connectat amb *SecurityRepositoryImpl* per fer la comprovació de credencials i rols. Si es correcte, es retornarà un *token* per poder fer el login, i si no, es llençarà una excepció, aquelles que hereten de *LoginException*, depenent del tipus d'error.

7.3 Desenvolupament

En aquesta apartat de desenvolupament, es vol explicar com és el flux que segueix l'aplicació per poder realitzar un login i, posteriorment, utilitzar un excel per a que les dades es puguin guardar a la base de dades.

7.3.1 Mkt-Login

El flux d'aquest micro-servei comença quan l'usuari introdueix el seu e-mail i contraseña a la pàginal inicial de login, i clica sobre el botó d'enviar. Llavors, el *controller* del *back-end* recull aquestes credencials (la contrasenya en base64). El primer pas que hi ha, és comprovar que a l'endpoint on s'ha de fer l'autenticació de l'intranet de Mango, retorni una resposta amb *body*, i si no és el cas, retornarà un *body* buit i conseqüentment, una excepció. En el cas de que les credencials siguin correctes, en el *body* es retornarà un nom d'usuari, els seus rols respectius i un token. El segon pas per poder fer l'autenticació, és comprovar que dins del *field* de rols, estigui el necessari per poder consumir l'aplicació de Mambo. En el cas de no que es tingui el necessari, es llençarà una excepció. El tercer i últim pas es preparar la resposta per enviar al *front-end*, i així mostrar per pantalla a l'usuari si el *login* es correcte o no, però abans, el Data Manager de Mango, valida el token que s'ha rebut des del *back-end*, i així finalitzar el procés del *login*.

7.3.2 Mambo

Un cop realitzat el *login* correctament, es tindran els permisos per poder navegar en l'aplicació. Centrant-nos en el gestor desenvolupat, s'haurà d'accedir a la pàgina d'aquest per seleccionar un .csv i enviar-lo. Primerament, i després d'haver enviat l'arxiu, el *controller* del *back-end* rebrà aquest. Si és un .csv i no està buit, es procedirà a enviar-lo a un *service*, i si no, es llançarà l'excepció corresponent. Un cop que aquest *service* rebi l'arxiu, procedirà a enviar-lo a un *reader*, i aquest enviar-lo a un *mapper* per comprovar si el fitxer té el format adequat. Si el format no és correcte, es llançarà la respectiva excepció. Un cop el *mapper* validi les dades, les retornarà en forma de llista al *reader*, i aquest les retornarà al *service*. El segon pas, un cop les dades ja estan validades i el *service* les ha rebut amb èxit, es procedirà a introduir-les a la base de dades en el cas de que el producte no existeixi, però pel contrari, es realitzarà un *update*. Es llançarà una excepció si les dades de la base de dades no són correctes, o la query realitzada no és correcte. Finalment, es retornarà al *front-end* un missatge de *feedback* en qüestió de si el procediment ha estat correcte, o pel contrari, es mostrarà el missatge de l'excepció corresponent.

7.4 Tests

En aquest apartat de desenvolupament, s'explicarà com funcionen els diferents tests que s'han anat desenvolupant durant el projecte, i quina és la finalitat d'aquests.

7.4.1 Tests de funcionalitat *front-end*

Començant per els tests de funcionalitat, aquests es desenvolupen amb la llibreria de react-testing-library juntament amb les funcionalitats de Jest. Bàsicament, el que aquests tests proven és, primerament, que els components es renderitzin correctament amb les dades que li pertoquen a cadascun, i posteriorment, realitzar un *snapshot*. Finalment, el que també es prova és que en cadascun d'aquests components realitzi les accions necessàries per a que els events de l'aplicació, es facin correctament i sense cap error.

```
it('has a submit button with text=Send File', () => {
  render(
    <MemoryRouter>
      <FileForm handleSubmit={mockSubmit} handleFile={mockFile}
isEmptyFile={false} />
    </MemoryRouter>
  );

  expect(document.querySelector("button").name).toBe('button');
  expect(document.querySelector("button").type).toBe('submit');
  expect(document.querySelector("button").textContent).toBe('Send File');
});
```

Fig. 14: Exemple de test de funcionalitat

7.4.2 Tests unitaris *back-end*

Continuant amb els tests unitaris, es provaran que el que retorna cada classe està en un format correcte i en el que li pertoca. Per poder realitzar aquest tipus de tests farà falta utilitzar Mockito per mockejar les diferents classes que es necessitin en el test. Una altra part interessant és poder testejar els repositoris, és a dir, les classes que accedeixen a la base de dades i que interactuen amb ella. Per fer-ho, s'utilitzarà una base de dades en memòria per a simular una real.

```
@Test
void uploadFile_with_OK_response_from_text_csv() {
    MockMultipartFile file = new MockMultipartFile("data",
"Mapeo_Sizegrids_success.csv", "text/csv", "some xml".getBytes());

    ResponseEntity<Object> response = controller.uploadFile(file);

    assertThat(response).isEqualTo(new ResponseEntity<>(HttpStatus.OK));
}
```

Fig. 15: Exemple de test unitari

7.4.3 TESTS D'INTEGRACIÓ *back-end*

Finalment, els tests d'integració, el que volen testejar no és només el que una classe retorna, sinó que el flux que aquesta segueix es realitzi correctament, i en aquest cas, sense la necessitat de mockejar res. També serà necessari utilitzar una base de dades en memòria.

```
@Test
void is_not_a_CSV_file() throws IOException {
    InputStream inputStream =
getClass().getResourceAsStream("/Mapeo_Sizegrids_other_extension.xls");
    MockMultipartFile file = new MockMultipartFile("data",
"Mapeo_Sizegrids_other_extension.xls", "text/xls", inputStream);

    ResponseEntity<Object> response = controller.uploadFile(file);
    MamboException exception = (MamboException) response.getBody();

    assertThat(exception).isNotNull();
    assertThat(exception.getMessage()).isEqualTo("Please upload a CSV
file!");

assertThat(response.getStatusCode()).isEqualTo(HttpStatus.BAD_REQUEST);
}
```

Fig. 16: Exemple de test d'integració

8 RESULTATS

En el següent apartat, es donarà una visió global de com han estat els resultats obtinguts en el desenvolupament dels diferents micro-serveis a nivell de lògica, és a dir, la part del *back-end*. A les figures 17, 18 i 19 es poden observar també els resultats visualment, la part del *front-end*. Finalment, es discutirà si són els resultats esperats o no.

8.1 MICRO-SERVEI MKT-LOGIN

Els resultats obtinguts amb el desenvolupament d'aquest micro-servei són molt satisfactoris i els esperats. Per poder provar que funcioni, independentment dels tests fets, es va demanar que a cadascun dels membres de l'equip se'ls afegís el rol corresponent per entrar a l'aplicació. Un cop amb el rol afegit, es va procedir a provar l'aplicació amb dues persones diferents accedint localment, una amb rol i una altra sense. Efectivament, la persona que tenia el rol necessari va poder accedir, però l'altra no.

8.2 MICRO-SERVEI MAMBO

Els resultats obtinguts per aquest micro-servei també han estat molt satisfactoris i realment els esperats. La lògica objectiu per aquest micro-servei era poder enviar un excel i que totes les dades d'aquest es guardin a la base de dades corresponent, si aquestes dades són correctes. Per poder provar aquesta lògica, independentment dels tests, es va procedir a crear diversos excels, cadascun amb característiques diferents. Per exemple, excels sense capçalera o amb una fila buida, i així també poder saber si el missatge de feedback rebut és el correcte o no un cop enviat el document.

Fig. 17: Credencials incorrectes al login

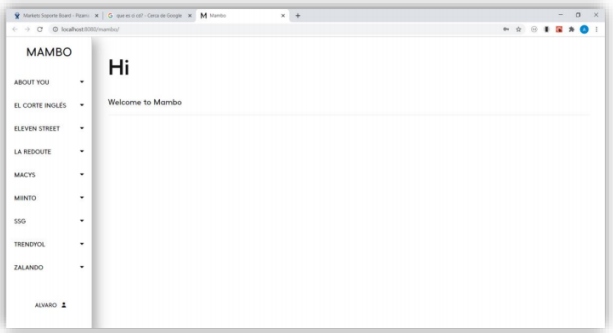

Fig. 18: Pàgina principal de l'aplicació

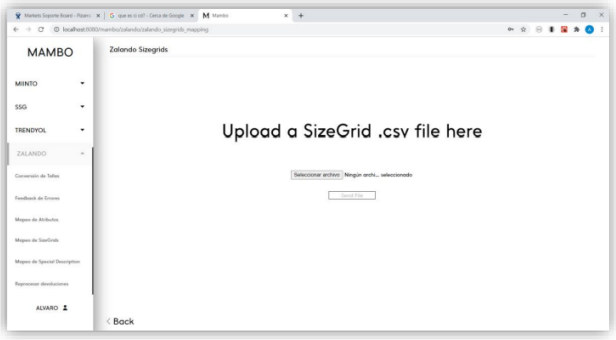

Fig. 19: Pàgina principal del gestor

A l'hora d'enviar diferents tipus de CSVs, es mostra un missatge de *feedback*:

- CSV correcte: *Uploaded the file successfully!*
- Format d'arxiu incorrecte: *Please upload a CSV file!*
- CSV buit: *Please upload a not empty file!*
- CSV sense capçaleres: *Error while processing CSV: Error of getting information from file, column X is missing!*
- CSV amb una capçalera diferent: *Error while processing CSV: Error of getting information from file, column X is not in first position!*
- CSV amb dades buides: *Error while processing CSV: Error of getting information from file, header is empty!*

*Nota: es pot observar a l'error de dades buides el següent missatge: *header is empty*. Es va decidir amb l'equip que cada fila d'un CSV s'anomenés com *header* per fer referència a aquests tipus d'errors.

A nivell visual, també s'ha volgut seguir un estil de Mango, fet que és un projecte per a aquesta empresa. També s'ha desenvolupat en un estil molt senzill per a que l'experiència d'usuari sigui molt bona i fàcil d'utilitzar.

9 Conclusions

Les conclusions que es poden treure d'aquest projecte d'empresa són diverses i bastant positives, tant a nivell personal com a nivell professional.

9.1 Nivell professional

Una de les principals, a nivell professional, és que els projectes no són tan fàcils de desenvolupar com en un principi es pensava. En un nivell tant avançat calen moltes més tasques de les que es fan a la universitat, com és l'exemple de la integració continua i distribució continua, en aquest cas amb Jenkins. També el control de les versions i les *pull requests* són bastant importants per a que els companys sàpiguen com és el codi i, posteriorment, es revisi per part de tot l'equip per a un correcte desenvolupament. Altre de les parts que un projecte requereix és el seu manteniment. Les versions de dependències o llenguatges es van actualitzant, i és molt important no deixar d'actualitzar aquests, per el motiu de que la corba de dificultat pot incrementar si es deixa de mantenir. Finalment, també s'ha pogut concloure que una bona documentació del treball realitzat pot ser molt útil per a un futur, ja que en el cas de que la persona desenvolupadora ja no estigui en el equip, gràcies a aquesta, es podria estalviar molt temps entenent la lògica, i sobretot, per a un equip que es dedica només al *back-end* i no al *front-end* (per la part de Reactjs).

9.2 Nivell personal

Continuant amb les conclusions que es poden treure a nivell personal, la principal és quan es comença a treballar, realment saps que queda molt per aprendre, i t'he n'adones que saps molt poc respecte als teus companys per la poca experiència. Respecte aquest, altre de les conclusions a treure, és que realment he aprés moltíssim desenvolupant aquest projecte, ja que s'ha realitzat amb diverses tecnologies que avui dia són bastant noves al mercat laboral, com és l'exemple de Spring, Reactjs, i també Gradle, que anteriorment no havia utilitzat mai. Finalment però no menys important, una altra de les conclusions que es podria considerar per a que un projecte estigui molt ben desenvolupat, és que els tests sempre són necessaris si o si. Desenvolupant aquests tests, es pot arribar a corregir bastants *bugs* i a provar la lògica del sistema.

10 Linies obertes

Després del desenvolupament d'aquesta base d'un gran projecte, amb l'empresa es va arribar a un acord a l'inici de tot. Aquest projecte s'haurà d'ampliar en un futur per seguir millorant algunes de les funcionalitats que avui dia a l'equip es realitzen. Per tant, un cop finalitzat aquest TFG, es vol continuar desenvolupant gestors durant el periode de temps que faci falta fins tenir-ho finalitzat. Caldrà un manteniment per a que el *backoffice* es pugui utilitzar durant un llarg periode de temps i segueixi facilitant la feina a l'equip.

Agraïments

Primer de tot, m'agradaria donar les gràcies als meus pares i germana per acompanyar-me en aquest llarg camí, des de parvulari fins aquest precís moment, per tot l'esforç que han fet per a que jo avui dia pugui estar redactant aquest document, i sobretot, per els valors que m'han trasmès. Seguidament, també vull donar les gràcies a la meva familia, per que també m'han acompanyat des de un segon pla. Després, donar les gràcies a tots els meus amics per el suport en tots aquests anys. Finalment, també m'agradaria incloure a tots aquells professors que sempre m'han ajudat i per l'educació que he rebut.

Referències

[1] https://www.iebschool.com/blog/que-es-para-que-sirve-sap-management/

[2] https://es.wikipedia.org/wiki/NetSuite

[3] https://en.wikipedia.org/wiki/Sage_Group

[4] https://www.gedesco.es/blog/que-es-sage/

[5] https://dynamics.microsoft.com/es-es/what-is-dynamics365/

[6] https://ubunlog.com/tryton-un-excelente-sistema-de-planificacion-de-recursos-empresariales/

[7] https://es.wikipedia.org/wiki/Tryton

APÈNDIX

A.1 TAULA DE PLANIFICACIÓ

Tasca	Data inici	Data final	Tasca	Data inici	Data final
1. Preparació inicial			Desenvolupar serveis *front-end*	05/04/2021	09/04/2021
Reunió inicial amb el tutor	12/02/2021	12/02/2021	Desenvolupar *hooks front-end*	12/04/2021	16/04/2021
Reunió inicial amb l'equip	15/02/2021	16/02/2021	Afegir estils CSS *front-end*	19/04/2021	20/04/2021
Reunió inicial amb els *stakeholders*	17/02/2021	19/02/2021	Afegir enrutador *front-end*	21/04/2021	23/04/2021
Captura de requeriments	18/02/2021	19/02/2021	Desenvolupar controladors *back-end*	26/04/2021	30/04/2021
2. Disseny			Desenvolupar serveis *back-end*	26/04/2021	30/04/2021
2.1 Disseny del Back Office			Configuració de la base de dades	03/05/2021	03/05/2021
Tecnologies a utilitzar	22/02/2021	22/02/2021	Creació de secrets amb AWS	03/05/2021	04/05/2021
Definició de l'arquitectura del codi	23/02/2021	24/02/2021	Connexió entre *back-end* i *front-end*	04/05/2021	04/05/2021
Mockup per el *front-end*	25/02/2021	04/03/2021	Sistema de *login*	05/05/2021	07/05/2021
Diagrama de classes per el *back-end*	05/03/2021	12/03/2021	Sistema de *logs back-end*	10/05/2021	10/05/2021
2.2 Memòries			Sistema de *feedback*	10/05/2021	13/05/2021
Informe inicial	20/02/2021	12/03/2021	Test dels *endpoints* amb Postman	05/04/2021	09/04/2021
3. Desenvolupament			Test de funcionalitat *front-end*	05/04/2021	07/04/2021
3.1 Fase inicial			Test unitaris *back-end*	26/04/2021	13/05/2021
Starli-template-micro-react *front-end*	15/03/2021	17/03/2021	Test d'integració *back-end*	26/04/2021	30/04/2021
Service-template *back-end*	18/03/2021	22/03/2021	3.4 Fase desplegament		
Familiarització amb Reactjs	22/03/2021	26/03/2021	Familiarització amb Docker	14/05/2021	18/05/2021
Familiarització amb SpringBoot	22/03/2021	26/03/2021	Familiarització amb Kubernetes	14/05/2021	18/05/2021
Configuració de Bitbucket	29/03/2021	29/03/2021	Desplegament *front-end*	19/05/2021	28/05/2021
Configuració de Jira	29/03/2021	29/03/2021	Desplegament *back-end*	19/05/2021	28/05/2021
Hello world	30/03/2021	30/03/2021	3.5 Memòries		
3.2 Fase pre-desenvolupament			Informe Progrés I	15/03/2021	23/04/2021
Definició de components *front-end*	31/03/2021	31/03/2021	Informe Progrés II	26/04/2021	28/05/2021
Definició de serveis *front-end*	31/03/2021	31/03/2021	4. Entrega final		
Definició de context *front-end*	01/04/2021	01/04/2021	4.1 Memòries		
Definició de hooks *front-end*	01/04/2021	01/04/2021	Proposta informe final	31/05/2021	18/06/2021
Definició de controladors *back-end*	02/04/2021	02/04/2021	Proposta de presentació	16/06/2021	25/06/2021
Definició de serveis *back-end*	02/04/2021	02/04/2021	Lliurament dossier final	17/06/2021	28/06/2021
3.3 Fase de desenvolupament			Lliurament de pòster	29/06/2021	02/07/2021
Desenvolupar components *front-end*	05/04/2021	09/04/2021			

A.2 DIAGRAMA DE CASOS D'ÚS

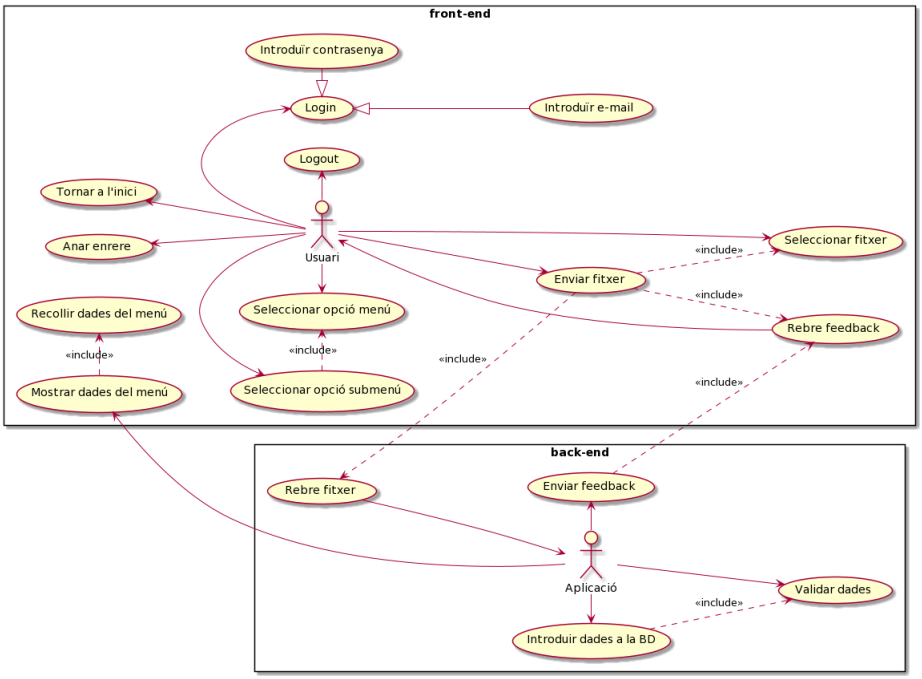

Fig. 20: Diagrama de casos d'ús

A.3 REQUERIMENTS FUNCIONALS

A.3.1 USUARIS

ID	REQ-F-1-01
Title	Accés
Description	Els usuaris han de poder accedir a l'aplicació a través de qualsevol navegador.
Priority	A
Verification	R
Parents	-

ID	REQ-F-1-02
Title	VPN
Description	Els usuaris han de poder accedir a l'aplicació amb la VPN de l'empresa.
Priority	A
Verification	R
Parents	-

ID	REQ-F-1-03
Title	*Login*
Description	Els usuaris han de poder logejar-se amb les seves credencials de Mango.
Priority	A
Verification	R, T, D
Parents	-

ID	REQ-F-1-04
Title	E-mail
Description	Els usuaris han de poder introduir el seu e-mail per accedir a l'aplicació.
Priority	A
Verification	R, D
Parents	REQ-F-1-03

ID	REQ-F-1-05
Title	Contrasenya
Description	Els usuaris han de poder introduir la seva contrasenya per accedir a l'aplicació.
Priority	A
Verification	R, D
Parents	REQ-F-1-03

ID	REQ-F-1-06
Title	Missatge d'error e-mail
Description	Els usuaris han de poder visualitzar un missatge d'error si deixen buida la casella de l'e-mail.
Priority	B
Verification	R, D
Parents	REQ-F-1-04

ID	REQ-F-1-07
Title	Missatge d'error contrasenya
Description	Els usuaris han de poder visualitzar un missatge d'error si deixen buida la casella de la contrasenya.
Priority	B
Verification	R, D
Parents	REQ-F-1-05

ID	REQ-F-1-08
Title	Botó de *login* activat
Description	Els usuaris han de poder clicar el botó de *login* un cop les credencials estiguin omplertes.
Priority	B
Verification	R, D
Parents	REQ-F-1-03

ID	REQ-F-1-09
Title	Botó de *login* desactivat
Description	Els usuaris no han de poder clicar el botó de *login* fins que les credencials no estiguin omplertes.
Priority	B
Verification	R, D
Parents	REQ-F-1-03

ID	REQ-F-1-10
Title	Nom personal
Description	Els usuaris han de poder visualitzar el seu nom un cop hagin entrat en l'aplicació.
Priority	M
Verification	R, D
Parents	REQ-F-1-03

ID	REQ-F-1-11
Title	*Logout*
Description	Els usuaris han de poder sortir de l'aplicació quan ho necessitin.
Priority	A
Verification	R, D, T
Parents	-

ID	REQ-F-1-12
Title	Botó de *logout* activat
Description	Els usuaris han de poder visualitzar el botó de *logout* un cop cliquin sobre el nom personal, i el botó estigui desactivat.
Priority	B
Verification	R, D
Parents	REQ-F-1-10, REQ-F-1-11

ID	REQ-F-1-13
Title	Botó de *logout* desactivat
Description	Els usuaris no han de poder visualitzar el botó de *logout* un cop cliquin sobre el nom personal, i el botó estigui activat.
Priority	B
Verification	R, D
Parents	REQ-F-1-10, REQ-F-1-11

ID	REQ-F-1-14
Title	Títol
Description	Els usuaris han de poder clicar al títol "Mambo" per poder anar a la pàgina principal de l'aplicació.
Priority	M
Verification	R, T, D
Parents	-

ID	REQ-F-1-15
Title	Menú
Description	Els usuaris han de poder visualitzar i clicar sobre qualsevol apartat del menú.
Priority	A
Verification	R, T
Parents	-

ID	REQ-F-1-16
Title	Submenú activat
Description	Els usuaris han de poder visualitzar i clicar sobre qualsevol subapartat del submenú, un cop s'hagi seleccionat qualsevol apartat del menú.
Priority	A
Verification	R, D
Parents	REQ-F-1-15

ID	REQ-F-1-17
Title	Submenú desactivat
Description	Els usuaris no han de poder visualitzar ni clicar sobre qualsevol subapartat del submenú, un cop s'hagi deixat de seleccionar qualsevol apartat del menú.
Priority	A
Verification	R, D
Parents	REQ-F-1-15

ID	REQ-F-1-18
Title	Gestor size-grids
Description	Els usuaris han de poder accedir al gestor de size-grids un cop s'hagi clicat sobre l'opció en el submenú.
Priority	A
Verification	R, T, D
Parents	-

ID	REQ-F-1-19
Title	Gestor size-grids - Escollir fitxer
Description	Els usuaris han de poder clicar sobre el botó d'escollir fitxer, i seleccionar-ne un.
Priority	A
Verification	R, T
Parents	REQ-F-1-18

ID	REQ-F-1-20
Title	Gestor size-grids - Botó d'enviar fitxer activat
Description	Els usuaris han de poder enviar un fitxer, clicant el botó d'enviar, un cop s'hagi seleccionat un qualsevol.
Priority	M
Verification	R, T, D
Parents	REQ-F-1-18

ID	REQ-F-1-21
Title	Gestor size-grids - Botó d'enviar fitxer desactivat
Description	Els usuaris no han de poder enviar el fitxer, clicant el botó d'enviar, fins que no s'hagi seleccionat un qualsevol.
Priority	M
Verification	R, T, D
Parents	REQ-F-1-18

ID	REQ-F-1-22
Title	Gestor size-grids - Rebre *feedback*
Description	Els usuaris han de poder visualitzar un missatge de *feedback* un cop s'hagi enviat el fitxer.
Priority	A
Verification	R, T, D
Parents	REQ-F-1-18

ID	REQ-F-1-23
Title	Botó d'enrere
Description	Els usuaris han de poder anar enrere, és a dir, a la pàgina anterior de l'aplicació en la que es trobaven, un cop s'hagi clicat el botó.
Priority	B
Verification	R, D
Parents	-

A.4 Requeriments no funcionals

A.4.1 Seguretat

ID	REQ-NF-1-01
Title	Dades d'usuaris
Description	Les dades dels usuaris queden sota confidencialitat amb accés no autoritzat.
Priority	A
Verification	R
Parents	-

ID	REQ-NF-1-02
Title	Accessos no desitjats
Description	L'aplicació ha de ser capaç de prevenir accessos o modificacions no autoritzades.
Priority	A
Verification	R
Parents	REQ-NF-1-01

ID	REQ-NF-1-03
Title	Contrasenya
Description	Els usuaris han de tenir una contrasenya segura per l'autenticació.
Priority	A
Verification	R, D
Parents	REQ-NF-1-01, REQ-NF-1-02

ID	REQ-NF-1-04
Title	Ús
Description	Per poder utilitzar l'aplicació localment, en preproducció i en producció, caldrà obtenir els secrets corresponents d'usuari.
Priority	A
Verification	R, T, D
Parents	-

ID	REQ-NF-1-05
Title	Base de dades
Description	Per poder introduir, eliminar, llegir i actualitzar dades de la base de dades, caldrà que les credencials d'aquesta es llegeixin sota secrets de la base de dades, conjuntament amb els secrets seleccionats de l'usuari.
Priority	A
Verification	R, T, D
Parents	REQ-NF-1-04

ID	REQ-NF-1-06
Title	Rols IAM
Description	Per poder accedir a l'aplicació, caldrà que la compta d'usuari tingui els permisos del rol IAM de l'aplicació, per a que no pugui entrar qualsevol persona de Mango ni de fora, únicament de l'equip. Aquest rol és *marketplaces_mambo_general*.
Priority	A
Verification	R, T, D
Parents	REQ-NF-1-01, REQ-NF-1-02

A.4.2 Rendiment

ID	REQ-NF-2-01
Title	Temps de resposta
Description	Els temps de resposta de l'aplicació ha de ser ràpid, evitant renderitzats innecessaris.
Priority	A
Verification	R, D
Parents	-

ID	REQ-NF-2-02
Title	Concurrència
Description	La concurrència que ha de tenir l'aplicació ha de ser entre els 5 i els 20 usuaris.
Priority	M
Verification	R, D
Parents	-

ID	REQ-NF-2-03
Title	Enviar fitxer
Description	Els temps en enviar el fitxer ha de ser menor a un segon.
Priority	A
Verification	R
Parents	REQ-F-1-20

ID	REQ-NF-2-04
Title	Rebre informació del menú sencer
Description	Els temps en rebre la informació de la base de dades del menú, ha de ser menor a un segon.
Priority	A
Verification	R
Parents	REQ-F-1-15

A.4.3 APARENÇA/USABILITAT

ID	REQ-NF-3-01
Title	UX
Description	L'experiència d'usuari ha de ser bona, amb un sistema fàcil d'utilitzar i atractiu.
Priority	A
Verification	R, D
Parents	-

ID	REQ-NF-3-02
Title	Ús
Description	L'aplicació s'ha de poder utilitzar des de qualsevol tipus d'ordinador.
Priority	A
Verification	R
Parents	-

A.4.4 MANTENIMENT

ID	REQ-NF-4-01
Title	Codi *front-end*
Description	El codi del micro-servei del *front-end* ha d'estar desenvolupat amb Reactjs.
Priority	A
Verification	R, D
Parents	-

ID	REQ-NF-4-02
Title	Codi *back-end*
Description	El codi del micro-servei del *back-end* ha d'estar desenvolupat amb Java Springboot.
Priority	A
Verification	R, D
Parents	-

ID	REQ-NF-4-03
Title	Estructura *front-end*
Description	El codi del micro-servei del *front-end* ha d'estar estructurat per components, un estil semblant al MVC.
Priority	A
Verification	D
Parents	REQ-NF-4-01, REQ-NF-4-02

ID	REQ-NF-4-04
Title	Estructura *back-end*
Description	El codi del micro-servei del *back-end* ha d'estar estructurat amb l'arquitectura hexagonal adaptada per l'equip.
Priority	A
Verification	D
Parents	REQ-NF-4-01, REQ-NF-4-02

ID	REQ-NF-4-05
Title	Codi *hardcoded*
Description	El codi del micro-servei, tant del *front-end* com del *back-end*, no pot estar *hardcoded*.
Priority	A
Verification	T, D
Parents	REQ-NF-4-01, REQ-NF-4-02

A.4.5 DISPONIBILITAT

ID	REQ-NF-5-01
Title	Navegadors
Description	L'aplicació ha d'estar disponible des de qualsevol navegador.
Priority	M
Verification	R
Parents	REQ-F-1-01

ID	REQ-NF-5-02
Title	VPN
Description	L'aplicació ha d'estar disponible un cop la VPN de l'empresa estigui connectada.
Priority	A
Verification	R
Parents	REQ-F-1-02