Disseny i Desenvolupament d'una Plataforma escalable i personalitzable per a la Gestió de Clients d'una gestoria

Memòria de Projecte Final de Màster

Màster Universitari en Aplicacions Multimèdia

Itinerari Professionalitzador

Autor: Roger Vidal Lloveras

Consultor: Mikel Zorrilla Berasategui
Professora: Laura Porta Simó

Desembre 2020

Memòria del projecte

Codi Font del projecte

Dependències, frameworks i recursos de tercers

Aquest projecte fa ús de frameworks, dependències i recursos de tercers (llibreries d'icones entre d'altres). Tots aquests estan distribuïts sota llicències MIT, Apache o similars. Es poden trobar detallades les dependències, frameworks i recursos de tercers utilitzats en el Capítol 3 d'aquesta memòria.

Títol del treball:	*Disseny i Desenvolupament d'una Plataforma escalable i personalitzable per a la Gestió de Clients d'una gestoria*
Nom de l'autor:	*Roger Vidal Lloveras*
Nom del consultor/a:	*Mikel Zorrilla Berasategui*
Nom del PRA:	*Laura Porta Simó*
Data de lliurament:	*12/2020*
Titulació o programa:	*Màster Universitari en Aplicacions Multimèdia*
Àrea del Treball Final:	*Treball Final de Màster Professionalitzador*
Idioma del treball:	*Català*
Paraules clau	*Plataforma Web, API Rest, Gestoria*

Resum del Treball:

El propòsit d'aquest TFM és dissenyar i desenvolupar una plataforma web escalable i personalitzable perquè els clients d'una gestoria puguin accedir a les seves dades i altra tipus d'informació proveïda per el seu gestor.

El punt de partida d'aquesta idea, sorgeix de la necessitat d'una gestoria en poder oferir un servei distintiu als seus clients, que els hi permeti destacar en un mercat amb una gran competència. Actualment, ja compten amb un portal per als seus clients, però aquest és una solució comercial estàndard i poc personalitzable. L'objectiu de desenvolupar un producte propi, és la capacitat de poder personalitzar la totalitat del contingut de la plataforma i poder-hi afegir característiques que no es troben disponibles en cap altra solució existent.

Per tal d'aconseguir desenvolupar un producte que estigui a l'altura de les expectatives, s'ha treballat seguint una metodologia de treball àgil, seguint la planificació plantejada i estant sempre en contacte amb el client. A nivell tecnològic, s'ha treballat sempre amb les noves tecnologies i tendències del mercat, tant a la part del servidor com en el client web.

Un cop acabat el projecte, es pot dir que s'han assolit amb èxit tots els objectius. El resultat final és una aplicació web totalment personalitzable i modular, que s'adapta a totes les plataformes i resolucions. Aquest treball ha estat una oportunitat perfecte per a l'autor del TFM per a poder posar en pràctica els conceptes assolits durant el Màster, per a formar-se en nous àmbits i per a tenir un producte que serveixi com a punt de sortida per al seu *portfolio*.

Abstract:

The aim of this Master's dissertation is to design and develop a customizable and scalable web platform for a consulting firm in order to provide to their clients a way to access all their personal data and some other kind of information.

The main idea of the project, comes from the necessity of a consulting firm to offer a distinguishable service for their customers, in order to be able to make a difference with their competitors. Currently, they already have a portal for their customers, but it's a standard and poorly customizable business solution. The goal of developing their own product is the ability to customize the entire content of the platform and add some features that are not available in any other commercial existing solution.

In order to be able to develop a product that meets the expectations, it has been done using agile methods, following the planned schedule and always being in contact with the customer. In the technological side of the project, the main objective has been to use the latest and best available technologies on the market in both sides of the project: Frontend and Backend.

After finishing the project, all the goals have been reached. The final result is a fully customizable and modular web application, that can be used in any platform and resolution. This project has been a good opportunity to test all the new skills learned during this Master and also to have a professional looking product for the author's portfolio.

"All our dreams can come true,
If we have the courage to pursue them"

Walter Elias Disney

Abstract

The aim of this Master's dissertation is to design and develop a customizable and scalable web platform for a consulting firm in order to provide to their clients a way to access all their personal data and some other kind of information.

The main idea of the project, comes from the necessity of a consulting firm to offer a distinguishable service for their customers, in order to be able to make a difference with their competitors. Currently, they already have a portal for their customers, but it's a standard and poorly customizable business solution. The goal of developing their own product is the ability to customize the entire content of the platform and add some features that are not available in any other commercial existing solution.

In order to be able to develop a product that meets the expectations, it has been done using agile methods, following the planned schedule and always being in contact with the customer. In the technological side of the project, the main objective has been to use the latest and best available technologies on the market in both sides of the project: Frontend and Backend.

After finishing the project, all the goals have been reached. The final result is a fully customizable and modular web application, that can be used in any platform and resolution. This project has been a good opportunity to test all the new skills learned during this Master and also to have a professional looking product for the author's portfolio.

Resum

El propòsit d'aquest TFM és dissenyar i desenvolupar una plataforma web escalable i personalitzable perquè els clients d'una gestoria puguin accedir a les seves dades i altra tipus d'informació proveïda per el seu gestor.

El punt de partida d'aquesta idea, sorgeix de la necessitat d'una gestoria en poder oferir un servei distintiu als seus clients, que els hi permeti destacar en un mercat amb una gran competència. Actualment, ja compten amb un portal per als seus clients, però aquest és una solució comercial estàndard i poc personalitzable. L'objectiu de desenvolupar un producte propi, és la capacitat de poder personalitzar la totalitat del contingut de la plataforma i poder-hi afegir característiques que no es troben disponibles en cap altra solució existent.

Per tal d'aconseguir desenvolupar un producte que estigui a l'altura de les expectatives, s'ha treballat seguint una metodologia de treball àgil, seguint la planificació plantejada i estant sempre en contacte amb el client. A nivell tecnològic, s'ha treballat sempre amb les noves tecnologies i tendències del mercat, tant a la part del servidor com en el client web.

Un cop acabat el projecte, es pot dir que s'han assolit amb èxit tots els objectius. El resultat final és una aplicació web totalment personalitzable i modular, que s'adapta a totes les plataformes i resolucions. Aquest treball ha estat una oportunitat perfecte per a l'autor del TFM per a poder posar en pràctica els conceptes assolits durant el Màster, per a formar-se en nous àmbits i per a tenir un producte que serveixi com a punt de sortida per al seu *portfolio*.

Paraules Clau

Treball de Final de Màster, Plataforma Web, API REST, Gestoria, Clients, ReactJS, Single Page Application, Aplicació Web

Notacions i convencions

TÍTOL 1

Fa referència al nom dels capítols. S'utilitza la tipografia **Trebuchet MS** negreta a mida 12. Tot el títol s'escriu utilitzant només majúscules.

Títol 2

Es fa servir per a definir els diferents apartats dintre d'un capítol. S'utilitza la tipografia **Trebuchet MS** negreta a mida 12.

Títol 3

S'utilitza per a definir els sub-apartats dintre de cadascun dels apartats principals de cada capítol. S'utilitza la tipografia **Trebuchet MS** negreta a mida 11.

Estrangerisme

Per a indicar les paraules escrites en un altre idioma, s'utilitza la tipografia **Trebuchet MS** cursiva a mida 11.

Nom destacat

Per a destacar noms importants dintre del propi apartat, s'utilitza la tipografia **Trebuchet MS** negreta a mida 11.

```
Elements Codi
```

Per a mostrar fragments de codi, s'utilitza la tipografia **Consolas** regular a mida 10'5 amb fons de color negre 15%. Per als colors de les paraules, s'utilitza el codi de colors definit de Visual Studio Code.

Per al contingut general de la memòria, s'utilitza la tipografia **Trebuchet MS** regular a mida 11, interlineat de 1'15 i justificació de text.

Índex de Continguts

Índex de figures

Índex de taules

CAPÍTOL 1: INTRODUCCIÓ

Aquest document és la memòria del Treball de Final de Màster (TFM) del màster universitari d'Aplicacions Multimèdia. Aquesta memòria té la funció de documentar tot el procés de disseny i desenvolupament d'una Plataforma Web de Gestió de Clients per a una Gestoria.

1. Introducció/Prefaci

El plantejament de la idea de treballar sobre aquest projecte, sorgeix de la necessitat d'una gestoria de Vilanova i la Geltrú, de poder oferir una solució actualitzada per als temps actuals als seus clients de cara a poder accedir a dades i fer certs tràmits de manera digital.

A Vilanova i la Geltrú, una ciutat de gairebé 70.000 habitants, hi ha aproximadament vint gestories disputant-se l'oferta en aquest mercat. Els principals serveis que ofereixen aquestes gestories, és el d'assessorament i consultoria jurídica i fiscal. Els seus possibles clients, poden ser tant persones a títol propi com diferents empreses del territori. A nivell d'Espanya, segons dades de l'Institut Nacional d'Estadística [1], hi ha unes 70.000 gestories. Si tenim en compte que hi ha aproximadament 1.500.000 empreses i 1.750.000 autònoms, vol dir que aproximadament cada gestoria només té 46 clients potencials.

Actualment l'empresa ja disposa d'una plataforma web per a que els seus clients puguin accedir a aquestes dades, però al ser un producte estàndard, consideren que aquesta no s'adapta prou bé a les seves necessitats. En un mercat objectiu amb tanta competència, per a poder destacar s'ha de tenir un producte únic i que ofereixi millor característiques que qualsevol producte que puguin oferir els seus competidors.

2. Descripció/Definició

El propòsit principal d'aquest Treball de Final de Màster, és desenvolupar una aplicació web que faci la funció de plataforma de gestió per als clients d'una gestoria. En aquesta, els clients de la gestoria han de poder realitzar gestions simples de manera digital i també poder disposar de totes aquelles dades que la gestoria cregui convenients.

Totes les dades de la plataforma venen donades per la gestoria, que ja disposa d'aquestes en una base de dades. Al estar tractant amb dades privades dels clients de la gestoria, és de vital importància la seguretat de l'aplicació i l'accés a aquesta.

La plataforma resultant d'aquest projecte, ha de ser un producte totalment escalable i multiplataforma, que encaixi en la seva totalitat amb la imatge d'empresa de la gestoria.

3. Objectius Generals i Abast

A continuació es defineixen els objectius del Treball de Final de Màster.

3.1. Objectius principals

- Dissenyar i desenvolupar una plataforma web amb la finalitat de que sigui un portal perquè els clients d'una gestoria puguin realitzar gestions de manera digital i tinguin accés a diferents dades i documents.
- Dissenyar i desenvolupar la plataforma web seguint un patró de disseny RWD [2] i tenint en compte en tot moments que aquesta ha de ser totalment escalable i personalitzable.

3.2. Objectius del client

- Oferir un servei distintiu respecte la competència i potenciar la fidelització de clients.
- Apostar per la digitalització de l'empresa [3] i així poder estalviar costos humans en un futur i reduir errors.

3.3. Objectius personals

- Consolidar i aplicar els coneixements adquirits durant el màster en els diferents àmbits del procés de creació d'una aplicació multimèdia.
- Finalitzar el màster tenint un producte amb acabats professionals que serveixi de *Portfolio* i oportunitat comercial.

3.4. Abast i Futur

Al acabar el treball, els usuaris han de poder entrar al seu compte de forma segura, i un cop dins han de poder accedir a documents, accedir a dades d'informació, contactar amb el seu gestor i reservar cita prèvia.

Com a treball futur un cop entregat el TFM, es continuaran desenvolupant mòduls per a la plataforma tals com un possible *chatbot* [4], sistema de gestió d'altes i baixes de la seguretat social per autònoms i petits empresaris, etc.

4. Metodologia i procés de treball

Tal i com s'ha vingut explicant en els punts anteriors, l'objectiu d'aquest projecte consisteix en desenvolupar una plataforma web per a que els clients d'una gestoria puguin realitzar tràmits de manera digital i accedir a certes dades. Per a duu a terme aquest desenvolupament, el projecte s'ha dividit en quatre fases principals:

- **Anàlisi:** Aquesta fase és el tret de sortida del projecte. En ella es defineixen els objectius del projecte i els requeriments finals. Un cop definits, es procedeix a fer un estudi del mercat, estudi de les possibles tecnologies de desenvolupament i es marquen les eines necessàries.

- **Planificació:** En l'etapa de planificació del projecte, es desenvolupa un calendari on es veuen reflectides les diferents tasques a duu a terme durant el desenvolupament total del projecte. Tenint en compte que aquest projecte s'ha desenvolupat com a producte d'un Treball de Final de Màster, aquesta planificació ve marcada per les guies establertes en el pla d'estudis, que es basen en dividir el desenvolupament en cinc grans fases (PAC).

- **Disseny:** En aquesta fase es treballa en el disseny del model de dades, l'arbre de navegació de l'aplicació, els diferents elements relacionats amb la interfície, i finalment, s'acaben dissenyant uns primers prototips del producte.

- **Desenvolupament:** En la fase de desenvolupament, es porten a terme totes les tasques relacionades amb el desenvolupament tècnic del producte. Per una banda, es treballa en la part del *Back-End* (tots els temes relacionats amb servidors i dades). Per altra banda, un cop es té una base feta en el *Back-End*, es treballa la part del *Front-End* del projecte (l'aplicació web).

Figura 1 - Diagrama de procés de treball

Per a treballar en les diferents fases, s'han dut a terme seguint una metodologia de treball àgil [5]. Aquesta metodologia es basa en l'adaptabilitat de qualsevol canvi en el projecte, per tal d'augmentar la qualitat final del producte. En cada etapa de desenvolupament del projecte, s'ha revisat amb el client l'estat actual del producte i s'ha adaptat la planificació inicial en el cas de que el resultat de la reunió hagi significat la necessitat d'aplicar canvis en el producte.

5. Planificació

La planificació del projecte ha estat dividida en cinc entregues corresponents a les diferents proves d'avaluació continua determinades per el calendari de l'assignatura Treball de Final de Màster.

5.1. Taula de fites de la planificació inicial

A continuació, es pot veure la proposta de planificació inicial en forma de taula de fites. En aquesta taula, es poden veure les dates corresponents a cada tasca i els dies total que haurien d'ocupar.

FASE	ID	TASCA	INICI	FINAL	DÍES
PAC 1: Proposta	1		21/09/2020	01/10/2020	11
	1.1	Debat de la Proposta	22/09/2020	23/09/2020	2
	1.2	Formalització de la Proposta	26/09/2020	30/09/2020	5
PAC 2: Mandat del Projecte i Planificació	2		02/10/2020	13/10/2020	12
	2.1	Estudi i analisi del format TFM	04/10/2020	04/10/2020	1
	2.2	Definició d'objectius	05/10/2020	05/10/2020	1
	2.3	Estat de l'Art	06/10/2020	09/10/2020	4
	2.4	Planificació del projecte	10/10/2020	11/10/2020	2
PAC 3: Disseny i Inici de desenvolupament	3		14/10/2020	09/11/2020	27
	3.1	Redacció memòria Parcial 1	14/10/2020	09/11/2020	27
	3.2	Definir limits del projecte amb client	14/10/2020	15/10/2020	2
	3.3	Investigar tecnologies auxiliars	16/10/2020	17/10/2020	2
	3.4	Aprenentatge llenguatge	18/10/2020	20/10/2020	3
	3.5	Disseny inicial UI	21/10/2020	22/10/2020	2
	3.6	Crear esquelet plataforma	23/10/2020	23/10/2020	1
	3.7	Crear entorn Dev. segur (BBDD de proves, etc..)	24/10/2020	24/10/2020	1
	3.8	Connectivitat BBDD i Servidors	25/10/2020	26/10/2020	2
	3.9	Safe Login i Sessió	27/10/2020	28/10/2020	2
	3.10	Definir parametrització RWD	29/10/2020	29/10/2020	1
	3.11	Reunió client	30/10/2020	30/10/2020	1
	3.12	Crear primera versió pàgina principal	02/11/2020	05/11/2020	4
	3.13	Bolcar dades BBDD a pàgina principal	06/11/2020	07/11/2020	2
PAC 4: Desenvolupament basat en dades	4		10/11/2020	09/12/2020	30
	4.1	Redacció memòria Parcial 2	10/11/2020	09/12/2020	30
	4.2	Aplicar estils	10/11/2020	15/11/2020	6
	4.3	Navegació plataforma	16/11/2020	19/11/2020	4
	4.4	Distribució de documentació des de Servidor	20/11/2020	23/11/2020	4
	4.5	Crear mòdul gestió S.S.	24/11/2020	28/11/2020	5
	4.6	Crear mòdul contacte	29/11/2020	01/12/2020	3
	4.7	Afegir idiomes (Cat / Cas / Ang)	02/12/2020	03/12/2020	2
	4.8	Reunió client	04/12/2020	04/12/2020	1
PAC 5: Pulir i Test	5		10/12/2020	31/12/2020	22
	5.1	Redacció memòria Final	10/12/2020	31/12/2020	22
	5.2	Aplicar canvis demanats per el client	11/12/2020	23/12/2020	13
	5.3	Testejar aplicació	24/12/2020	30/12/2020	7

Taula 1 - Planificació inicial

5.2. Diagrama de GANTT de la planificació inicial

En aquest apartat es mostra la planificació inicial del projecte en forma de diagrama de *GANTT*, amb l'objectiu de poder veure de manera visual l'evolució del projecte respecte el temps disponible.

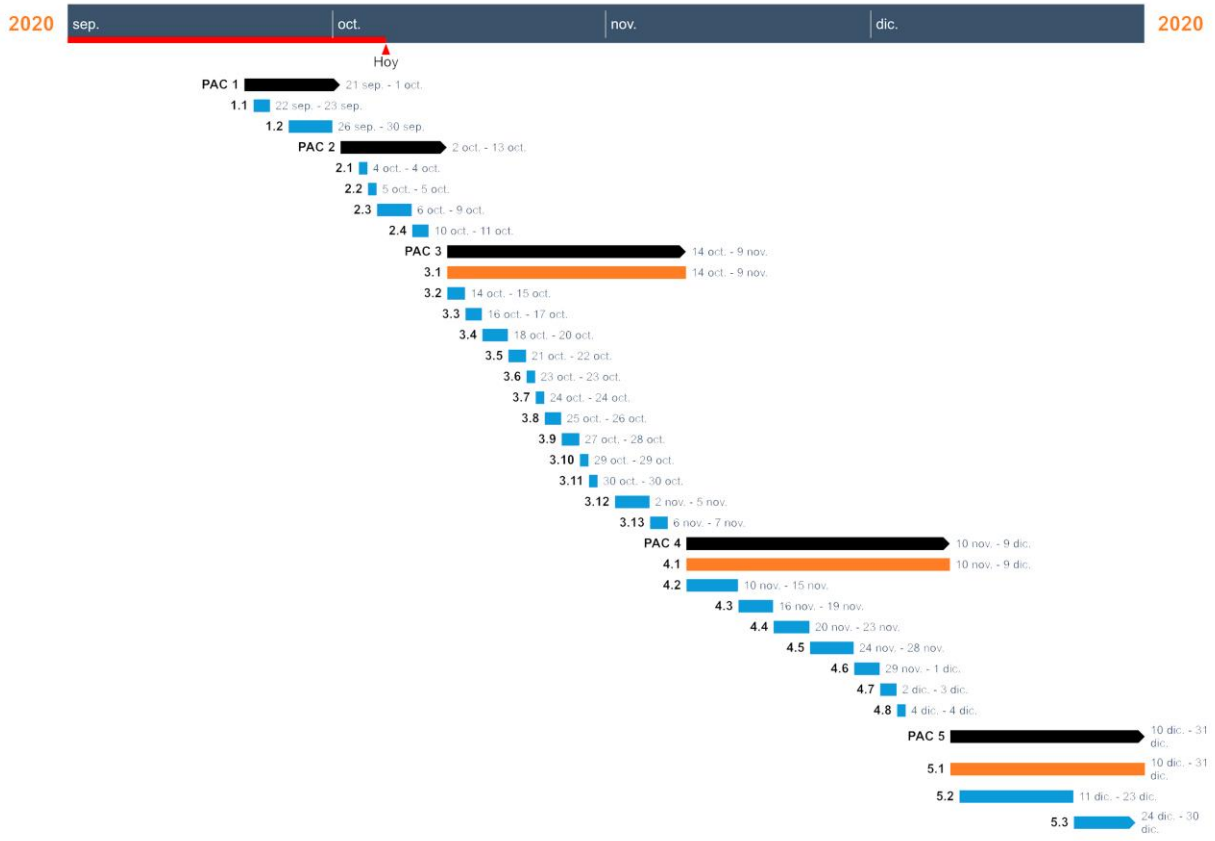


Figura 2 - Diagrama de GANTT planificació inicial

5.3. Taula de fites de la planificació final

En la següent taula, es pot veure la planificació final del projecte en forma de taula de fites.

FASE	ID	TASCA	INICI	FINAL	DIES
PAC 1: Proposta	1		21/09/2020	01/10/2020	11
	1.1	Debat de la Proposta	22/09/2020	23/09/2020	2
	1.2	Formalització de la Proposta	26/09/2020	30/09/2020	5
PAC 2: Mandat del Projecte i Planificació	2		02/10/2020	13/10/2020	12
	2.1	Estudi i analisi del format TFM	04/10/2020	04/10/2020	1
	2.2	Definició d'objectius	05/10/2020	05/10/2020	1
	2.3	Estat de l'Art	06/10/2020	09/10/2020	4
	2.4	Planificació del projecte	10/10/2020	11/10/2020	2
PAC 3: Disseny i Inici de desenvolupament	3		14/10/2020	09/11/2020	27
	3.1	Redacció memòria Parcial 1	14/10/2020	07/11/2020	25
	3.2	Definir limits del projecte amb client	14/10/2020	14/10/2020	1
	3.3	Investigar tecnologies auxiliars	15/10/2020	17/10/2020	3
	3.4	Aprenentatge llenguatge	18/10/2020	20/10/2020	3
	3.5	Disseny inicial UI	21/10/2020	22/10/2020	2
	3.6	Crear esquelet plataforma	23/10/2020	23/10/2020	1
	3.7	Crear entorn Dev. segur (BBDD de proves, etc..)	24/10/2020	24/10/2020	1
	3.8	Creació Rest API	25/10/2020	26/10/2020	2
	3.9	Connectivitat BBDD i Servidors	27/10/2020	27/10/2020	1
	3.10	Safe Login i Sessió	28/10/2020	28/10/2020	1
	3.11	Reunió client	29/10/2020	29/10/2020	1
	3.12	Crear primera versió pàgina principal	30/10/2020	04/11/2020	6
PAC 4: Desenvolupament basat en dades	4		10/11/2020	09/12/2020	30
	4.1	Redacció memòria Parcial 2	10/11/2020	07/12/2020	28
	4.2	Aplicar estils	10/11/2020	12/11/2020	3
	4.3	Navegació plataforma	13/11/2020	16/11/2020	4
	4.4	Distribució de documentació des de Servidor	17/11/2020	20/11/2020	4
	4.5	Crear mòdul gestió Mur Client-Assessor	21/11/2020	29/11/2020	9
	4.6	Crear mòdul contacte	30/11/2020	30/11/2020	1
	4.7	Afegir idiomes (Cat / Cas / Ang)	01/12/2020	02/12/2020	2
	4.8	Reunió client	03/12/2020	03/12/2020	1
	4.9	Millora de codi	04/12/2020	06/12/2020	3
PAC 5: Pulir i Test	5		10/12/2020	31/12/2020	22
	5.1	Redacció memòria Final	10/12/2020	24/12/2020	15
	5.2	Preparar presentacions (Pública i acadèmica)	10/12/2020	28/12/2020	19
	5.3	Arreglar compatibilitats CSS	11/12/2020	12/12/2020	2
	5.4	Integrar certificats SSL	13/12/2020	14/12/2020	2
	5.5	Treballar en les Queries de SQL	15/12/2020	16/12/2020	2
	5.6	Canvis varis en client (codi i estètica)	17/12/2020	23/12/2020	7
	5.7	Testejar aplicació	24/12/2020	30/12/2020	7

Taula 2 - Planificació final del projecte

5.4. Diagrama de GANTT de la planificació final

A continuació es pot veure la planificació final del projecte en forma de diagrama de GANTT, per tal de poder veure de manera visual l'evolució del projecte respecte el temps.

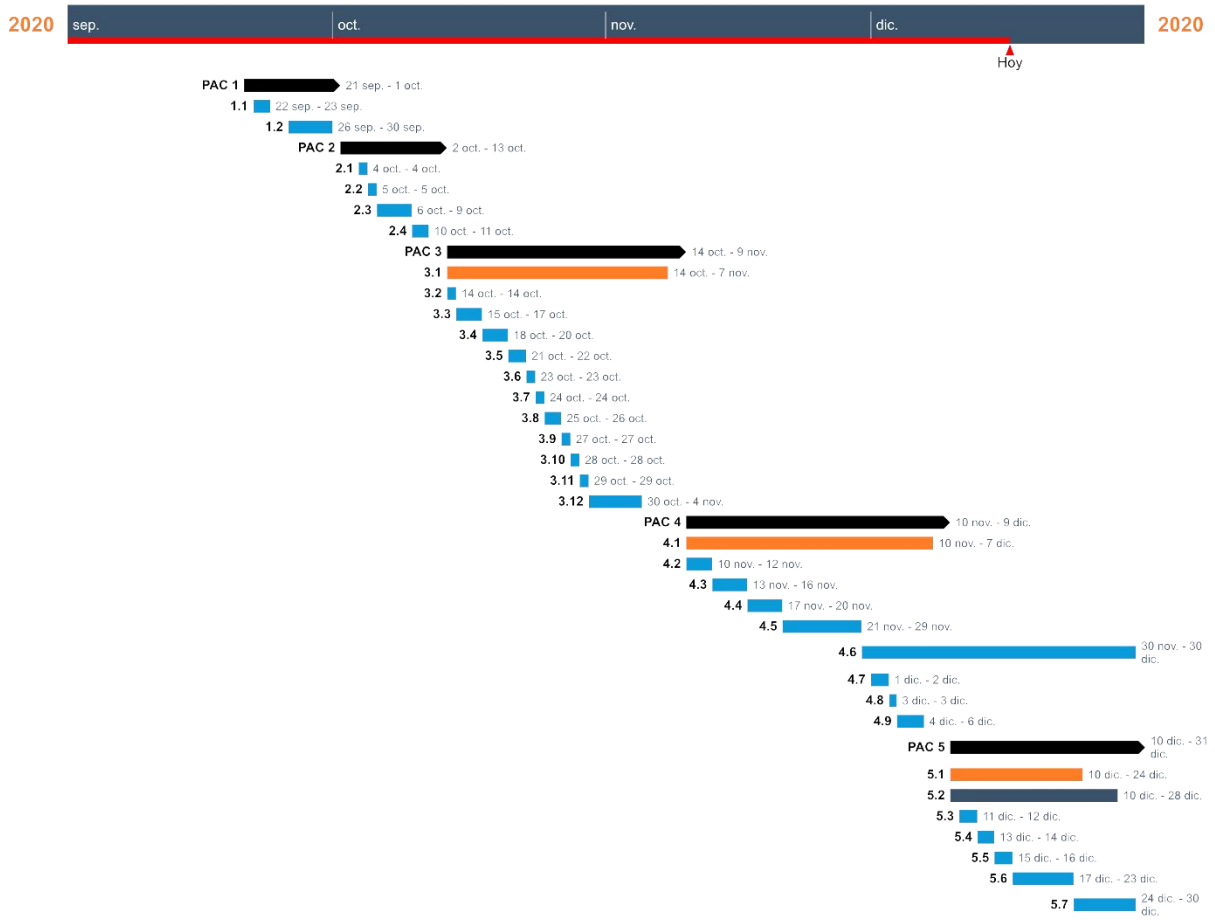


Figura 3 - Diagrama de GANTT planificació final

5.5. Anàlisi dels canvis soferts en la planificació del projecte

Un cop acabat el Treball de Final de Màster, s'ha comparat la planificació que s'ha acabat complint respecte la planificació inicial del projecte.

El primer que es pot observar, es que en cap dels casos s'ha hagut d'aplaçar una tasca corresponent a un dels grans blocs a el següent bloc. En les cinc entregues, s'ha pogut complir de manera molt satisfactòria amb el calendari establert per les diferents PAC de l'assignatura.

En els dos primers gran blocs, com es pot observar no hi ha cap canvi, ja que per calendari de l'assignatura, les primeres tasques a treballar ja venen establertes.

Com es pot observar, en el tercer bloc podem veure els primers petits canvis. Per una banda, es pot observar com s'ha pogut retallar temps en algunes tasques, com en la reunió amb el client i la connectivitat a les bases de dades. Gràcies a això, s'ha pogut dedicar un dia més a investigar les tecnologies, ja que és una tasca amb un pes molt important dintre el projecte. També, degut a que quan es va fer el plantejament inicial encara no estaven definides les tecnologies a treballar, s'ha afegit la tasca de crear l'API REST (que inicialment no estava contemplat). Tot i aquests canvis, es pot observar que el calendari s'ha pogut complir, fins i tot podent avançar el dia final previst.

En el cas del quart bloc corresponent al "Desenvolupament basat en dades", tan mateix com en el bloc anterior, s'ha pogut complir sense cap problema i els canvis soferts son molt petits. Primer de tot, s'han reduït els temps en algunes de les tasques, com per exemple en la redacció de la memòria parcial, ja que s'ha considerat necessari tenir dos dies de marge per a poder completar l'entrega. També, degut al procés d'aprenentatge que comporta treballar amb tecnologies que s'estan aprenent, s'ha afegit una última tasca de millora de codi. Finalment, el canvi més gran en aquest bloc es tracta d'haver canviat la creació del mòdul de la seguretat social, al mòdul de missatgeria. Aquest canvi s'ha fet per decisió del client, però al tenir dies de marge que s'han retallat en altres tasques, s'ha pogut col·locar complint el calendari.

Per últim, en el últim bloc, hi han hagut diversos canvis però que estaven previstos des del començament. A l'hora de planificar el calendari, en l'última entrega es van generalitzar les tasques a "Aplicar canvis demanats per el client". En el nou calendari, aquestes tasques s'han desglossat per tal de poder planificar correctament el temps invertit en cadascuna d'elles. També, s'ha afegit al calendari el temps corresponent a la preparació de les presentacions, que s'ha fet de manera paral·lela en el temps corresponent a la redacció de la memòria.

Com a conclusió final del seguiment de la planificació, es pot considerar que s'ha complert de manera molt exitosa el calendari, i s'han gestionat de manera correcte els canvis necessaris que han sorgit durant el desenvolupament del projecte.

6. Pressupost

El següent pressupost, es tracta d'un pressupost estimatiu del cost de desenvolupament del projecte. S'ha de tenir en compte que el número d'hores és una estimació de la realitat, ja que s'ha de tenir en compte que hi ha hagut un procés d'aprenentatge important, horaris no fixes i s'ha invertit molt de temps en la part més teòrica del TFM (redacció de la memòria i altres formalitats acadèmiques).

Tenint això clar, aclarir que el client ja disposa d'un servidor propi, per tant, tots els costos son en hores unipersonals de l'autor del projecte.

CONCEPTE	UNITATS (HORES)	COST PER UNITAT	COST TOTAL
Estudi situació actual i estudi possibilitats	8	€18.00	€144.00
Planificació del Projecte	6	€18.00	€108.00
Disseny Interfície	12	€18.00	€216.00
Adaptació bases de dades	6	€18.00	€108.00
Desenvolupament API REST	40	€22.00	€880.00
Desenvolupament Aplicació Web	120	€22.00	€2,640.00
Desenvolupament Eina Missatgeria Gestors	16	€22.00	€352.00
QA	28	€18.00	€504.00
Desplegament aplicació	8	€18.00	€144.00
TOTAL			€5,096.00
		+IVA	**€6,166.16**

Taula 3 - Pressupost del projecte

7. Estructura de la resta del document

En el CAPÍTOL 2, englobat sota el títol de "Anàlisi", es troba l'estat de l'art. En ell s'analitzen els diferents aspectes rellevants necessaris previs a començar amb el disseny i desenvolupament del producte. En aquest cas, s'ha fet un anàlisi de la competències i diferents anàlisi de les tecnologies necessàries per a desenvolupar cadascun dels blocs del projecte.

El CAPÍTOL 3 es tracta del disseny del projecte. Per una banda, es mostra l'estructura de l'aplicació i l'arquitectura de la informació. Per altra banda, es mostra un llistat detallat de les diferents tecnologies i eines emprades per a realitzar el projecte. Finalment, es troba un apartat de disseny gràfic i interfícies on de manera molt superficial, es mostren els elements visual més destacables.

El CAPÍTOL 4 es la fase del projecte on es tracta el desenvolupament més tècnic. Aquest s'ha dividit en tres blocs corresponents a l'API REST, l'Aplicació Web i l'Eina de Missatgeria per als Gestors. En aquests tres apartats, primer es mostra l'estructura de fitxers, desprès el funcionament general i finalment, es mostra en detall les parts amb contingut més tècnic.

En el CAPÍTOL 5, es troba l'apartat de Validació del projecte. Com el seu nom indica, l'objectiu d'aquest apartat és donar fe del bon funcionament del projecte i deixar constància de les diferents característiques del projecte.

Finalment, el CAPÍTOL 6 tanca la memòria del projecte amb les conclusions i les línies de futur.

CAPÍTOL 2: ANÀLISI

1. Estat de l'Art

Sempre que hi ha una situació extraordinària, com la que hi ha actualment al planeta, hi ha canvis tant per part bona com per dolenta. Els canvis negatius tothom els sap identificar, però a vegades no es dóna prou importància a les noves oportunitats. En la situació actual, s'està vivint una nova etapa de la digitalització.

Tal i com s'ha vist anteriorment, en una economia inestable i sense que ningú tingui clar com avançaran les coses, per a qualsevol empresa el més important és destacar respecte la resta.

En l'àmbit de les gestories, la digitalització interna en els despatxos fa temps que hi està present, però la utilització de les noves tecnologies de cara a oferir millor servei per als seus clients encara està despagant. Aquest tipus de serveis, es centra en una plataforma online on els clients puguin obtenir documentació i informació de forma segura i àgil, i a la vegada poder realitzar gestions a qualsevol hora sense la necessitat del seu assessor.

Tenint això en compte, hi ha dos aspectes importants a analitzar prèviament abans de desenvolupar el projecte:

- Quines eines hi ha actualment al mercat que s'acostin a les necessitats del nostre projecte?

- En cas de desenvolupar una plataforma pròpia, quines tecnologies serien més adients pel desenvolupament del projecte?

1.1. Anàlisi de la competència

Fent un anàlisi del mercat [6][7][8] de portals que permetin facilitar aquesta feina i s'adaptin de manera senzilla a un *Back-End* de bases de dades i arxius en servidors, es poden trobar les següents opcions:

Suasor

Suasor [9] és un ERP [10] per a despatxos i assessories creat per l'empresa Summar Software, que integra diferents funcionalitats per a cobrir totes les necessitats que pugui tenir un despatx. Aquestes funcionalitats van des de serveis per a treball intern a serveis per als clients.

Dintre dels serveis que ofereix Suasor, és el **Portal Multidispositiu Suasor**, el que ofereix les funcions més similars a l'objectiu d'aquest projecte.

Aquest portal permet als usuaris:

- Consultar dades personals emmagatzemades en bases de dades
- Accedir a documents disponibles en un servidor
- Adjuntar documents per a facilitar-los a la gestoria
- Accés directe per a enviar correus electrònics al teu assessor (no disposa de client propi, és un enllaç per a poder accedir a correus com "Outlook", "Correos" o "Gmail").
- Consultar més d'un usuari en que el client en sigui administrador

El portal té un disseny multiplataforma i està disponible en Català i Castellà.
El preu del paquet Suasor varia segons serveis contractats i consta d'un pagament inicial més una quota anual.

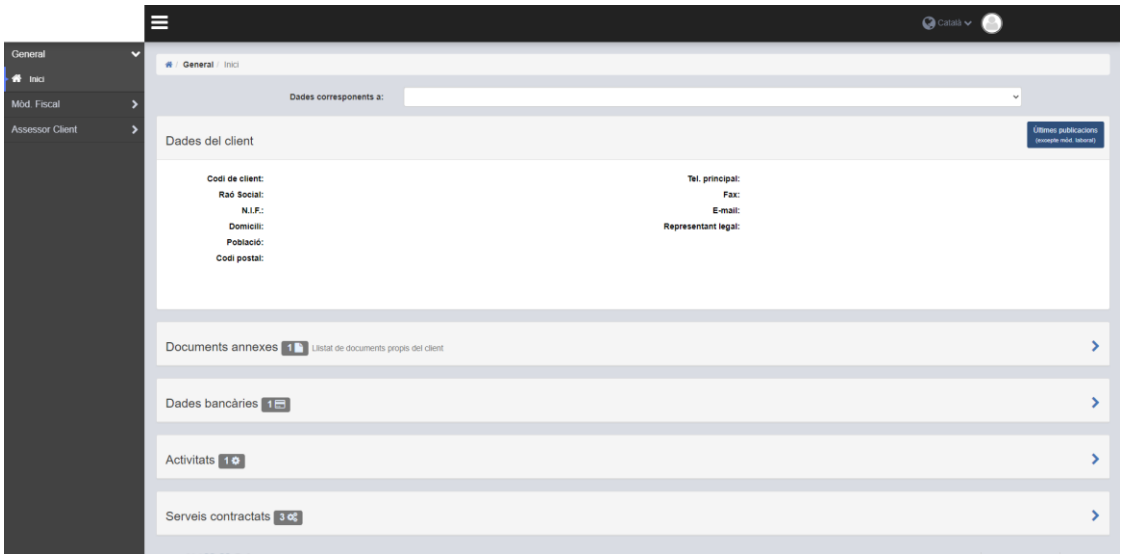


Figura 4 - Suasor portal cliente

Sage Despachos Connected

Sage Despachos Connected [11] és un *software* especialitzat en despatxos i assessories, el qual disposa de diverses funcionalitats per tal que el despatx treballi de la manera més eficient possible i pugui oferir solucions innovadores per als seus clients.

En el catàleg de productes que ofereix Sage, és el seu **Portal de clients** el que és interessant de tenir en compte per analitzar.

El Portal de clients permet als usuaris:

- Consultar dades personals emmagatzemades en bases de dades
- Obtenció d'informes de facturació (Balanços, Tresoreria, etc..) a partir de dades emmagatzemades prèviament
- Accedir a documents disponibles en un servidor
- Adjuntar documents per a facilitar-los a la gestoria
- Visualització de gràfiques a partir de dades financeres prèviament facilitades per el gestor

El portal client té un disseny adaptatiu a les diferents plataformes i està disponible en castellà. El preu del paquet Sage Despachos Connected varia segons els mòduls contractats i després té una quota anual variable segons el número d'usuaris.

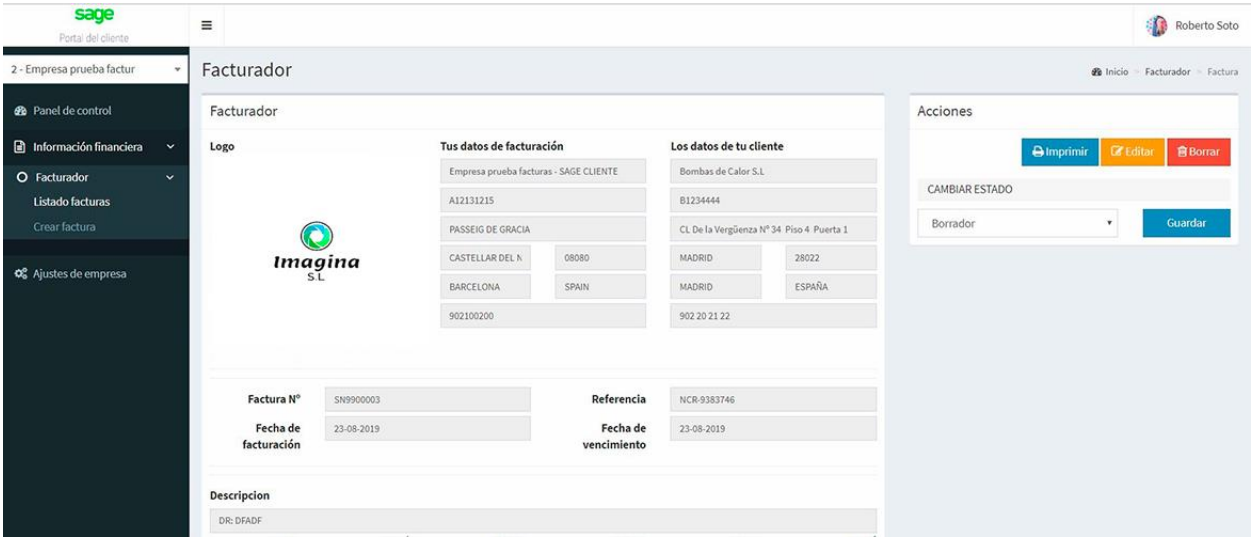


Figura 5 - Sage Despachos Connected

Sudespacho.net

Sudespacho.net [12] és un *software* ubicat en el *cloud* per a despatxos professionals. Segons l'especialitat de cada despatx, disposen de diferents versions de plataforma.

Per a tal de poder fer una comparació efectiva amb el projecte en què s'està treballant, és interessant tenir en compte el **mòdul portal clients**.

El mòdul portal clients permet als usuaris:

- Accedir a expedients facilitats prèviament per el gestor
- Accedir a documents disponibles en un servidor
- Comunicació àgil amb el gestor mitjançant una eina de xat
- Emplenar formularis prèviament facilitats per el gestor
- Accedir a dades de diferents empreses en el cas que el client sigui administrador múltiple

El mòdul portal client té un disseny multiplataforma i també té la versió aplicació en les principals *stores* (Apple Store i Google Play).
Té una versió gratuïta molt limitada, i després té diferents plans segons nivell de negoci amb un pla de pagament mensual (no té pagament inicial).

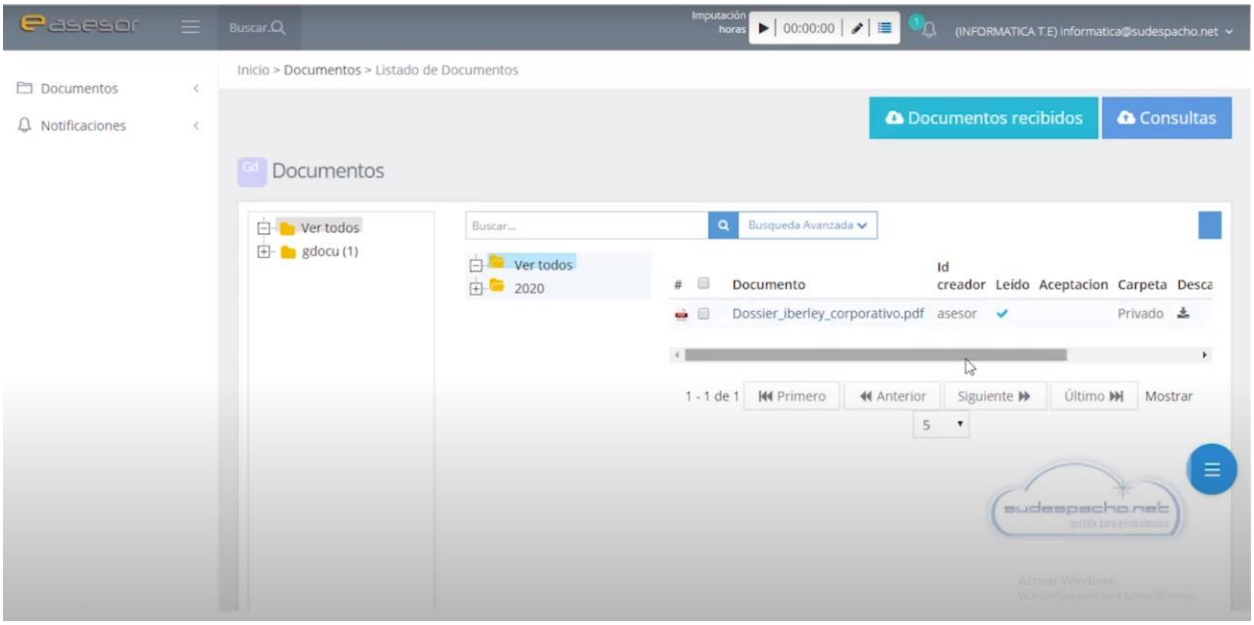


Figura 6 - Sudespacho.net

Com es pot observar, les diferents plataformes presentades tenen la finalitat de facilitar l'accés a dades per part dels clients i en alguns casos també ofereixen algun altre servei extra. Ara bé, com a punt comú, podem trobar que en certa manera presenten una navegació complicada i son molt tancades a nivell d'escalabilitat.

Per al client, és essencial que pugui accedir a la informació que necessita amb el mínim de passos possibles, i en les diferents plataformes que s'han vist resulta mitjanament complicat filtrar la informació.

Com s'ha comentat prèviament, el punt més important, és l'escalabilitat i personalització de la plataforma. Les plataformes presentades, es poden personalitzar mostrant o amagant opcions que ofereixen com a mòduls. No obstant, en el cas de voler implementar funcions pròpies, resulta molt complicat o directament no és possible. Si tenim en compte que l'objectiu del client és tenir un punt diferencial respecte la competència, resulta difícil d'aconseguir-ho mitjançant una aplicació "prefabricada".

És per aquests motius, que el desenvolupament d'una plataforma de gestió de clients pròpia és una molt bona opció, en comptes d'utilitzar una solució comercial ja en el mercat.

1.2. Anàlisi tecnologies desenvolupament: Aplicació Web

Tenint en compte l'experiència i coneixements de l'autor del projecte, es comparen diferents tecnologies que tinguin com a base el llenguatge **Javascript**. A partir d'aquí, es poden trobar una gran quantitat de llibreries i *framework*, però per a marcar una limitació, l'anàlisi es centrarà en **VueJS**, **Angular** i **ReactJS**.

Figura 7 - Comparativa tecnologies (logos)

Vue.js

Vue.js [13] és un *framework* de codi obert escrit en JavaScript enfocat per al desenvolupament d'interfícies d'usuari i aplicacions de pàgina única (SPA) [14].

Dintre de les característiques a destacar, trobem:

- Té una corba d'aprenentatge relativament senzilla
- L'arquitectura del *framework* ajuda a mantenir un projecte net i organitzat
- Amb un únic arxiu .vue es pot escriure html i css sense necessitat de passar per un arxiu de javascript
- Recentment, està agafant molta força i cada cop té més comunitat

Angular

Angular [15] és un *framework* de codi obert desenvolupat per Google en llenguatge TypeScript, enfocat en el desenvolupament d'aplicacions de pàgina única.

Dintre de les característiques a destacar, trobem:

- Dintre el sector, destaca per oferir un gran rendiment en SPA de gran exigència
- Compta amb un CLI [16] molt potent que permet començar a desenvolupar ràpidament
- L'arquitectura està enfocada en crear projectes modulars
- Permet adaptar-se de manera molt senzilla a altres tecnologies

ReactJS

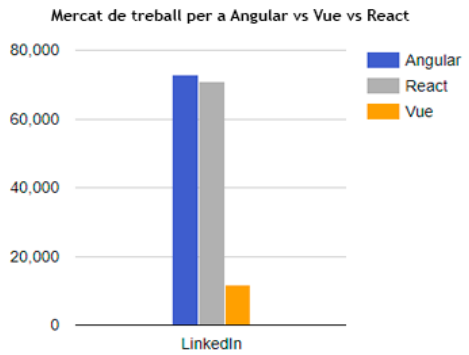
ReactJS [17] és una biblioteca [18] Javascript de codi obert desenvolupat per Facebook, enfocat en el desenvolupament d'aplicacions de pàgina única.

Dintre de les característiques a destacar, trobem:

- Es treballa amb composició de components en comptes de composició funcional
- La propagació d'informació és unidireccional i jeràrquica
- Utilitza una DOM virtual per oferir un gran rendiment de renderitzat
- Té una comunitat molt gran

Havent vist les diferents opcions més populars, es pot veure que tenen unes característiques bastant semblants, i no hi ha una resposta única per a decidir entre un i altre.

Tenint en compte que per a desenvolupar el projecte l'autor necessita passar per un procés d'aprenentatge, el fet de que hi hagi una gran comunitat i que hi hagi molta documentació, és essencial. També, de cara al futur, les ofertes de feina és un plus a tenir en compte.



Estadístiques de GitHub Angular vs Vue vs React

	Angular	React	Vue
# Watchers	3.2k	6.7k	6.2k
# Stars	62k	151k	166k
# Forks	16.9k	29.4k	25.2k
# Contributors	1,129	1,390	293

Font: codeinwp / Angular vs React vs Vue: Which Framework to Choose in 2020 bt Shaumik Daityari

Figura 8 - Comparativa frameworks (Dades)

A partir de les dades que podem veure en aquesta comparació [19], la tecnologia més interessant per a desenvolupar el projecte és **ReactJS**.

1.3. Anàlisi tecnologies desenvolupament: API REST

En la part del Back-End de la plataforma, l'objectiu del projecte és desenvolupar una API REST [20] encarregada de gestionar les peticions http fetes per part de l'aplicació web. Al igual que en el cas del Front-End, per aprofitar les aptituds i interessos de l'autor del projecte, aquesta API REST es vol desenvolupar amb **Node JS** (un entorn que utilitza **Javascript**). Tot i que les opcions que ofereix el mercat dintre aquest àmbit són molt grans, per a delimitar l'anàlisi, aquest es centrarà en **Express**, **Koa** i **Hapi**.

Figura 9 - Comparativa tecnologies Back-End (logos)

Express

Express [21] és un entorn de treball per a aplicacions creades amb Node.js, enfocat per al desenvolupament d'API's.

Dintre de les característiques a destacar, trobem:

- Té molts anys de trajectòria i es troba en una etapa molt madura (la primera versió va ser publicada el 3 de gener del 2010)
- La seva simplicitat, recorda a treballar amb entorns natius de Javascript
- Està disponible en els principals distribuïdors de paquets Node (npm i yarn)
- És molt senzill començar un projecte nou (amb unes 5 línies, es pot tenir el servidor funcionant)
- Actualment, el projecte està mantingut per la "Node.js Foundation Incubator"

Koa

Koa [22] és un *framework* de nova generació, enfocat en el disseny d'API's web basades en Node.js.

Dintre de les característiques a destacar, trobem:

- Està desenvolupat per membres de l'equip de "Express", amb l'objectiu de millorar-lo
- Treballa amb mòduls (a diferència de Express que treballa amb *templates*)
- Utilitza Ecmascript 6 [] (ES2015)
- És molt senzill començar un projecte nou (amb unes 5 línies, es pot tenir el servidor funcionant)
- És un projecte molt lleuger a nivell d'espai (té una extensió aproximada de 500 línies de codi)

Hapi

Hapi [23] és un *framework* dissenyat per al desenvolupament d'API's web, centrat en l'escalabilitat dels projecte i la seguretat.

Dintre de les característiques a destacar, trobem:

- Disposa d'una gran quantitat de *plugins* per a treballar diferents aspectes de la API (per exemple, en comptes de programar un *middleware* es pot utilitzar un *plugin*)
- Està enfocat a la llegibilitat (tot i que això signifiqui incrementar el número de línies de codi necessàries)
- Està pensat per a grans projectes
- Incorpora de manera nativa un sistema de Autorització i Autenticació

Havent vist les característiques de les tres opcions, es pot coincidir que totes elles es poden adaptar a les necessitats del projecte. Per a decidir-se entre elles, s'ha tingut en compte la quantitat de documentació i comunitat de cadascuna d'elles.

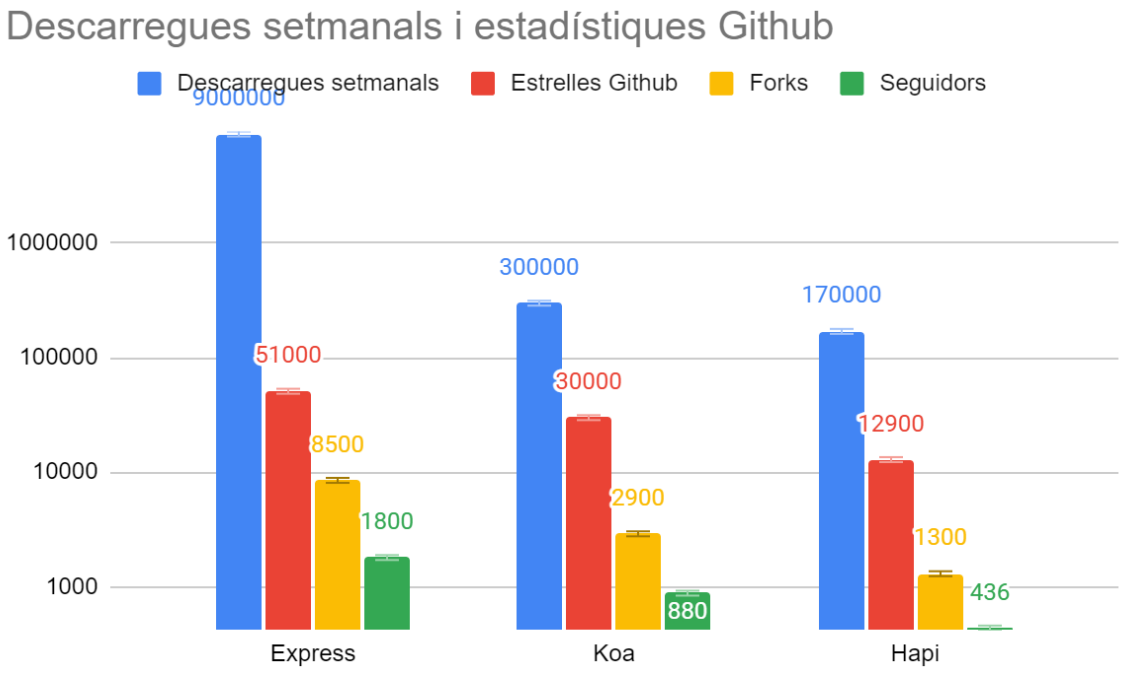


Figura 10 - Gràfic de tecnologies back-end (GitHub)

Tal i com es pot observar, a nivell d'ús i de possibles recursos, Express domina el mercat de manera significativa. És per aquest motiu, que tenint en compte els objectius principals del TFM, s'ha optat per utilitzar **Express** per a desenvolupar l'API REST del projecte.

1.4. Anàlisi tecnologies desenvolupament: Eina comunicació assessors

Un dels mòduls a desenvolupar en la part de l'aplicació web (client), és un sistema de missatgeria no instantània ("Mur assessor-client"). Per la part dels usuaris, aquest mòdul va totalment integrat a l'aplicació a desenvolupar, però per als treballadors de l'assessoria, s'ha decidit desenvolupar una petita eina externa per a poder gestionar tots els missatges de manera més còmode i efectiva.

L'objectiu d'aquesta eina, és que sigui una petita aplicació molt senzilla, on els assessors es puguin connectar amb la seva compta d'usuari i allà vegin en mida gran i organitzat, els diferents missatges dels clients. Tenint en compte aquesta simplicitat i la preferència d'usar **Javascript,** s'ha elaborat una llista amb tres opcions: **MooTools**, **jQuery** i **Dojo**.

Figura 11 - Comparativa frameworks (logos)

Mootools

Mootools [24] és un *framework* orientat a objectes per a Javascript, pensat en facilitar la interacció amb el codi html.

Dintre de les característiques a destacar, trobem:

- És modular (el desenvolupar pot elegir quines parts del *framework* vol incloure en el projecte)
- És un *framework* orientat a objectes
- Disposa d'una gran quantitat de funcions per a continguts visuals (efectes)
- Simplifica les interaccions amb la DOM

jQuery

Jquery [25] és una biblioteca de Javascript, que permet d'una manera molt senzilla, interaccionar amb els elements de la DOM i els diferents elements *html*.

Dintre de les característiques a destacar, trobem:

- Té una corba d'aprenentatge molt senzilla
- Ofereix una manipulació molt senzilla sobre qualsevol element de la DOM
- Es complementa de manera molt eficient amb AJAX
- És compatible amb pràcticament tots els navegadors

Dojo

Dojo toolkit [26] és una biblioteca modular de Javascript, dissenyada per a facilitar el desenvolupament ràpid d'aplicacions i llocs webs basats en Javascript.

Dintre de les característiques a destacar, trobem:

- És modular (el desenvolupar pot elegir quines parts del *framework* vol incloure en el projecte)
- Es complementa de manera molt eficient amb AJAX
- Té una gran quantitat de *Widgets multi-browser*, que ofereixen solucions prefabricades com menús, gràfiques dinàmiques, etc.
- Té un sistema d'emmagatzematge propi per la banda de client (com un localStorage)

Com s'ha pogut veure, cadascun té algun punt característic respecte la resta, però en general són molt similars. Per a prendre una decisió sobre quin d'aquests frameworks es vol utilitzar, s'ha avaluat la comunitat de cadascun d'ells.

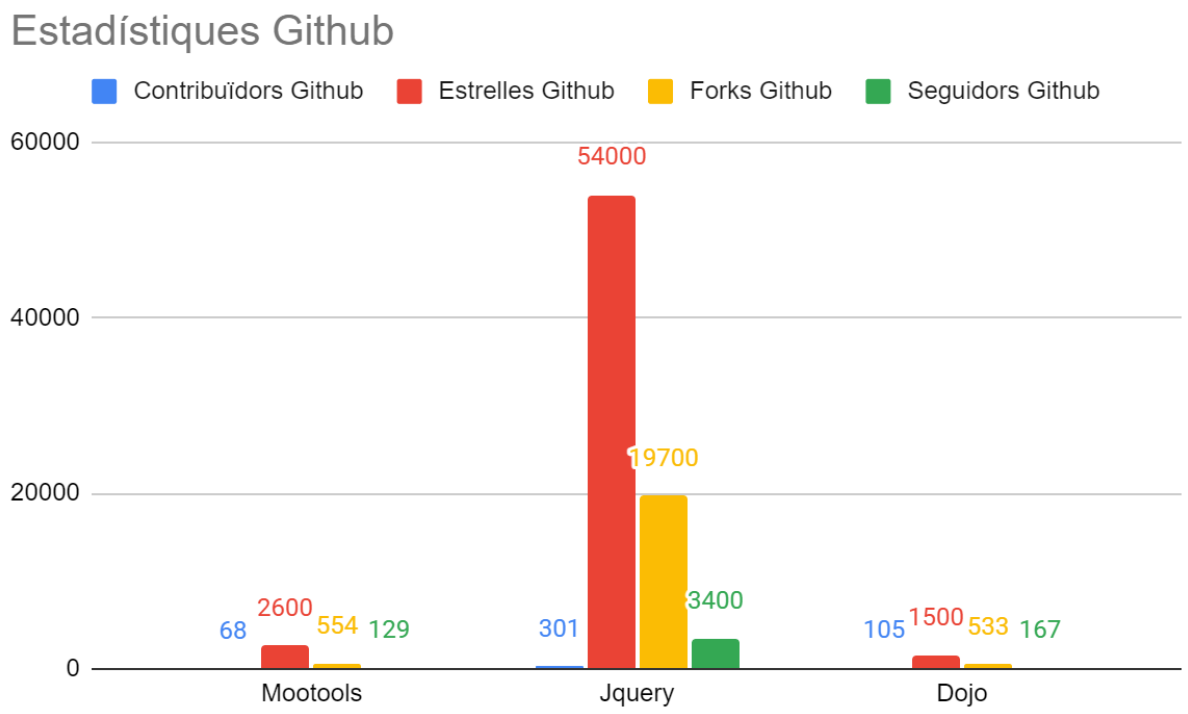


Figura 12 - Gràfic frameworks (GitHub)

Tenint en compte aquestes dades, per a desenvolupar aquesta eina s'ha decidit utilitzar **jQuery**.

1.5. Conclusions dels anàlisi

Després d'haver analitzat el mercat actual en el àmbit de plataformes web per a la gestió de client i les diferents opcions per a desenvolupar-ne una de pròpia, s'han pogut extreure unes conclusions generalitzades.

El primer que s'ha de destacar, és que actualment al mercat ja s'hi poden trobar diferents opcions que en certa manera cobreixen la necessitat existent. No obstant, com s'ha vist reflectit amb els resultats de l'anàlisi, totes les opcions resulten ser molt tancades a nivell de personalització i visualment no s'han adaptat a l'estil del segle 21. També, cal destacar els alt costos que suposen, ja que la majoria d'elles compten d'un pagament inicial i desprès una quota periòdica o per ús. Cal recordar, que per l'anàlisi s'han seleccionat les tres opcions que més s'aproximen a nivell de característiques al projecte d'aquest Treball de Final de Màster, si es fa una visió més general, encara s'allunyen més.

Tenint això en compte, s'ha arribat a la conclusió que tot i que en el mercat ja hi hagi moltes opcions disponibles, no hi ha cap opció que cobreixi a la totalitat les necessitat d'aquest projecte.

Per altra banda, en quant a l'anàlisi de tecnologies de desenvolupament, s'ha vist clarament que les opcions son gairebé infinites. Degut al requisit previ de l'autor del projecte en buscar llenguatges on es puguin aprofitar al màxim possibles els seus coneixements previs, s'ha pogut limitar en certa manera aquesta cerca. Tot i això, després s'ha hagut d'acabar aplicant diversos criteris (popularitat, ofertes de treball, etc..) per acabar tancant una llista de tres opcions.

En el cas del *framework* de desenvolupament per el sistema de missatgeria i per l'API REST, en ambdós casos s'ha acabat escollint l'opció més popular per diferència. En part, la seva popularitat ve per què ofereixen moltes més facilitats (a nivell de *framework* i de recursos) que tota la seva competència. Molt probablement, una gran part de desenvolupador que han treballat tant amb Express com jQuery, no han sentit mai ni anomenar les altres opcions que s'han analitzat (i s'ha de tenir en compte, que s'han comparat amb la resta d'opcions més populars).

A diferència d'aquests dos casos, en el cas del *framework* per a desenvolupar l'aplicació web, aquest és un mercat molt més actiu i amb competència real (tant a nivell de característiques de *framework* com de recursos disponibles). Totes les opcions son àmpliament conegudes i totes compten amb gran quantitat de recursos i sortides laborals. També, altres opcions que s'han quedat fora del anàlisi, hi haguessin pogut entrar perfectament.

CAPÍTOL 3: DISSENY

1. Arquitectura de l'Aplicació

L'arquitectura d'un sistema [27], defineix quins patrons i abstraccions coherents s'han fet servir com a guia durant el desenvolupament d'un programari.

L'aplicació en que basa l'elaboració d'aquest Treball de Final de Màster, utilitza un patró de desenvolupament MVC [28][29] (*Model-View-Controller*). Aquest patró de desenvolupament, comunament emprat en el desenvolupament d'interfícies d'usuari, es caracteritza en dividir l'aplicació en tres parts diferenciades: el model de dades, la interfície d'usuari i la lògica de control.

- **Model:** És la capa on es treballa amb les dades, per tal cosa, aquesta conté mecanismes per a poder accedir a informació i també per a actualitzar el seu estat. En el cas d'utilitzar una base de dades, és en el model a on es realitzen les consultes com les cerques, filtratges i actualitzacions.

- **Vista:** És la capa encarregada de mostrar la interfície d'usuari, és a dir, gestionar el codi que fa possible la renderització de l'aplicació HTML. En certa manera, es pot dir que es la principal encarregada de gestionar el Front-End.

- **Controlador:** És la capa responsable a fer d'enllaç entre la capa Model i la capa Vista. Aquesta capa, s'encarrega de gestionar les instruccions utilitzades per comunicar les dos capes i processar-les per a la optima comunicació.

Tal i com s'ha vist prèviament, l'aplicació consta d'un model de tres capes [30] que ha estat implementat en una arquitectura de dos nivells. En el nivell de "Servidor", hi ha la lògica de negoci (Model) i l'intermediari (Controlador). Per altra banda, en el nivell del client, s'hi troba la part necessària perquè l'usuari pugui interactuar amb el sistema (Vista).

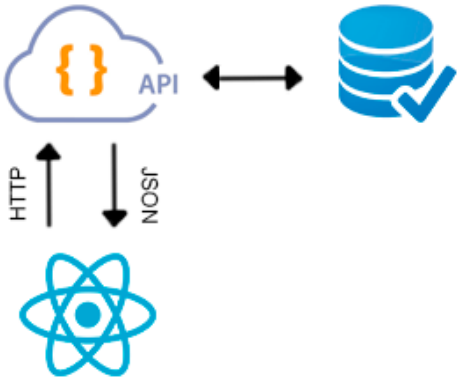


Figura 13 - Esquema comunicació client a DB

En el cas concret de l'aplicació que s'està desenvolupant en aquest TFM, es disposa d'una API REST fent la funció de controlador i que s'encarrega de gestionar les comunicacions entre la vista (ReactJS) i les dades. Aquesta comunicació, es duu a terme mitjançant peticions HTTP i respostes en format JSON, que posteriorment la vista les processarà per mostrar-les correctament per pantalla.

2. Arquitectura de la informació i diagrames de navegació

2.1. Model de dades:

Una de les parts que composa aquest projecte son les bases de dades, no obstant, en la seva gran majoria aquestes les ha proporcionat el client. El client disposa d'una gran quantitat de taules on es pot trobar tot tipus de dades, però per el desenvolupament del projecte, no totes són necessàries.

Tant per poder treballar en un entorn de desenvolupament segur (l'autor del TFM no té experiència prèvia treballant amb bases de dades) i per poder mantenir en tot moment la privacitat dels clients de la gestoria, s'ha pres la decisió de replicar en un entorn local les diverses taules que siguin necessàries per al projecte (retallant en alguns casos les columnes no necessàries).

És per això, que el diagrama que es mostra a continuació, s'ha realitzat només tenint en compte les taules que s'han replicat. En un futur, és possible que se'n necessitin més per incorporar nous tipus d'informació, però aquestes seguiran el mateix esquema.

A continuació es mostra un diagrama simplificat de les taules implicades en la gestió del producte:

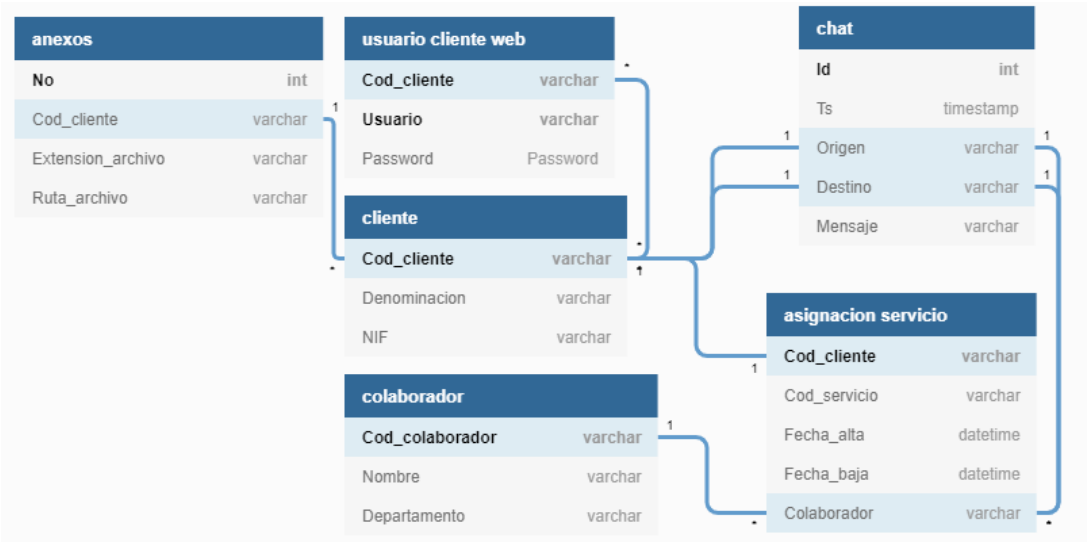


Figura 14 - Diagrama base de dades

2.2. Arbre de navegació:

Amb l'objectiu de mostrar de manera gràfica quina és l'estructura de navegació de l'aplicació, a continuació es mostra un arbre de navegació de l'aplicació [31]. Els mapes de navegació són una representació gràfica de l'estructura de navegació d'una aplicació o lloc web, amb els que es pot visualitzar de manera general i esquemàtica quina informació s'ofereix a l'usuari i com està distribuïda.

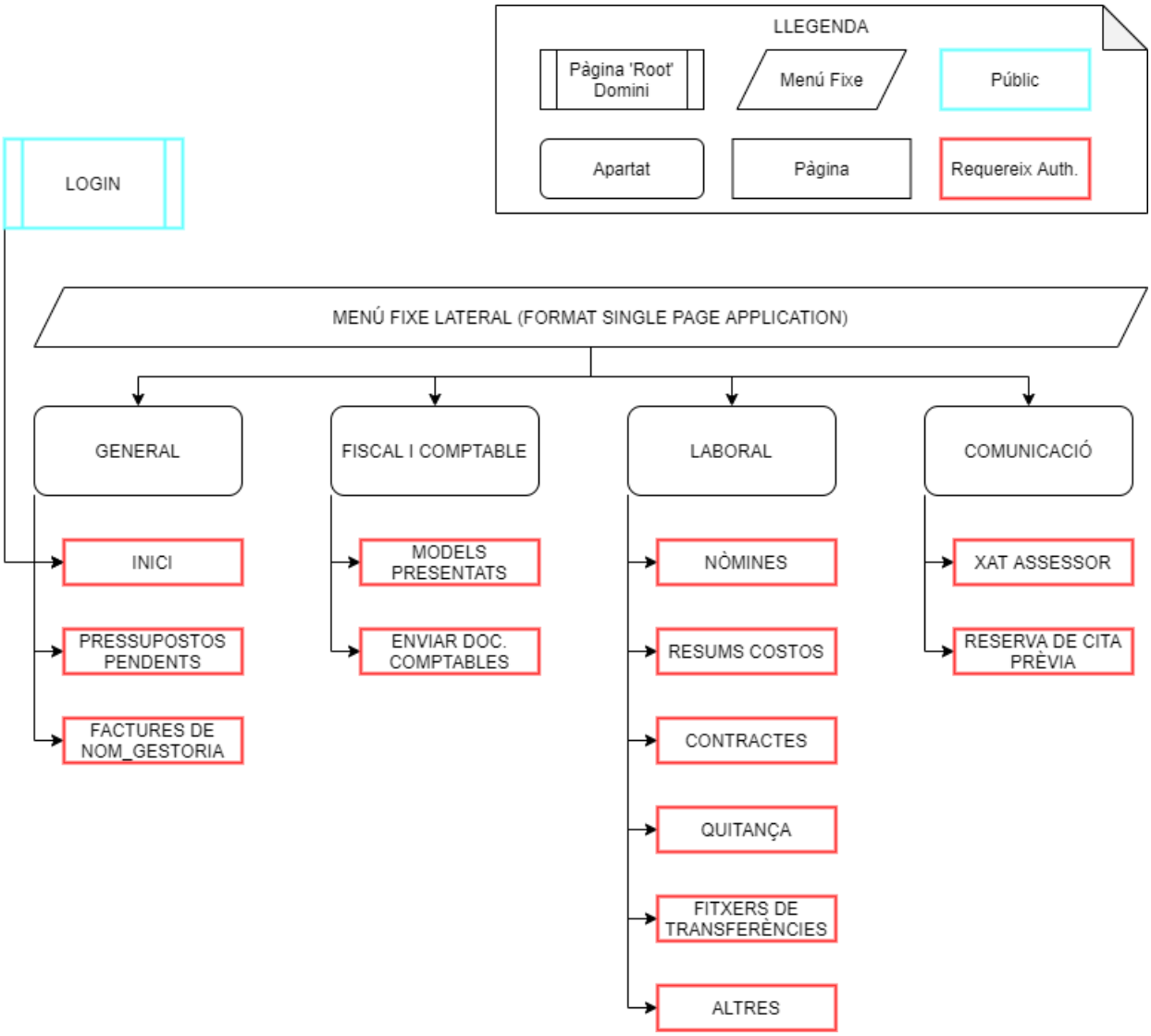


Figura 15 - Arbre de navegació de l'Aplicació

2.3. Especificació dels mòduls:

Per tal d'organitzar l'aplicació, s'ha dividit en la pàgina d'accés i en diferents mòduls que estan compostos de diferents pàgines en el seu interior.

Login

És la pàgina d'accés a l'aplicació i ocupa la posició d'arrel del domini. És l'únic contingut públic de l'aplicació, doncs els usuaris l'han d'utilitzar per a poder accedir al contingut protegit sota autorització.

Mòdul General

Un cop l'usuari entra accedeix a l'aplicació a través de la pàgina d'accés, aquest és redirigit a la pàgina d'inici que es troba dintre el mòdul general. Aquest mòdul, té com a objectiu oferir a l'usuari informació ràpida i també aquella informació, relativa a la gestoria.

- **Inici:** És la pàgina principal de l'aplicació. Aquesta consta de quatre apartats diferents: una *feed rss* [32] on es mostren les últimes publicacions del *wordpress* de la gestoria, un petit menú on hi consten quatre botons per a realitzar tasques comuns de manera ràpida, un apartat on es mostren les dades personals de l'usuari i una última zona on l'usuari pot visualitzar algunes ràtios referents a les seves dades.

- **Pressupostos pendents:** En aquesta pàgina, l'usuari hi pot trobar els diferents pressupostos sobre projectes o tasques que ha encarregat a la gestoria.

- **Factures de Gestoria:** En aquesta pàgina, s'hi mostren les diferents factures generades per part de la gestoria referents als serveis prestats al client.

Mòdul Fiscal i Comptable

Aquest mòdul està enfocat a la fiscalitat i comptabilitat.

- **Models presentats:** En aquesta pàgina, l'usuari pot consultar diferents models fiscals que s'han presentat i se'ls pot descarregar. Entre els models, s'hi troben per exemple el 303, el 111, el 115, etc.

- **Enviar documents comptables:** Aquesta pàgina està enfocada en la comunicació de client cap al gestor. En ella, l'usuari pot carregar i enviar documents que li hagin sigut sol·licitats per el seu gestor.

Laboral

Aquest mòdul principalment està enfocat a petits empresaris o autònoms, ja que des d'aquí podran consultar i gestionar diversos documents de caràcter laboral.

- **Nòmines:** En aquesta pàgina, l'usuari pot consultar i descarregar-se les nòmines que la gestoria li ha preparat.

- **Resum costos:** Aquesta pàgina permet a l'usuari consultar diverses ràtios i resums de costos del seu exercici, extrets a partir de les diferents dades que han estat proporcionades prèviament a la gestoria.

- **Contractes:** En la pàgina de contractes, l'usuari pot consultar i descarregar els contractes dels seus treballadors o persones a qui ha empleat de manera temporal.

- **Quitança:** En aquesta pàgina, l'usuari pot consultar i descarregar-se la documentació referent a les quitances (conegut popularment com el "*finiquito*").

- **Fitxers de transferència:** A l'apartat de Fitxers de transferència, l'usuari hi pot consultar i descarregar extractes referents a pagaments bancaris.

- **Altres:** Aquest apartat queda reservat a documentació variada referent a temes laborals. En general, aquí s'hi troben tots aquells arxius laborals que no estan indicats en algun dels altres apartats.

Comunicació

Tal i com indica el nom, aquest mòdul està enfocat a assumptes de comunicació entre el client i el seu assessor.

- **Xat assessor:** En aquesta pàgina, el client pot comunicar-se de manera directa amb el seu assessor. El xat no és a temps real, si no que és un "mur de missatges" ordenat en xat, de manera que quan alguna de les dos parts vol contestar, el missatge s'afegeix a una conversa ordenada (que queda guardada com un xat).

- **Reserva de cita prèvia:** Aquesta pàgina permet a l'usuari demanar hora per a visitar el seu assessor. La reserva de cita, es gestiona amb el servei de calendari de *office365*.

3. Tecnologies de desenvolupament

Per al desenvolupament d'aquest Treball de Final de Màster, s'han utilitzat diferents eines, llenguatges i tecnologies.

3.1. Tecnologies Front-End:

- **ReactJS** [33]**:** És una biblioteca de Javascript de codi obert enfocada en el desenvolupament d'interfícies d'usuari i les aplicacions d'una sola pàgina. ReactJS està mantingut per Facebook i una gran comunitat de programari lliure.

- **HTML** [34]**:** És el principal llenguatge utilitzat en el desenvolupament de continguts per a navegador. HTML és un llenguatge de marcat, el qual permet estructurar textos i relacionar-los en forma d'hipertext, per a poder ser visualitzats com a contingut web.

- **Javascript** [35]**:** És un llenguatge de programació orientat a objectes, lleuger i interpretat, dissenyat principalment per a poder implementar funcions complexes en pàgines web.

- **CSS3** [36]**:** És la tercera aproximació del llenguatge CSS (*Cascading Style Sheets*), un llenguatge de fulls d'estil utilitzat per a determinar la presentació de contingut escrit en llenguatge de marcat com el HTML. Les especificacions CSS estan sota el control de W3C (*World Wide Web Consortium*), un consorci internacional que treballar per a marcar els estàndards per al *World Wide Web* (www).

- **SASS** [37]**:** És un metallenguatge de nivell superior a CSS utilitzat per a poder escriure de manera neta i estructurada un full d'estils. Per a fer-ho, aquest dóna la possibilitat d'escriure en forma aniuada, utilitzar variables, mixins, herència de selecció i moltes altres facilitats.

- **NPM** [38]**:** És un sistema de gestió de paquets per a Node.js, un entorn d'execució per a Javascript. Consisteix en facilitar l'accés a una gran quantitat de dependències, tant de lliures com de privades sota pagament, mitjançant l'ús d'una línia de comandes (CLI).

- **Axios** [39]**:** És una una llibreria de Javascript, que facilita la feina a l'hora de duu a terme operacions HTTP. De manera resumida, la llibreria Axios és una API per a gestionar de manera còmoda i segura les operacions en Ajax.

- **jQuery** [40]**:** És una biblioteca de Javascript, creada per a facilitar la interacció amb els elements HTML. D'una manera molt senzilla, aquesta ofereix la possibilitat de manipular elements del DOM, gestionar esdeveniments, desenvolupar animacions, etc.

3.2. Tecnologies Back-End:

- **Node.js** [41]**:** És un entorn de programació enfocat per el desenvolupament d'arquitectures de servidor, dissenyat per a escriure aplicacions d'Internet escalables. La seva arquitectura està orientada a esdeveniment, essent el Javascript el seu llenguatge d'escriptura.

- **Express.js** [42]**:** És un entorn de treball per a aplicacions web per al programari Node.js, de codi obert i amb llicència MIT. El seu ús, es basa en el desenvolupament d'aplicacions web i APIs.

- **JSON** [43]**:** És un estàndard obert basat en text, dissenyat amb l'objectiu de duu a terme intercanvis de dades que puguin ser llegibles per els humans. Com el seu nom indica, *JavaScript Object Notation* (JSON), la seva estructura són objectes que segueixen l'estructura del llenguatge Javascript.

- **JWT** [44]**:** És un estàndard obert basat en JSON, utilitzat per a la creació de tokens d'accés que permeten gestionar de manera segura la identitat i privilegis dels usuaris en una relació servidor-client. El seu acrònim fa referència a *JSON Web Token*, que vindria a significar "moneda" web en format JSON.

- **Bcrypt** [45]**:** És una funció de *Hash* basada en el algoritme "Blowfish cipher". Aquesta s'ha fet molt popular per el fet d'incorporar un *Salt* (conjunt de bits aleatoris) per a protegir-se contra les *Rainbow Tables*.

- **SQL** [46]**:** És un llenguatge estàndard de comunicació amb bases de dades relacionals. El seu ús és molt extens, per la seva facilitat de ser utilitzat dintre d'altres llenguatges de programació. Una de les seves característiques principals, és la seva senzillesa deguda a la similitud en el llenguatge humà: "CREATE", "SELECT", "DELETE"..

- **MySQL** [47]**:** És un sistema de gestió de bases de dades relacionals creat per IBM, havent aconseguit gran popularitat gràcies a la seva versatilitat amb PHP, que usa el llenguatge SQL.

3.3. Altres:

- **Visual Studio Code** [48]**:** És un editor de codi font, desenvolupat per Microsoft, disponible per a Windows, Linux i MacOS. Gràcies a la gran quantitat de complements disponibles, aquest s'adapta de manera eficaç a gairebé qualsevol llenguatge de programació.

- **Git** [49]**:** És un programari de control de versions dissenyat per al manteniment de les versions en aplicacions amb una gran quantitat de fitxers de codi font. El seu propòsit, és portar un registre dels diferents canvis en els arxius que comprenen el codi d'un projecte i facilitar la coordinació dels membres en un grup de treball.

- **GitHub** [50]**:** És un sistema d'allotjament de repositoris Git, oferint als usuaris una interfície gràfica per a duu a terme les diferents opcions que ofereix Git i afegint altres de pròpies. També ofereix diferents serveis per a grups de treball, com el registre de *bugs*, administració de tasques i *wikis*.

- **Heroku** [51]**:** És una plataforma per a la gestió de serveis en el núvol. Aquesta conta amb un gran públic per la gran quantitat de llenguatges de programació que suporta i també per la facilitat de complementar els projectes amb *plugins* oferts per la mateixa plataforma: Bases de dades, serveis de monitorització, *firewalls*, optimització d'imatge, etc.

- **Xampp** [52]**:** És un paquet de programari lliure que ofereix la possibilitat de crear un servidor HTTP Apache, una base de dades de MySQL, accés a FileZilla i altres eines que ajuden al desenvolupament en local d'un projecte amb servidor.

- **Phpmyadmin** [53]**:** És una eina d'administració de codi lliure, que ofereix una interfície gràfica per a gestionar bases de dades com MySQL i MariaDB.

- **Postman** [54]**:** És una aplicació que permet a l'usuari crear de manera senzilla peticions HTML. Durant el procés de desenvolupament d'una API, aquest és de gran ajuda ja que l'usuari no necessita dependre d'un client Front-End per a comprovar el bon funcionament del seu codi.

- **Adobe Photoshop** [55]**:** És un editor de gràfics desenvolupat per Adobe Systems, utilitzat principalment en el sector del retoc fotogràfic i arts gràfiques.

- **Zoom** [56]**:** És una plataforma que permet realitzat videoconferències de manera senzilla. Aquesta està disponible en la majoria de plataformes, i també es pot utilitzar sense necessitat d'instal·lació mitjançant un navegador web.

- **DBeaver** [57]**:** és una aplicació de programari de client SQL i una eina d'administració de bases de dades, gratuïta i de codi obert.

4. Disseny gràfic i interfícies

En aquest apartat, es pretén deixar constància sobre quines guies a nivell gràfic i de disseny s'han utilitzat per a crear la part visual de l'aplicació.

4.1. Patró de disseny

Un dels requisits principals de l'aplicació, és que aquesta pugui ser visualitzada en diferents tipus de dispositius. És per això, que per al desenvolupament de l'aplicació s'ha utilitzat un patró de disseny web *responsive*. En concret, s'ha optat per a utilitzar el patró "Mostly Fluid"[58].

Aquest patró consisteix, tal i com el seu nom ens ve a indicar, en una quadrícula "fluida". En les pantalles més grans, es manté la mida del elements i es centren al mig de la pantalla. En el cas de les pantalles més petites, es re-processa el contingut principal i les columnes passen a apilar-se verticalment. Una de les principals avantatges d'aquest disseny, és que en la majoria de casos amb un sol punt d'interrupció es pot duu a terme tot el re-posicionament de la interfície.

A continuació, es pot observa un exemple simplificat de com funciona el patró "Mostly Fluid":

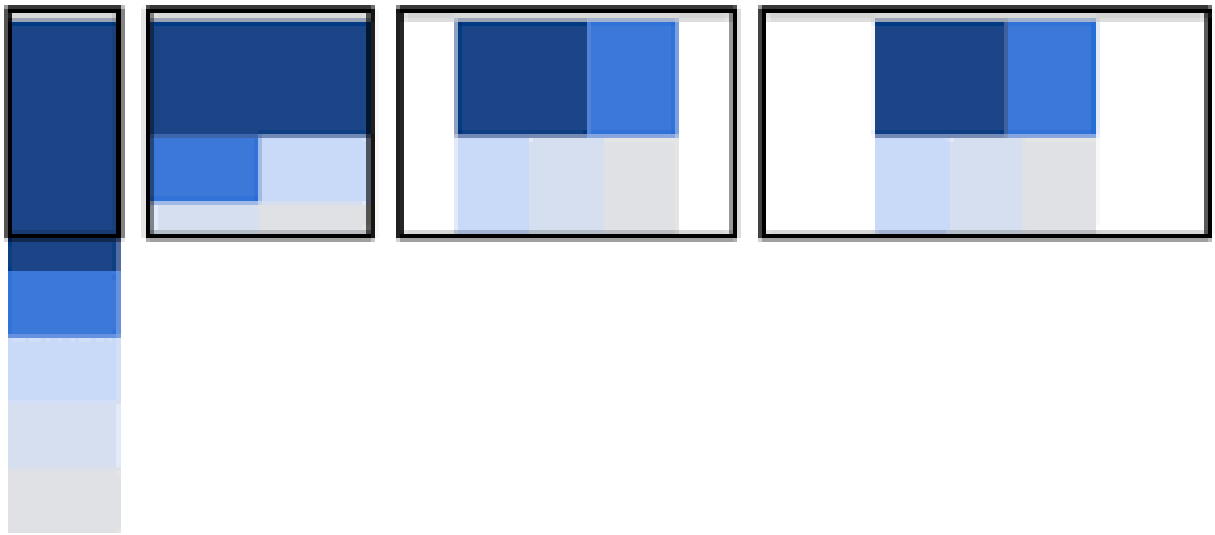


Figura 16 - Exemple de Mostly Fluid. FONT: Google Developers

Tal i com es pot observar en aquesta imatge, els diferents apartats s'acaben apilant de forma lògica (com si fos un fluid), mentre que en el cas de les pantalles més grans, aquests no es veuen alterats i simplement s'afegeixen marges en els laterals.

4.2. Estils

Font:

Com a fonts del projecte, s'ha optat per usar la família de fonts de Roboto [59][60], desenvolupada per Google i creada per Christian Robertson.

Aquesta es caracteritza per tenir un estil "modern", però al mateix temps un toc clàssic, ja que pertany al gènere tipogràfic de *sans-serif* [61]. Tota la família de fonts està sota la llicència Apache [62] i és totalment gratuïta des del 12 de gener del 2012.

Les variants usades en el projecte, són les següents:

Regular 400

Almost before we knew it, we had left the ground.

Regular 400 italic

Almost before we knew it, we had left the ground.

Bold 700

Almost before we knew it, we had left the ground.

Figura 17 - Estils de fonts del projecte

Paleta de colors:

Com a base de la paleta de colors, s'ha tingut en compte el color central, en que està basada tota la pàgina web informativa de la gestoria (#003366). A partir d'aquí, s'ha creat una paleta amb la qual es puguin resoldre les diferents necessitats per a l'aplicació que s'està desenvolupant.

Figura 18 - Paleta de colors del projecte

Botons:

Per als botons de la plataforma, s'ha optat per una versió simple i que compleixi amb la paleta de colors proposada.

A continuació, es poden veure els estats de "normal", "hover" i "bloquejat".

Figura 19 - Exemple d'estils de botons del projecte

Icones:

Per als icones del projecte, s'han utilitzat tres llibreries diferents: Font Awesome [63], TypIcons [64] i HeroIcons [65]. En totes elles, s'ha utilitzat la versió gratuïta que ofereixen una llicència totalment lliure d'ús en projectes amb finalitat comercial. Les tres llibreries estan disponibles en el repositori de "React-icons" [66].

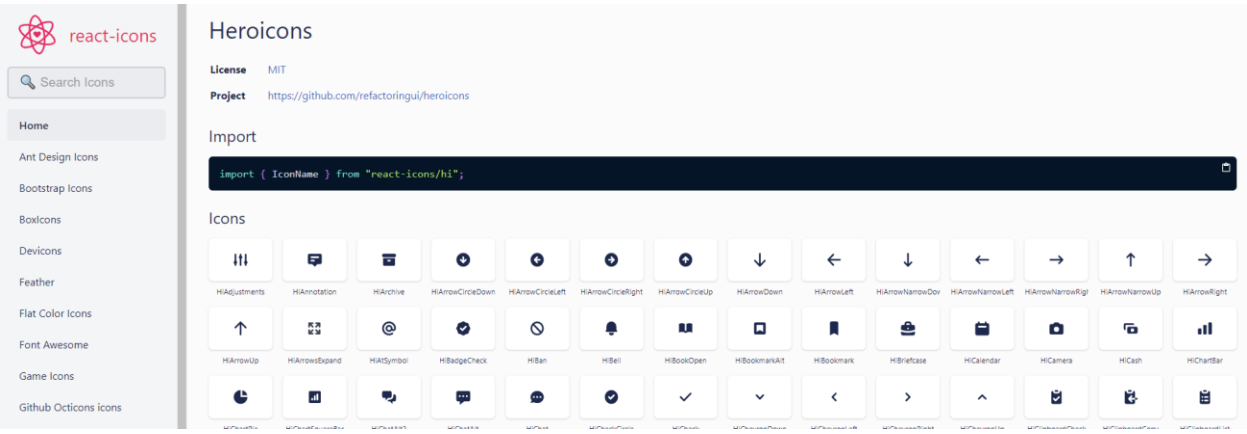


Figura 20 - Captura de pantalla repositori React-Icons

4.3. Estructura

Tal i com s'ha dit prèviament en diverses ocasions, el disseny principal de l'aplicació es basa en una aplicació de pàgina única (SPA).

Tenint això en compte, la idea és dividir la pantalla en tres parts:

- **Menú superior:** En aquesta zona, es mostra el menú bàsic d'accions d'usuari (logout, canviar contrasenya, etc..) i poder canviar el idioma de l'aplicació.

- **Menú lateral:** Aquest és el menú principal de navegació de l'aplicació. En ell, es pot navegar per tots els diferents apartats i pàgines que s'ofereixen. En el cas de la versió per escriptori aquest és fixe, mentre que en el cas de la versió per a dispositiu mòbil aquest es mostra i s'amaga mitjançant un botó.

- **Contingut central:** És la zona on es mostra el contingut corresponent a la pàgina o apartat en que es troba l'usuari. La idea és que el contingut d'aquesta zona s'apili segons l'amplada del container.

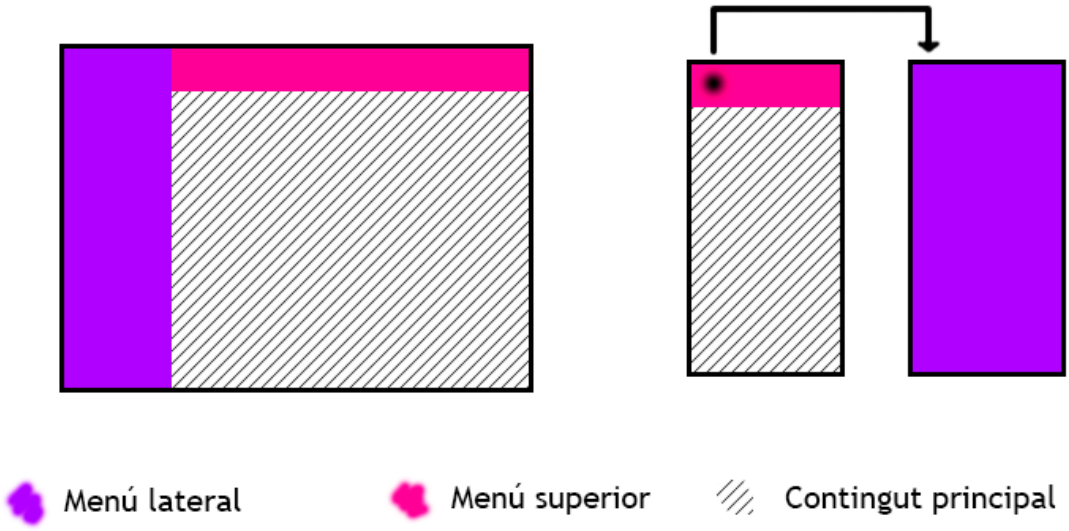


Figura 21 - Estructura simplificada aplicació web

4.4. Prototipatge

Durant les primeres reunions del projecte, s'ha consultat al client quines son les seves idees i expectatives referents a l'apartat visual de l'aplicació. Com a desenvolupador amb més coneixements sobre UX [67] que el client, se l'ha guiat i se li han presentat diferents opcions (patrons, tipus d'aplicació, menús, etc..), fins que el client s'ha acabat fent una idea de com vol l'aplicació.

A continuació, es mostren els prototips que s'han creat a partir de les indicacions obtingudes per part del client en les diverses reunions. Aquest prototips, se li han presentat al client i ens els casos necessaris s'han hagut de fer petits canvis. També destacar, que l'objectiu principal dels prototips és tenir una imatge general de com estan ordenats els diferents elements en l'aplicació, deixant de banda els detalls d'estil final que s'aplicaran en la versió final.

Versió escriptori:

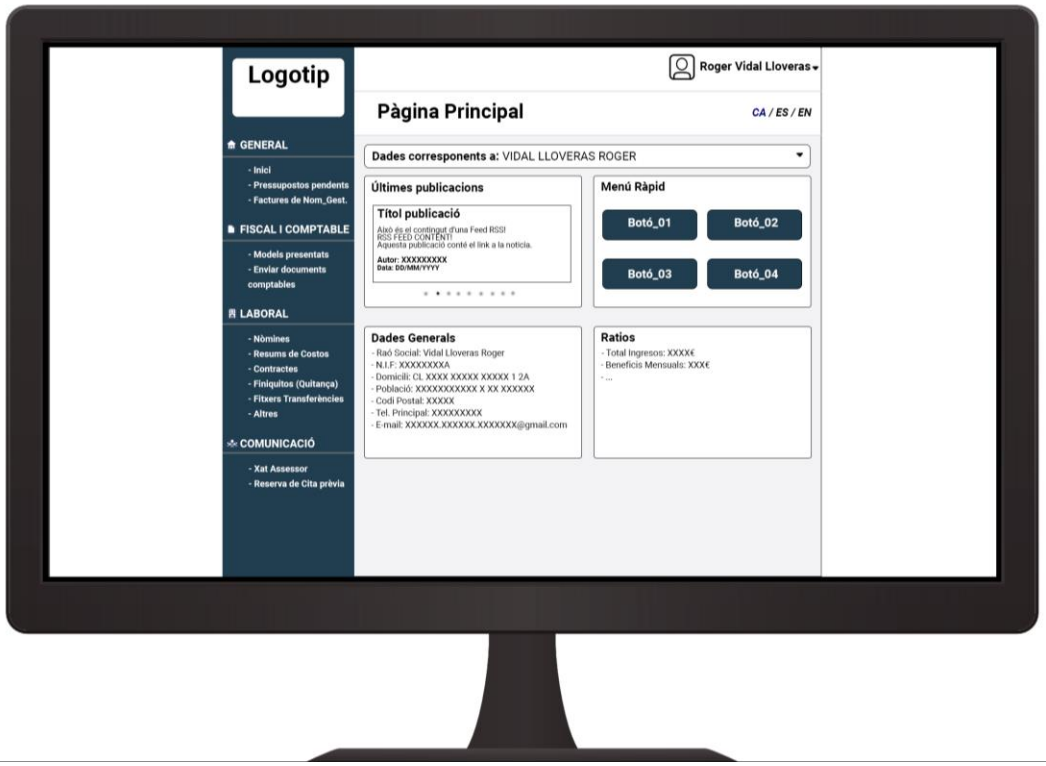


Figura 22 - Prototip aplicació web versió escriptori

Versió dispositiu mòbil:

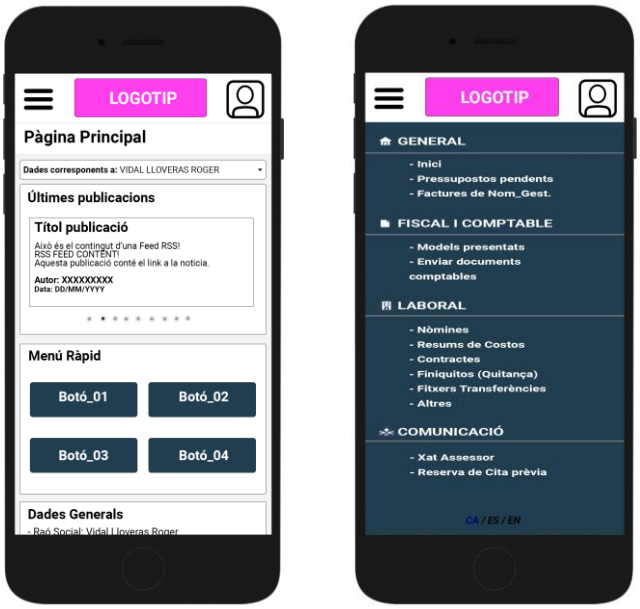


Figura 23 - Prototip aplicació web versió mobile

CAPÍTOL 4: IMPLEMENTACIÓ

Tal i com s'ha pogut veure amb anterioritat, el desenvolupament d'aquest producte principalment consta d'una API REST i una aplicació web desenvolupada amb ReactJS. A continuació, es destaquen alguns dels aspectes més importants que s'han realitzat durant el desenvolupament del producte.

1. Implementació de l'API REST

Per a gestionar la comunicació entre l'aplicació web amb les bases de dades, s'ha desenvolupat una API REST utilitzant el entorn de programació NodeJS i l'entorn de treball ExpressJS. La funció principal d'aquesta API REST és que mitjançant el protocol HTTP, es puguin realitzar les diferents funcions de CRUD [68] (*Create*, *Read*, *Update* i *Delete*).

1.1. Estructura de fitxers

Per tal de mantenir ordenat i net l'entorn de treball, s'ha procurat treballar amb una estructura de fitxers que permeti diferenciar la funcionalitat de les parts que composen l'API REST.

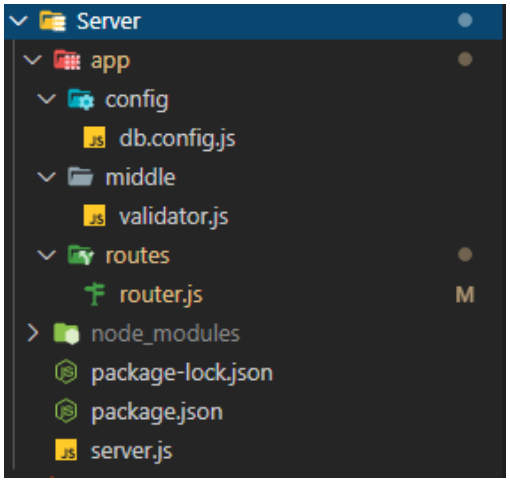


Figura 24 - Estructura VScode Servidor (API)

- **Carpeta "config":** En aquesta carpeta s'hi troba l'arxiu que conté les dades d'accés a la base de dades i també es crea des d'aquí la connexió.

- **Carpeta "middle":** Aquesta carpeta té la funció de contenir els arxius que treballen entre l'API REST i l'aplicació web. En ella s'hi pot trobar per exemple la funció que s'encarrega de determinar si l'usuari que està intentant obtenir dades a través de l'API REST, està correctament autentificat.

- **Carpeta "routes":** Com el seu nom indica aquesta carpeta conté els arxius necessaris per a crear les diferents rutes de l'API REST. Les rutes són la URL a on l'aplicació web envia les seves peticions HTTP (Get o Post) per a realitzar accions o consultar dades.

- **Carpeta "node_modules":** En aquesta carpeta es descarreguen els diferents arxius corresponents als paquets npm que s'estan utilitzant.

1.2. Funcionament API REST

El primer que fa l'API REST és configurar el *middleware* d'Express utilitzant la funció "use" [69]. Aquí es defineixen els cors, *headers*, etc..

```
const app = express();
app.use(cors({ origin: "xxxxxxxxxxxxxxxxxxxxxxxxxx" }));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

Amb els *headers* definits, l'API REST es connecta a la base de dades per a poder passar a definir les rutes. Per a fer-ho, s'utilitza "mysql2", un paquet de node que incorpora diferents funcions referents a la connexió a bases de dades MySql.

```
const mysql = require('mysql2');

const connection = mysql.createPool({
    host: 'xxxxxxxxxxxxxxxxxxxxxxx.com',
    user: 'xxxxxxxxxxxxxxxx',
    database: 'xxxxxxxxxxxxxxxxxxx',
    password: 'xxxxxxxxxxxxxxxxxx'
});
module.exports = connection;
```

Després, s'afegeixen les diferent rutes que s'han definit prèviament, les quals l'aplicació web hi enviarà peticions HTTP per a consultar dades o realitzar accions. Aquestes defineixen el mètode en que el client enviarà les peticions (Get o Post) i indiquen quina serà la ruta necessària per accedir-hi. El cos de la funció, és el conjunt d'accions que es realitzen quan el servidor rep una petició HTTP a la ruta indicada.

```
router.post('/login', (req, res, next) => {
    db.query( .. );
});
```

```
app.use('/api', router);
```

Finalment, l'API REST crida la funció de "*listen*", la qual indica que el servidor està llest per a rebre les peticions per part de l'aplicació web.

```
const PORT = XXXX;
app.listen(PORT, function() {
    console.log("Server is running on port "+PORT);
});
```

1.3. En detall: JSON Web Token

Un dels punts més importants en un projecte com el que s'ha desenvolupat en aquest TFM, és la manera en que els usuaris s'identifiquen per a poder accedir a les dades. És molt important que aquesta identificació sigui del tot segura, ja que resultaria crítica que qualsevol persona podés accedir a dades que no li corresponen.

Per a gestionar aquest procés, hi ha moltes opcions, però per aquest projecte s'ha decidit treballar amb l'estàndard de codi obert JSON Web Token (JWT). Com el seu nom indica, aquest estàndard es basa amb el JavaScript Object Notation (JSON).

L'element principal de JWT són els *tokens*, una cadena de caràcters que ve formada per una capçalera, la carga útil (*payload*) i la signatura, cadascun d'aquests elements separats per un punt.

Figura 25 - Esquema sintaxis JWT

La capçalera (*Header*), consisteix en dos parts: el tipus de *token* (que com s'ha vist és JWT) i el tipus d'algorisme utilitzat per a l'encriptació del *token* (HMAC, SHA256 o RSA).

```
{
     "alg": "HS256",
     "typ": "JWT"
}
```

La carga útil (*payload*) conté els camps d'informació que es volen transferir utilitzant el *token*. En el cas de la utilització per a autoritzacions (com en el cas d'aquest projecte), presenten una estructura similar a aquesta:

```
{
    "id": "AAAAA01",          // Identificació de l'usuari
    "username": "ABCDEFGH",   // Nom d'usuari
    "iat": "16004152101",     // Data en milisegons que s'ha creat el token
    "exp": "1604155701"       // Data en milisegons en que caduca el token
}
```

La signatura es calcula codificant la capçalera i la carga útil en base64url, juntament amb una clau secreta (que només sap el servidor i serveix posteriorment per a poder comprovar la veracitat del *token*).

```
HMACSHA256(base64UrlEncode(capçalera)+"."+base64UrlEncode(carga útil),secret)
```

Finalment, el resultat obtingut d'aquesta signatura és el JWT que s'utilitza per a poder verificar l'autorització del usuaris:

"eyJhbGciOiJIUzIR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IjQ3ODM1OIsInZCI6IjQ3ODM1ODEySyIsImlhd
CI6MTYwNDMSwiZXhwIjoxNjA0MTU1NzAxfQ.d_dsBPPY5-PlDQHVab_Hz0Wz5yBHEJqC_-2AjUZvPWU"

Tenint clar com funciona la creació del *token*, el flux de funcionament d'aquest és el següent:

- L'usuari envia les seves credencials a l'API REST
- Comprova les credencials amb la base de dades i retorna *token* si son correctes
- Quan l'usuari vol fer qualsevol crida que necessita autorització, com a *header* de la petició HTTP adjunta el *token*.
- L'API REST comprova la validesa del *token* i si és correcte, consulta les dades i retorna la resposta.

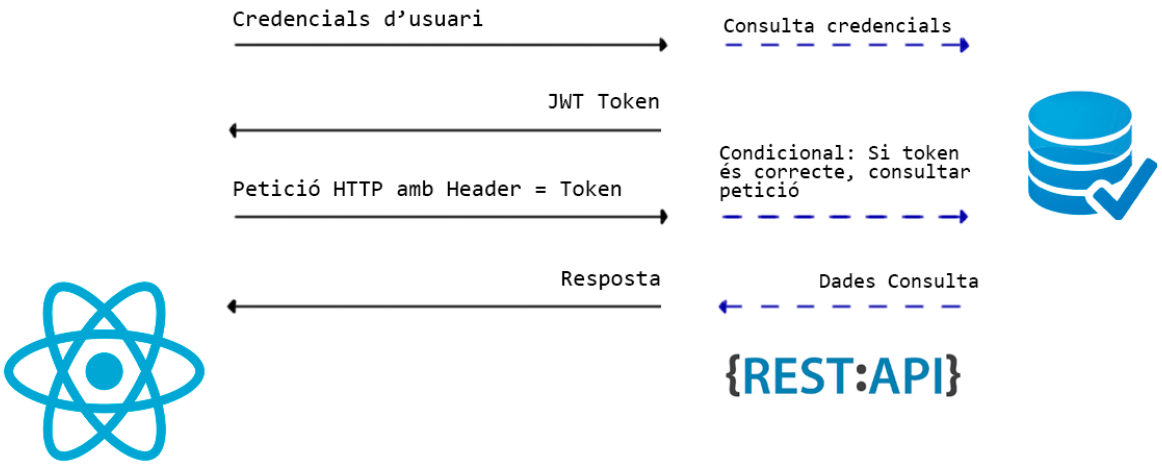


Figura 26 - Exemple JWT Client-Servidor

1.4. En detall: Bcrypt

Quan es treballa amb bases de dades i usuaris, el tractament de les contrasenyes és un punt de gran rellevància al qual se li ha de donar un pes especial.

Com s'ha vingut comentant durant tota la memòria, en aquest projecte, ha estat el client qui ha proporcionat la major part de les taules de les bases de dades. Entre d'elles, la taula on es guarda la informació referent a les credencials d'usuari.

Tot i que aquesta part s'hagués pogut aprofitar tal i com està feta fins ara, s'ha decidit renovar-la. Actualment, les contrasenyes estan guardades amb un encriptat molt bàsic basat en Hexadecimal. Tenint això en compte, s'ha pres la decisió de treballar amb un sistema "d'encriptació" basat en la funció de *hash* "bcrypt" [70], fent ús del paquet de node "bcryptJS".

Bcrypt és una funció de *hashing* per a contrasenyes dissenyada per Niels provos i David Maxieres, basada en el xifratge *Blowfish* [71]. La seva característica principal és que porta un valor anomenat "salt", que és un fragment aleatori que s'utilitza per a generar el *hash* associat a la contrasenya i que es guardarà amb ell dintre la base de dades. Gràcies a aquest sistema, s'evita que dos contrasenyes iguals, generin el mateix *hash*. També, aquesta funció de xifratge destaca per oferir gran protecció contra atacs de força bruta i atacs de "Rainbow Table" [72].

Per aplicar Bcrypt, el primer que es fa és definir quina complexitat té el salt i calcular-lo:

```
var salt = bcrypt.genSaltSync(10); // Calcular un Salt de complexitat 10 ->
2^10 = 1024 iteracions
```

Amb el valor de salt calculat, és el moment de generar el *hash* a amb la contrasenya i el valor salt:

```
var hash = bcrypt.hashSync(contrasenya, salt);
```

La variable *hash*, ens retorna un valor similar a:

```
$2a$10$vI8aWBnW3fID.ZQ4/zo1G.q1lRps.9cGLcZEiGDMVr5yUP1KUOYTa
```

Aquest valor, en realitat són tres camps diferents que venen separats per signes del dolar ($).

- 2a: Aquest camp indica quin tipus d'algoritme d'encriptació s'ha usat (bcrypt).
- 10: Aquí s'indica la complexitat del Salt que s'ha utilitzat per a encriptar.

- vI8...YTa: Aquest és el resultat del encriptat entre el Salt i la contrasenya codificat en Base-64. Els primers 22 caràcters (16 bytes) fan referència al salt, mentre que la resta fan referència a l'autenticació (24 bytes, 31 caràcters).

Aquest *hash* que acabem de veure, és el valor que es guarda a la base de dades en el camp de la contrasenya.

A diferència de encriptacions més antigues, on es basaven en encriptar una paraula clau per a poder desencriptar les contrasenyes, amb bcrypt això no passa. BcryptJS no treballa encriptant, sinó que com s'ha dit és una funció de *hash*, per tant el valor guardat no es pot "desecriptar".

```
var boolea = bcrypt.compare(contrasenya, hash);
```

Al utilitzar la funció compare, aquesta retornarà *true* o *false* segons si la contrasenya és correcte o no.

1.5. En detall: Mòdul comunicació Assessor-Client

Per al client, un dels punts més importants a tenir en compte per a l'aplicació web, és el fet de disposar d'un sistema centralitzat de comunicació. Actualment, els clients de l'assessoria poden contactar amb els seus assessors a través d'una gran quantitat de canals: WhatsApp, correu electrònic, missatgeria tradicional, etc.. L'objectiu, és que a través de l'aplicació web, es consolidi el mòdul de comunicació com element principal per a dur aquestes tasques.

A nivell de requeriments, aquest mòdul ha d'oferir la possibilitat d'enviar missatges a els diferents treballadors de la gestoria per part dels clients i també al revés, dels treballador a els clients. Per altra banda, aquesta no té cap necessitat a ser en temps real, si no que pot funcionar perfectament amb peticions de *pull*.

Tenint això en compte, aprofitant que actualment el client ja disposa d'una base de dades, s'ha decidit realitzar aquest mòdul de comunicació mitjançant bases de dades de MySQL i la mateixa API REST que gestiona les peticions de la resta de l'aplicació web.

La primera preocupació a l'hora de prendre aquesta decisió, ha estat en saber quantes *rows* es poden manejar de manera àgil amb MySQL (ja que hem de tenir en compte que cada missatge serà un nou registre a la taula). Tot i que no hi hagi una resposta única, ja que tot depèn del *hardware* del servidor i del disseny de les *queries*, un servidor bastant senzill pot treballar amb taules de per exemple 500.000.000 de *rows* sense massa complicació [73].

A nivell tècnic de limitació de MySQL, el límit el podem trobar en el màxim índex possible en el cas que tinguem una columna primària única amb increment automàtic, tot i així, tampoc no ens ha de preocupar massa aquest límit [74]:

```
6-byte ROW ID => Màxim ROWS = 2^48 => 281,474,976,710,656
```

Com es pot veure, el número màxim de *rows* a nivell tècnic de MySQL no és un problema real.

Per fer una aproximació a la alta (son uns números molt més alts dels reals) del número de missatges que es poden acumular, fem la següent hipòtesi:

- Hi ha un total de 500 usuaris actius a l'aplicació web
- Cada usuari, envia 20 missatges cada dia
- Tots els missatges es responen cada dia amb igualtat numèrica (20 missatges més)

Amb aquestes dades, tenim que cada dia s'envien 20.000 missatges (500 x 40), el que seria anualment 7.300.000 missatges. Com es pot veure, fins arribar a una xifra que s'acostés a nivells d'exigència a la base de dades, passaran molts anys. Tot i així, també s'ha de tenir en compte que per exemple un cop a l'any, es poden ajuntar certa quantitat de missatges per reduir *rows*, etc..

Tenint clar que no hi ha cap problema tècnic que impedeixi desenvolupar aquest mòdul mitjançant MySQL, s'ha dissenyat l'estructura de dades. Per a fer-ho, s'ha buscar un equilibri entre rendiment, simplicitat i llegibilitat.

Per una banda, hi ha una primera taula, es guarda les dades de la conversa. La columna 'id', fa referència al número de conversa que tenen dos usuaris entre ells. Després, les columnes 'Colaborador' i 'Usuario', formen un índex únic, ja que indiquen de manera única quins son els dos participants en aquesta conversa. Qualsevol comunicació entre aquest dos mateixos usuaris, sempre tindrà el mateix 'id' identificatiu.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	**Id**	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	**Colaborador**	varchar(30)	utf8mb4_spanish_ci		No	*Ninguna*		
3	**Usuario**	varchar(30)	utf8mb4_spanish_ci		No	*Ninguna*		

Figura 27 - Estructura taula conversa (DB)

Per altra banda, hi ha una segona taula on guarden els missatges corresponents a aquestes converses. Hi ha un primer camp 'id', que és la clau primària única, que serveix com element únic identificatiu de cada missatge i per a millor rendiment de les *queries*. Després, el camp de 'Chat', fa referència a la 'Id' de la conversa (emmagatzemat a la taula anterior). El camp de 'Mensaje' com el nom indica és el missatge en si i la 'Direcció' indica quin dels dos usuaris ha enviat el missatge. Finalment, el camp de 'Lectura' és per saber si els usuaris han llegit el missatge i el 'Nombre', és per el cas de les comunicacions amb empreses, on diferents persones poden parlar des de la conta de l'empresa.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	**Id**	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	**Chat**	int(11)			No	*Ninguna*		
3	**timestamp**	timestamp			No	current_timestamp()		
4	**Mensaje**	varchar(2500)	utf8mb4_spanish_ci		No	*Ninguna*		
5	**Direccion**	tinyint(1)			No	*Ninguna*		
6	**Lectura**	tinyint(1)			No	0		
7	**Nombre**	varchar(160)	utf8mb4_spanish_ci		No	*Ninguna*		

Figura 28 - Estructura taula Missatge (DB)

Gràcies a aquesta distribució de les dades, d'una manera molt ràpida, MySQL pot obtenir tots els missatges referents a una conversa entre dos punts.

A nivell d'usuari, el que fa l'aplicació web és:

- Per una banda, cada "x" minuts, comprova si hi ha missatges nous en alguna de les converses de l'usuari (també ho comprova al encendre l'aplicació). Al no ser un "xat en viu", l'interval de temps entre peticions de *pull*, no ha de ser excessivament curt per a funcionar segons l'objectiu del mòdul.

- Per altra banda, quan l'usuari entra en una conversa, fa una petició de *pull* dels últims 50 missatges (en cas de voler veure missatges més antics, l'usuari pot anar al principi del "xat" i carregar tot el historial sencer).

2. Implementació de l'aplicació web

Com s'ha vist en l'apartat anterior, el Back-End està cobert per una API REST desenvolupada en NodeJS. Per la part del Front-End, s'ha creat una *single-page application*, desenvolupada amb ReactJS. La funció principal d'aquesta, és pintar per pantalla totes aquelles dades obtingudes a partir de crides *http* fetes sobre l'API REST.

2.1. Estructura de fitxers

Tot i comptar amb una quantitat important de fitxers, s'ha intentat dintre de lo possible mantenir el màxim ordenada tota l'estructura de fitxers del projecte.

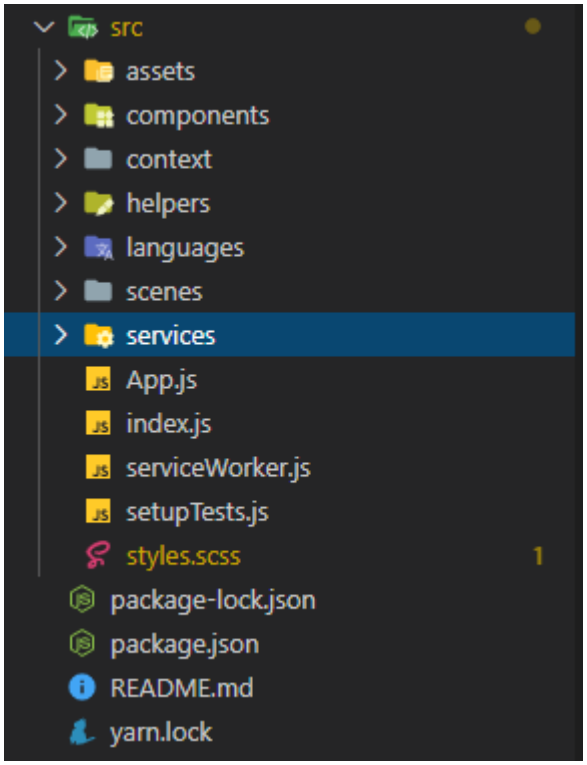


Figura 29 - Estructura VScode Aplicació (ReactJS)

- **Carpeta "assets":** La funció d'aquesta carpeta és emmagatzemar tots aquells recursos visuals necessaris per a desenvolupar el Front-End del projecte (imatges, *banners*, etc..)

- **Carpeta "components":** En la carpeta de "components", s'hi troben tots aquells components de React que no son una escena i són re-aprofitables. Per exemple, hi trobem el component de canviar d'usuari (userchanger.js), que és una barra de tipus input desplegable, on l'usuari pot canviar entre els diferents comptes que té.

- **Carpeta "context":** Aquesta carpeta conté els arxius per a inicialitzar els diferents React Contexts que s'utilitzen en el projecte. A mode de resum [75], aquests són els equivalents a variables globals, però en format *hook* de React. Per aquest projecte, hi tenim un que fa una funció general entre les diverses escenes (s'hi guarden variables d'estat necessàries entre diferents components) i un altre que s'encarrega de gestionar els diferents idiomes.

- **Carpeta "helpers":** Com el seu nom indica, aquesta carpeta conté arxius que subministren funcions que serveixen com "ajuda" per a fer diferents tasques. Entre aquestes funcions, hi podem trobar una de tant simple com un generador de número aleatoris o una altra de més complexa, que serveix per filtrar l'accés dels usuaris no registrats.

- **Carpeta "languages":** En aquesta carpeta s'hi troben els diferents arxius en format JSON, corresponents a els diferents idiomes de l'aplicació.

- **Carpeta "scenes":** Aquesta és la carpeta on es troben els components de React què són una escena. És a dir, ve a ser l'equivalent de la clàssica pàgina *html* referent a un apartat concret de l'aplicació (per exemple, el "home").

- **Carpeta "services":** La carpeta de "services", conté els diferents arxius necessaris per a comunicar-se amb la API REST del projecte. Bàsicament, un arxiu que conté les funcions necessàries per temes de gestió d'usuari (login, canviar contrasenya, etc..), un altre arxiu que conté les funcions corresponents a totes les crides de l'aplicació i un tercer arxiu, que conté configuracions de *headers*.

- **Arrel del projecte:** A l'arrel del projecte, s'hi troben "App.js" i "Index.js", que venen a ser com el index.html del projecte. En ells, s'hi inicien diferents processos i també contenen els enrutadors a tota la resta d'escenes. Per altra banda, també hi trobem les configuracions de *npm,* les de *yarn* i el arxiu *readme* del projecte.

2.2. Funcionament general de l'aplicació

Tot el projecte de React, es centra sobre el index.js. En ell, s'hi declara l'enrutador [76] (component per a gestionar les diferents rutes de l'aplicació web) i s'hi indica que s'ha de renderitza per pantalla l'App.js (amb tots els seus descendents).

```
ReactDOM.render(
    <BrowserRouter>
        <App />
    </BrowserRouter>,
    document.getElementById("root")
);
```

A partir d'aquest punt, entrem dintre el component App, que és el principal de l'aplicació. Per una banda, en aquest s'hi inicialitzen certs elements necessaris que es necessiten posteriorment en les diferents escenes. Per altra banda, s'hi troba l'esquelet principal de l'aplicació. Aquest esquelet defineix quines escenes són públiques, quines són privades i també, quines són les seves rutes. Finalment, també s'hi indica quin tipus de dada "global" (contexts de React) pot accedir cadascuna d'aquestes rutes.

```
function App () {
    ..inicialitzacions varies..
    return(
        <Context>
            <Switch>
                <Route pública .. />
                <PrivateRoute privada .. />
                ...
            </Switch>
        </Context>
    )
}
```

En aquest punt, l'aplicació ja ha inicialitzat les dades que necessita prèvies a començar amb el renderitzat i també té preparades les diferents rutes referents a les diferents escenes.

La primera escena que renderitza per defecte, és la ubicada a la ruta arrel "/" (que fa referència a l'escena de *login*). Aquesta és la única ruta pública del projecte, és a dir, és la única escena que els usuaris sense estar acreditats poden visualitzar.

Un cop l'usuari ha introduït les seves credencials, aquestes s'envien a l'API REST per a determinar si son correctes o no. En el cas de ser correctes, el servidor respon a la petició http amb un *token* d'autorització (tal i com s'ha vist en l'apartat referent a el Back-End, un *token* de JWT). Aquest *token*, posteriorment serà utilitzat en totes les peticions *http* que l'usuari faci per tal d'acreditar la seva identitat.

Un cop l'usuari ja està acreditat, l'aplicació ja li permet navegar a les diferents rutes protegides sota acreditació. En aquestes, l'usuari fa peticions a l'API REST per obtenir la informació que necessita cadascuna de les escenes.

Aquestes, són components funcionals de React, els quals tenen una estructura com aquesta:

```
function Exemple() {
    // Inicialització de Hooks variats i funcions referents a la renderització de
l'escena
    const [dataContext, setContext] = useContext(DataContext);

    function createElements () {
        //...
        return element;
    }
    return(
        <div className="Example">
            {createElements()}
        </div>
    )
}
export default Exemple;
```

Com es pot veure en l'exemple exposat, les diferents escenes tenen una estructura molt definida. Per crear escenes més complexes, s'utilitzen components que s'han creat a mode reutilitzable, i s'importen i es renderitzen en les diferents escenes per a muntar l'aplicació web.

En vistes molt generals, aquest és el funcionament bàsic de l'aplicació web.

- Es declara l'enrutador i es defineix que l'element App és la base del projecte
- S'inicialitzen dades prèvies i s'indiquen quins components formen part del projecte
- Quan l'usuari accedeix a una ruta, aquesta renderitza per pantalla diferents components que formen l'escena.
- Totes les dades es demanen a una API REST mitjançant peticions *http*

2.3. En detall: Aplicació d'estils

Tal i com s'ha indicat prèviament en aquesta memòria, per a la realització d'aquest projecte, s'ha pres la decisió de no utilitzat cap llibreria d'estils (com podrien ser Bootstrap o Tailwind).

Tenint això en compte, per a l'estilització del projecte, hi ha dos principals blocs a destacar: SASS i Flexbox. Aquestes dos eines, tenen la finalitat de facilitar la feina a l'hora de crear els estils de l'aplicació i també a millorar la qualitat final del producte.

Per a crear la fulla d'estils del projecte, s'ha utilitzat SASS, un metallenguatge de nivell superior caracteritzat per la seva estructura neta i organitzada.

Dintre les seves principals característiques, s'hi troba el fet de poder "nestejar" elements:

```
.example {
    width: 100%;
    .sendbutton {
        color: white;
        &:hover {
            cursor: pointer;
        }
    }
}
```

I també la capacitat de definir variables per a poder-les utilitzar posteriorment en els estils aplicats:

```
$customcolor: rgb(47,64,80);
.component {
    background-color: $customcolor;
}
```

Finalment, SASS també destaca per a la possibilitat de crear funcions dintre el mateix full d'estils:

```
@mixin lamevafuncio($paràmetre) {
    background-color: red;
    border: 1px solid black;
}
```

Per altra banda, a nivell d'estilització del projecte, cal destacar FlexBox, un model de *layout* dissenyat per a distribuir els diferents elements d'una escena per pantalla. Gràcies a aquest, es poden definir les posicions dels elements per pantalla sense necessitat d'haver de recórrer a posicions absolutes i també, ofereix una facilitat a l'hora de generar interfícies responsives.

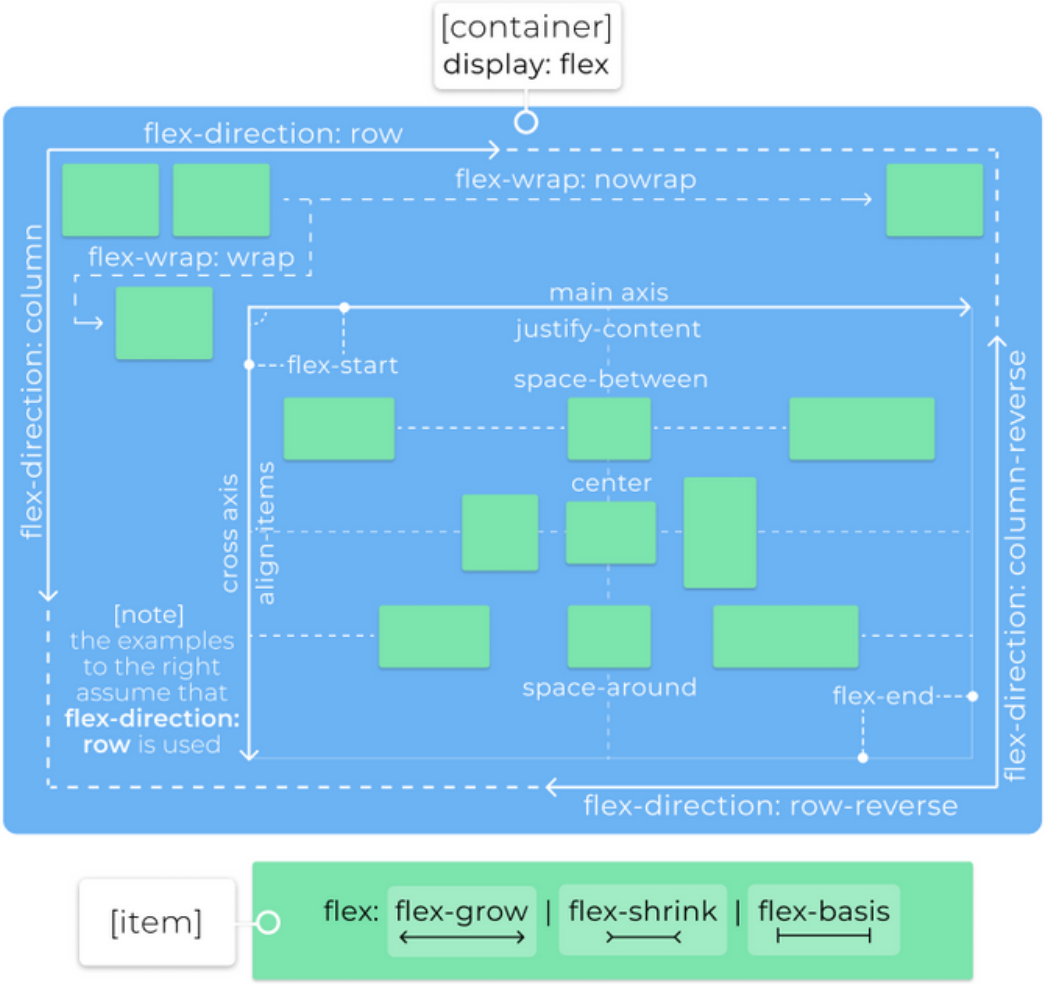


Figura 30 - Cheat Sheet de FlexBox. FONT: 30secondsofcode

Gràcies a aquestes dos "eines", la creació d'una interfície totalment responsiva i visualment atractiva, resulta una feina molt més senzilla que treballant amb CSS bàsic.

SASS permet escriure la fulla d'estils de manera ordenada i seguint una estructura de codi semblant a "Javascript". Si combinem això amb la facilitat de posicionament i re-dimensió que ofereix FlexBox, aquests formen una combinació perfecta a l'hora d'estilitzar qualsevol aplicació web.

2.3. En detall: Tractament token autorització

Anteriorment, en un dels apartats de la implementació de l'API REST, s'ha explicat que per l'autenticació s'utilitza un JWT i com funciona.

Sabent això, un cop l'API REST envia el *token* a l'usuari, aquest l'ha de guardar en algun lloc. Cal recordar, que de manera pública, aquest *token* no conté cap informació que es pugui considerar delicada (conté el nom d'usuari, la data de creació i la data d'expiració).

Tot i que realment poden ser moltes les maneres de com guardar el *token* en el client, les dos més comunes passen per utilitzar el *localStorage* [77] o per utilitzar una *cookie* [78].

Realment, a nivell de seguretat no existeix una fórmula perfecte, doncs si un usuari és l'objectiu directe d'algú amb coneixement de *hacking*, es podria dir que no hi ha defensa possible. La idea, és desenvolupar un sistema, que protegeix davant d'atacs no personals.

Tenint tot això present, tant *localStorage* com les *cookies*, ambdues opcions tenen punts forts i altres de fluixos.

localStorage

- Qualsevol usuari amb accés directe a l'ordinador, el pot consultar de manera senzilla
- S'hi pot accedir a través de Javascript (per exemple, amb un *framework* com React)
- Al ser accessible per JS, el seu contingut es pot enviar com a *header* en una petició *http* del tipus autorització (*Auth*)
- És vulnerable a atacs XSS [79]

Cookies

- Qualsevol usuari amb accés directe a l'ordinador, les pot consultar de manera senzilla
- No son accessibles a través de Javascript si s'utilitza *httpOnly* i *Secure* (ni amb *frameworks* que l'utilitzin)
- S'envien de manera automàtica a cada petició *http* feta a l'API REST
- Son vulnerables a atacs CSRF [80]

El primer que cal és entendre exactament que son aquests atacs a que s'exposen ambdues opcions de manera molt resumida:

- **XSS:** És quan l'atacant pot executar un codi de Javascript. La manera més comuna d'introduir el codi, és a través de dependències de tercers (que el mateix desenvolupador ha inclòs per ajudar-se en el codi)

- **CSRF:** És quan l'atacant s'envia una petició http (que porta la cookie inclosa) en el seu mateix servidor. Per a fer-ho, s'acostumen a usar pàgines *phishing* enviades a través d'un correu electrònic o similar (creen una copia de la pàgina web, en el nostre cas seria l'aplicació, però amb les peticions http redirigides).

Entenent els dos tipus d'atacs més possibles, cal destacar que en certa manera els dos es poden protegir (dintre del possible):

- **XSS:** Tal i com s'ha fet en el desenvolupament d'aquest projecte, és interessant usar el menor nombre possible de dependències externes, i les que s'usen, que siguin sempre d'un origen de confiança. També, cal destacar que els frameworks moderns com ReactJS ofereixen protecció contra aquest tipus d'atac (per defecte ReactJS utilitza *escape* [81] en tots els *strings*).

- **CSRF:** Es poden utilitzar anti-CSRF tokens [82] i també usar l'opció de *sameSite* [] a la *cookie*.

Veient totes les possibilitats i problemes que presenten cadascuna de les opcions, si es té en compte que en aquest projecte el contingut del token no suposa cap risc a nivell de confidencialitat, s'ha optat per usar el localStorage. El principal motiu, és el fet de poder accedir a la informació del *token* des de l'aplicació web i la facilitat que ofereix *express* a la validació per *header* respecte a *cookies*.

Finalment, aclarir que el *token* proporcionat per l'API REST a l'usuari, té una validesa limitada. Per tant, si un atacant aconseguís el *token* (ja sigui amb un atac directe o amb accés físic/control de l'ordinador), aquest tindria un temps limitat d'accés a la plataforma.

2.4. En detall: Components funcionals

Un dels objectius d'aquest Treballs de Final de Màster, és el fet d'aprendre noves tecnologies i aprofitar-ho com una possible entrada al mercat laboral. És per això, que al igual que s'ha cercat un *framework* que està a l'ordre del dia, dintre d'aquest s'ha procurat aprofitar les seves últimes actualitzacions.

Des dels seus inicis, ReactJS està enfocat a la Programació Orientada a Objectes [83] (OOP). En aquesta, com el seu nom indica, es basa en que els components de ReactJS són un objecte i tenen estats (this.state).

```
class Example extends React.Component {
    constructor(props) {
      super(props);
      this.state = {key: _key};
    }

    render() {
        return (
            <div></div>
        );
    }
}
```

Aquesta manera de treballar els components, ha estat la única fins fa poc. Recentment, ReactJS ha apostat per els components funcionals. Aquest, en comptes de basar-se en Classes amb estats, com diu el seu nom es basen en funcions. En aquests, en comptes de que els components tinguin estats, els valors venen donats com arguments de la funció.

```
function Example (prop) {
    let key = prop.key;
    return(
        <div></div>
    );
}
export default Example;
```

Tenint en compte que en el projecte d'aquest TFM, en la seva majoria el Front-End tan sols pinta de manera ordenada les dades proporcionades per l'API REST, la programació funcional és la millor opció per aquest projecte.

2.5. En detall: React Hooks

Seguint la mateixa dinàmica que en el punt anterior, no només amb la declaració de components s'ha tingut en compte les últimes novetats de ReactJS.

Com s'ha vist en el punt anterior, un dels aspectes més característics dels nous components funcionals, és el fet que no tenen estat. No obstant, tot i que en la majoria de casos la lògica ve totalment donada per l'API, en alguns casos concrets es necessita aplicar lògica.

Per posar un exemple ben senzill, en els diferents mòduls que proporcionen descàrregues, hi ha filtres i aquests necessiten guardar el seu estat en algun lloc. Aleshores, ve es podria guardar-ho en una variable, però a l'igual que en la programació orientada a objectes no tot es guarda en variables, aquí també ho hem d'evitar. Aleshores, per a proporcionar una solució a la falta d'estats en els components funcionals, s'han agregat els *hooks* [84].

Tot i que els *hooks* compten amb una gran complexitat i són molts profunds, la manera més senzilla d'entendre-ho, és que venen a simular ser un estat.

Per exemple, per crear un filtre senzill amb un component de classe, seria un codi semblant a aquest:

```
class MyClass extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            filtre1: false,
            filtre2: false,
            data1: "dd/mm/yyyy",
            data2: "dd/mm/yyyy"
        };
    }
    render() {
        return (
            <div></div>
        );
    }
}
```

Aleshores, la mateixa solució aplicada a un component funcional (tal i com s'ha fet en el projecte), seria un codi com aquest:

```
function MyFunctionComponent (prop) {
    const [filtre1, setFiltre1] = useState(false);
    const [filtre2, setFiltre2] = useState(false);
    const [date1, setDate1] = useState("dd/mm/yyyy");
    const [date2, setDate2] = useState("dd/mm/yyyy");

    return(
        <div></div>
    );
}
export default MyFunctionComponent;
```

Com es pot veure, el codi tampoc és tan diferent, però en aquest segon cas s'utilitzen recursos més moderns de cara al mercat i també té una sintaxis més semblant al Javascript (per aprofitar coneixements previs autor del projecte).

Finalment, un dels principal avantatges a l'hora d'utilitzar *hooks* en comptes de per exemple, guardar els mateixos filtres en variables, és el fet que al cridar un *hook* es força el re-renderitzat del component.

Si fos el cas de tenir els filtres en una variable i aquests es canvien, el resultat pintat per pantalla (html), no es canviaria fins que de manera automàtica (cada certs temps), ReactJS fes un re-renderitzat.

```
    //...
    let filtre1 = false;
    //...
    filtre1 = !filtre1;
```

En canvi, en el cas que aquest canvi es faci mitjançant un *hook* (simulant un canvi d'estat), automàticament es crida la "funció" de re-renderitzat i el canvi es veurà reflectit de manera automàtica per pantalla (actualització del html).

```
    //...
    const [filtre1, setFiltre1] = useState(false);
    //...
    setFiltre1(!filtre1);
```

3. Implementació Eina missatgeria per a Gestors

Com s'ha vist en els punts anteriors, un dels mòduls principals de l'aplicació web és un sistema de missatgeria no instantani. Aquest, s'ha de convertir en la principal eina de comunicació entre els diferents gestors i els clients.

Al ser una eina de treball amb molta importància amb la rutina diària, s'ha decidit desenvolupar una petita eina externa a l'aplicació, per tal que els diferents treballadors puguin gestionar la missatgeria amb els clients de manera més eficient. Aquesta eina, s'ha desenvolupat utilitzant la biblioteca de jQuery.

3.1. Estructura de fitxers

L'estructura de fitxers d'aquesta eina, és molt senzilla, doncs realment l'aplicació només consta d'un sistema per a autenticar-se i la pàgina de l'eina de missatgeria.

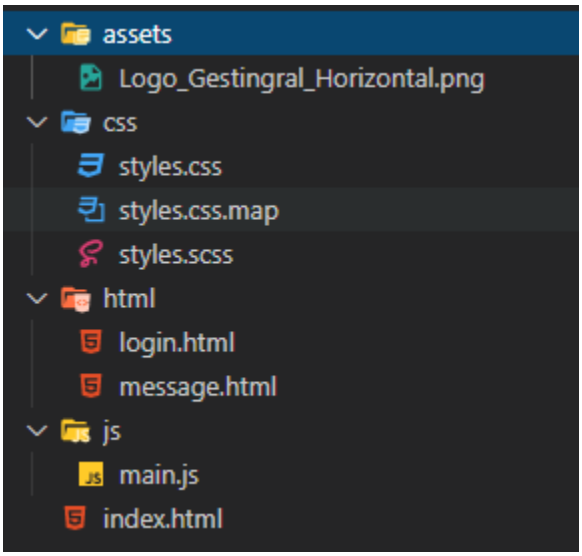


Figura 31 - Estructura VScode Eina Missatgeria (jQuery)

- **Carpeta "assets":** Aquesta carpeta conté les imatges necessàries per a composar l'interfaç de l'eina. Com es pot veure, actualment la carpeta tan sols conté una imatge amb el logotip de la gestoria.

- **Carpeta "css":** La carpeta "css", és l'encarregada de contenir els arxius necessaris per a crear les fulles d'estil de l'eina. Com es pot veure, s'hi troba el arxiu en sass (juntament amb el seu .map) i també l'arxiu compilat en css.

- **Carpeta "html":** Aquí hi ha els elements html corresponents a les dos pàgines principals de l'eina de missatgeria. Aquests es carreguen de forma dinàmica en jQuery, seguint l'estructura d'una SPA.

- **Carpeta "js":** Aquesta carpeta conté el codi Javascript del projecte (jQuery). Aquí hi ha el main, què és l'arxiu encarregat de tota la lògica de l'eina.

3.2. Funcionament de l'eina de missatgeria

Tanmateix com en el cas de l'aplicació web, per a l'eina de missatgeria per al gestors, s'ha optat per a utilitzar una estructura de SPA.

El primer que es fa, des de el *index.html*, és carregar les diferents dependències necessàries per a l'eina (jQuery, google fonts, jQueryUI i els nostres arxius).

Amb aquestes carregades, el *main.js* s'encarrega de carregar de manera dinàmica la pàgina del *login.html* dintre del *div* principal de l'índex.

```
function loadLogin () {
   $("#mainzone").load("html/login.html", function(responseTxt, statusTxt, xhr) {
  if(statusTxt == "error") alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
}
```

Quan la pàgina ja està carregada del tot, la lògica comença a funcionar (l'usuari es pot autenticar com en qualsevol altra aplicació).

Per a fer les crides a servidor (API REST), s'ha utilitzat AJAX de manera asíncrona.

```
$.ajax(settings).done(function (response) {
     // Gestionar la resposta
 }).fail(function(xhr, status, error) {
     // Gestionar l'error
 });
```

Un cop l'usuari s'ha autenticat, la lògica del "xat" és molt similar a la de l'aplicació web, però amb la diferència que en aquesta eina l'usuari pot iniciar conversa en qualsevol usuari.

Els canvis més significatius, són a nivell visual, ja que l'eina està totalment dissenyada per a l'ús del sistema de missatgeria. També, conté un tema fosc i un altre de clar, per a que cada treballador pugui decidir com prefereix treballar.

CAPÍTOL 5: VALIDACIÓ

Aquest capítol, té com objectiu deixar constància del bon funcionament de tots els elements que s'ha descrit durant aquesta memòria. Per a fer-ho, a continuació es faran breus explicacions de cadascun dels passos que pot fer un usuari a l'aplicació i aquests aniran acompanyats de captures de pantalla a mode de validació visual.

1. Procés d'autenticació

Per a accedir a la plataforma web, cada client té un usuari i contrasenya que han estat facilitats per part de la gestoria.

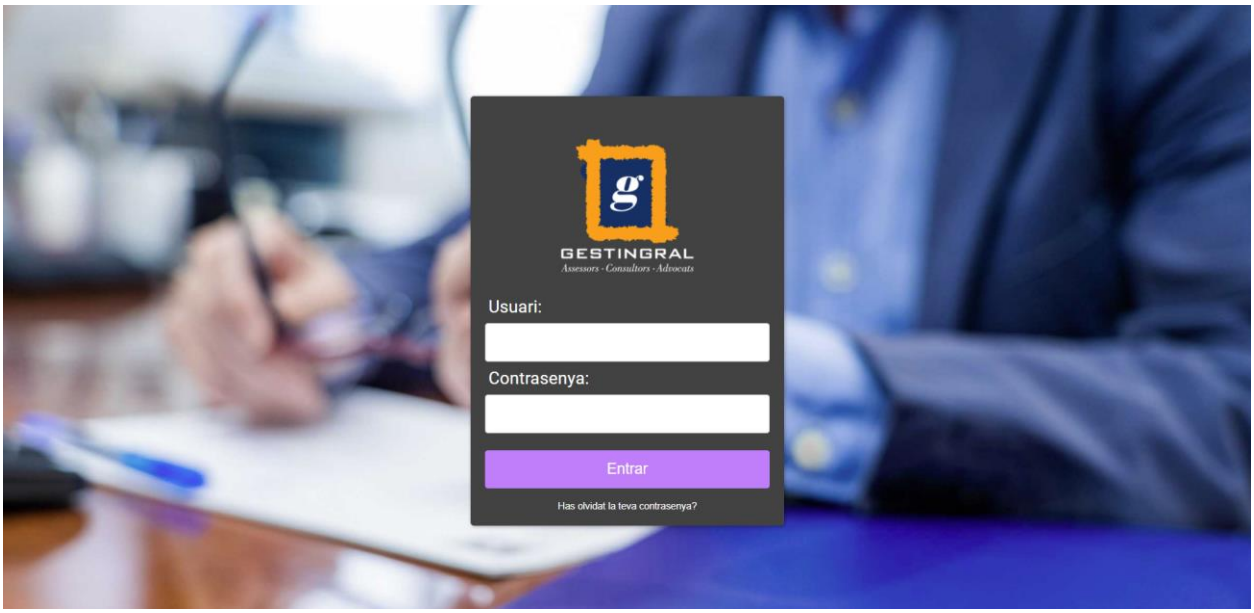


Figura 32 - Captura de pantalla escena Login

Per tal de mantenir la confidencialitat del client, el primer cop que l'usuari entra a l'aplicació, aquesta li demana canviar la contrasenya. Un cop l'usuari canvia la contrasenya, aquesta queda encriptada a la base de dades i la gestoria no la pot saber (tal i com s'ha explicat, s'utilitza un sistema de xifrat amb *bcrypt*).

2020-11-11 11:30:17	00000000A	User001	$2a$10$jVnWR9VcCsIW6YOJcI.YJewRfm6/2ThD6id8C6rH6RF...

Figura 33 - Exemple xifratge de contrasenya DB

Ara que l'usuari té la seva contrasenya, ja pot autenticar-se de manera normal a l'aplicació. Dintre de l'aplicació, en cas que ho necessiti, l'usuari pot canviar la contrasenya i el correu electrònic associat al compte.

2. Procés de recuperació de contrasenya

Actualment, amb la plataforma web que disposa actualment la gestoria, no existeix la possibilitat per part de l'usuari de recuperar la seva contrasenya en cas de pèrdua. Si un usuari perd la contrasenya, el client s'ha de posar en contacte amb la gestoria i aquests han d'editar la base de dades per a crear una nova contrasenya per el client.

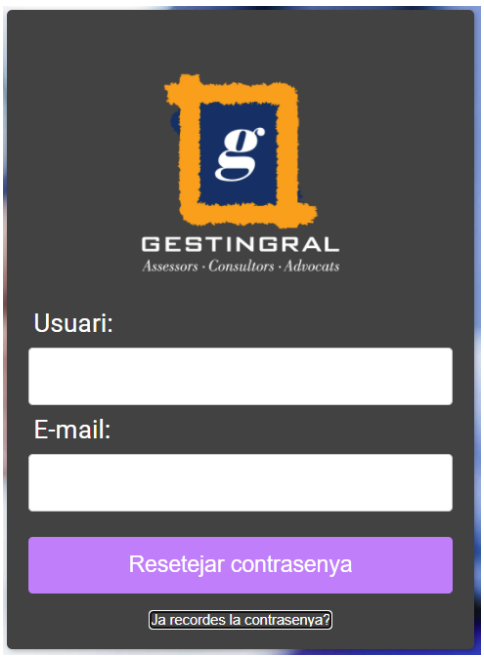
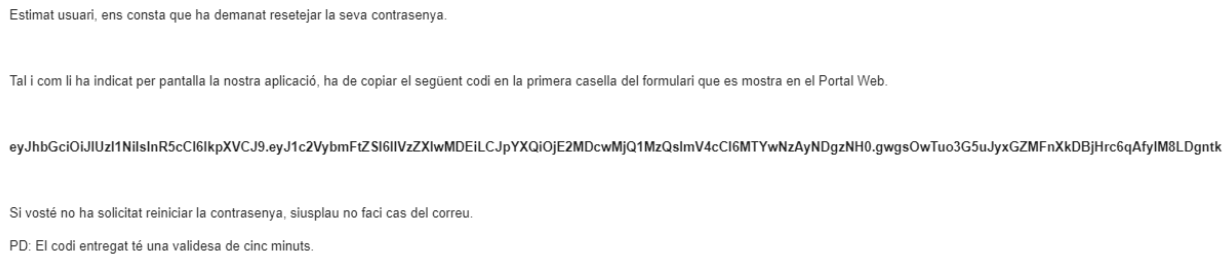


Figura 34 - Captura pantalla recuperar contrasenya

És per aquest motiu, que s'ha incorporat l'opció de recuperar la contrasenya en la plataforma web. Per a fer-ho, el client ha d'indicar el seu correu electrònic i el nom d'usuari. Amb això, l'usuari rebrà en el seu correu electrònic un missatge amb un codi únic (JWT) amb expiració de cinc minuts que li permet realitzar el canvi.


Estimat usuari, ens consta que ha demanat resetejar la seva contrasenya.

Tal i com li ha indicat per pantalla la nostra aplicació, ha de copiar el següent codi en la primera casella del formulari que es mostra en el Portal Web.

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IlVzZXIwMDEiLCJpYXQiOjE2MDcwMjQ1MzQsImV4cCI6MTYwNzAyNDgzNH0.gwgsOwTuo3G5uJyxGZMFnXkDBjHrc6qAfyIM8LDgntk

Si vosté no ha solicitat reiniciar la contrasenya, siusplau no faci cas del correu.

PD: El codi entregat té una validesa de cinc minuts.

Figura 35 - Captura pantalla correu electronic recuperar contrasenya

3. Mòdul General i gestió d'usuaris

Un cop els usuaris ja s'han identificat, accedeixen a la pàgina principal. La funció d'aquesta, és mostrar les dades personals de l'usuari, ratis que li poden interessar veure a primer cop d'ull, les últimes notícies publicades al *blog* de la gestoria i un menú amb quatre tecles d'accés ràpid (per exemple, es podria posar accés ràpid al servei de missatgeria, accés ràpid a reservar cita, etc..)

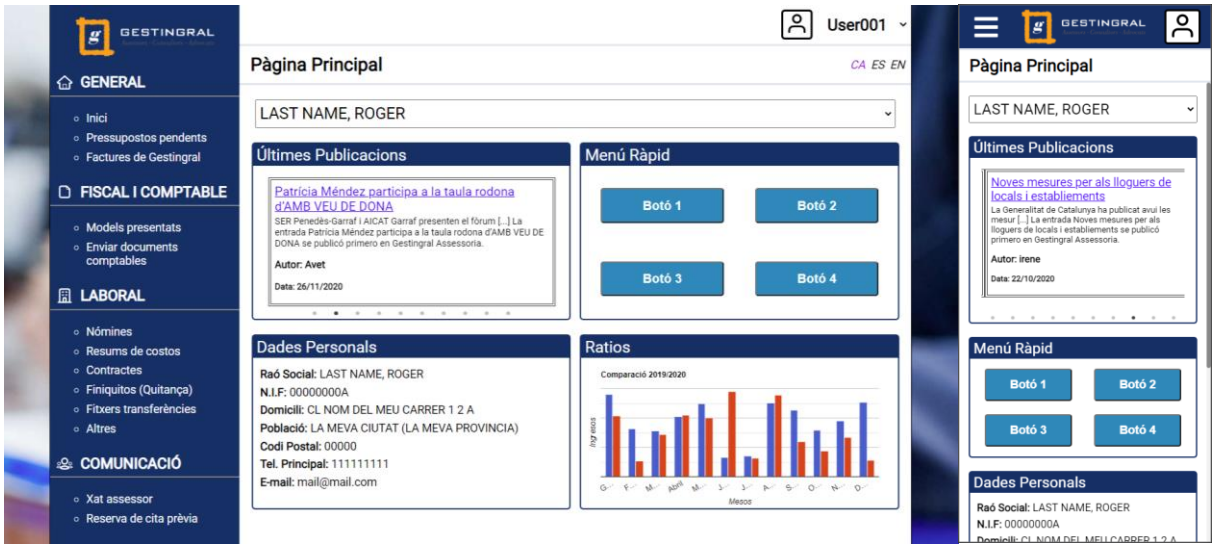


Figura 36 - Captura de pantalla escena Home (Desktop/mobile)

Com s'ha vist al llarg de la memòria, la plataforma web és totalment responsiva. En la versió *desktop* afegeix els marges de fons fins a una mesura predeterminada. En el cas de la versió *mobile*, ofereix una distribució de *UI* totalment adaptada.

A nivell de *Back-End*, el que fa l'aplicació web és donada una *Array* amb els diferents usuaris que té el client (que van associats al compte de la plataforma), retorna una estructura de dades que conté les dades personals bàsiques (i algunes dades extra que són necessaries des de un bon començament per el bon funcionament de l'aplicació).

Headers Preview Response Initiator Timing

```
▼{userdata: [,…],…}
  ▼services: [{Cod_ cliente: "00000000A", Cod_ servicio: "NOM", Fecha baja: "1753-01-01T00:14:44.000Z"},…]
    ▶0: {Cod_ cliente: "00000000A", Cod_ servicio: "NOM", Fecha baja: "1753-01-01T00:14:44.000Z"}
    ▶1: {Cod_ cliente: "00000000A", Cod_ servicio: "PWS", Fecha baja: "1753-01-01T00:14:44.000Z"}
    ▶2: {Cod_ cliente: "00000000A", Cod_ servicio: "REN", Fecha baja: "1753-01-01T02:38:32.000Z"}
    ▶3: {Cod_ cliente: "00000000B", Cod_ servicio: "REN", Fecha baja: "1753-01-01T00:14:44.000Z"}
  ▼userdata: [,…]
    ▶0: {timestamp: "2020-11-11T12:58:06.000Z", Cod_cliente: "00000000A", Denominacion: "LAST NAME, ROGER",…}
    ▶1: {timestamp: "2020-11-11T12:59:14.000Z", Cod_cliente: "00000000B", Denominacion: "MY COMPANY",…}
```


Figura 37 - Resposta http a petició dades client

També, com a element comú en la majoria de pantalles de la plataforma, el client pot navegar entre els diversos usuaris que tingui. Això està pensat, per a clients que per exemple tinguin tant la seva empresa com a clients i també ells a títol personal.

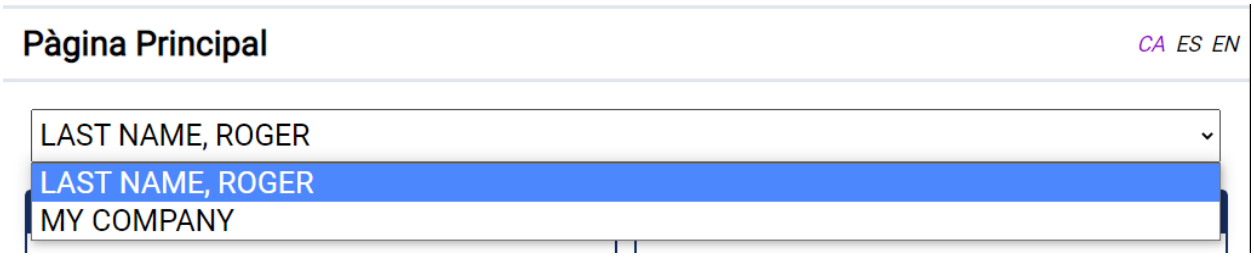


Figura 38 - Desplegable usuaris disponibles

Finalment, destacar que tota l'aplicació està traduïda a Català, Castellà i Anglès.

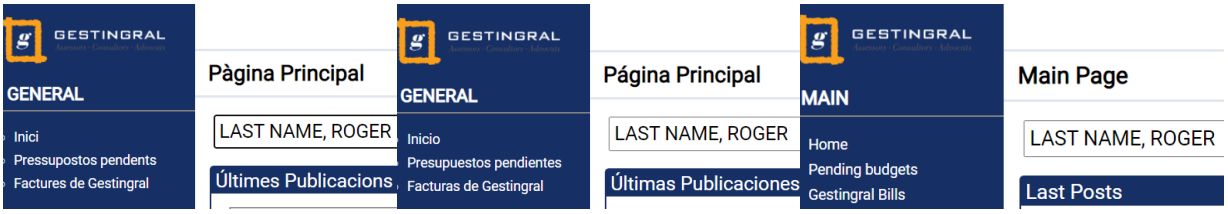


Figura 39 - Comparativa idiomes aplicació web

4. Mòduls de Descarregues de fitxers

Dintre de la l'aplicació web, hi ha diversos apartats que compleixen la funció de proveir fitxers per a l'usuari (Factures Gestingral, Models Presentats, Contractes, etc..). A nivell d'estructura, totes elles són iguals, però a part de canviar els fitxers a mostrar, canvien els filtres segons la categoria i també alguna de les columnes (per exemple, en l'apartat de contractes, hi ha una columna indicant a qui fa referència el contracte).

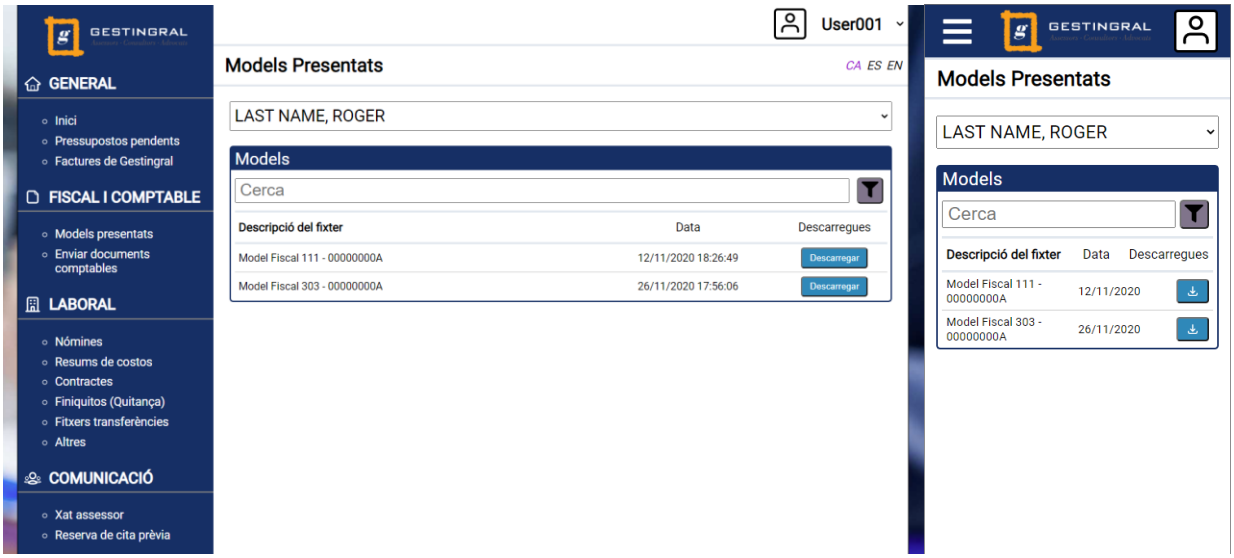


Figura 40 - Captura de pantalla escena Models presentats (Desktop/mobile)

Per a filtrar, es pot utilitzar la barra de cerca (introducció de text) o utilitzar algun del filtres. Dintre dels filtres, primer de tot trobem el filtre per data.

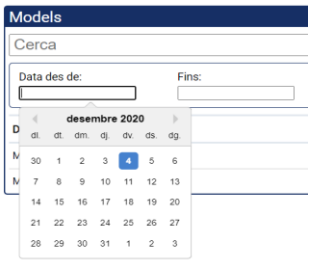


Figura 41 - Captura de pantalla filtre per data

Aquest filtre està disponible en tots els apartats de descàrrega de fitxers. Després, tal i com s'ha comentat, segons l'apartat també hi ha sub-filtres.

Figura 42 - Captura de pantalla filtre general

A nivell de *Back-End*, a la base de dades hi ha guardat la informació respectiva als fitxers disponibles (Nom, extensió, ruta del servidor, propietari, apartat, etc..)

\\DESKTOP-2E4AHSQ\Anexos\00000000A	pdf	0	2020-11-12 23:18:37	2020-11-12 23:18:37	7	00000000A	NOMINA	1	1	USER001-RESUM-COSTOS01

Figura 43 - Dades de fitxer a la Base de dades

Aleshores, l'aplicació web li demana a l'API REST un llistat dels fitxers propietats de l'usuari actual en forma d'*Array* (la crida es fa quan s'entra per primer cop en qualsevol apartat de descàrregues).

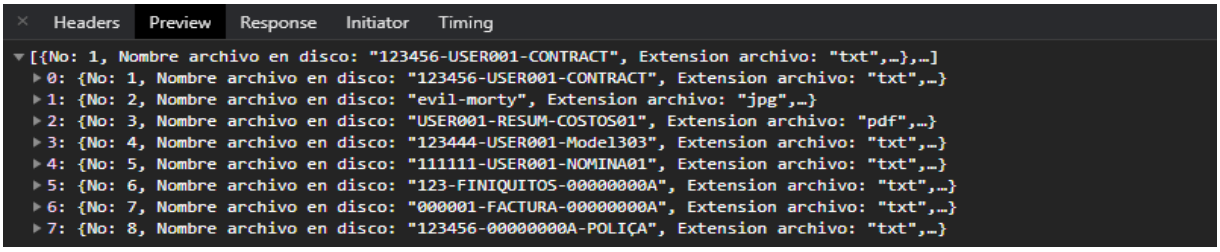


Figura 44 - Resposta http a petició llistat de fitxers

Desprès, com que a la base de dades també hi ha l'adreça on es troben els fitxers, quan l'usuari demana descarregar un fitxer, l'agafa d'allà.

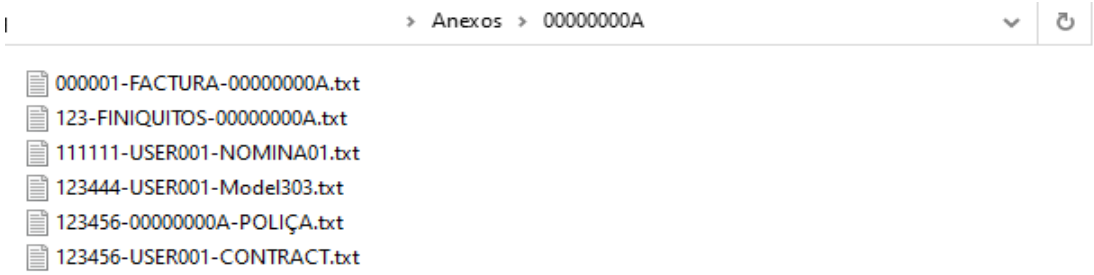


Figura 45 - Estructura fitxers servidor anexes

Finalment, per a descarregar aquest fitxer a l'ordinador de l'usuari, utilitza un Blob (Binary Large Object).

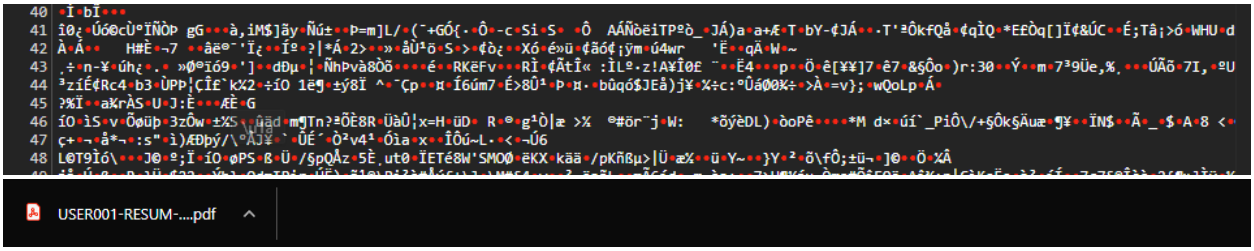


Figura 46 - Resposta BLOB i fitxer descarregat

5. Mòdul Reserva de cita prèvia

Per a la reserva de cita prèvia, s'ha optat per integrar l'*API* que ofereix *Office365*. Actualment, la gestoria ja compta amb aquest servei a la seva pàgina web.

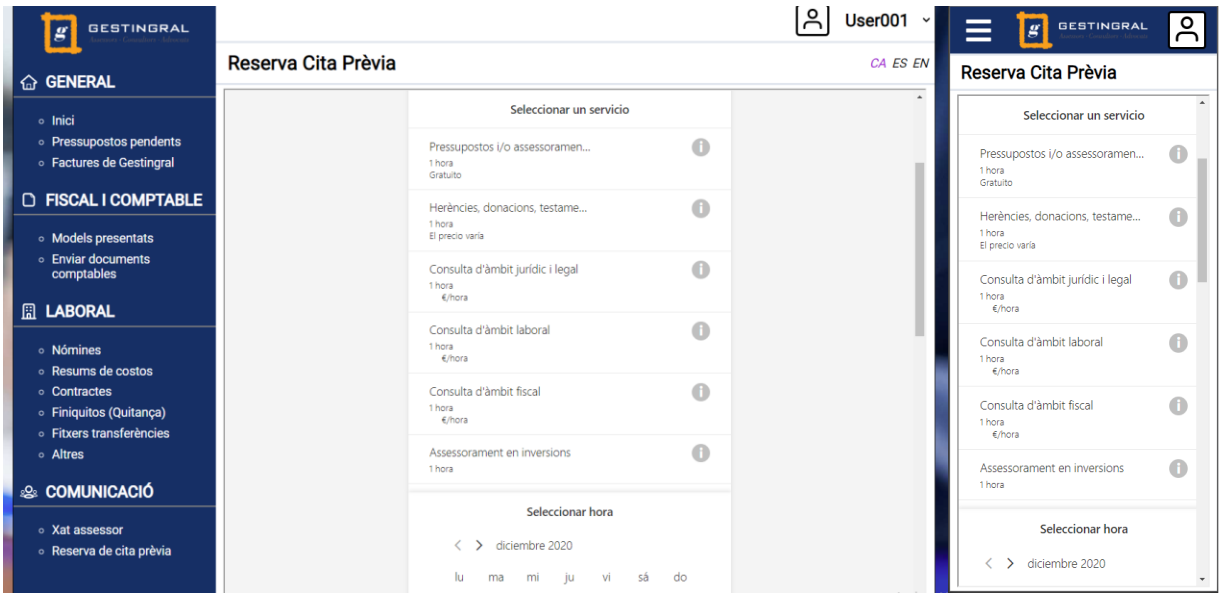


Figura 47 - Captura de pantalla escena Cita prèvia (Desktop/mobile)

Tenint aquest servei ja contractat, s'ha valorat que no surt a compte replicar un servei semblant desenvolupat per l'autor del projecte. Amb aquest servei de *Office365*, les cites queden reservades directament a l'agenda del gestor que s'ha seleccionat.

6. Mòdul de missatgeria Client-Gestor

A nivell de individual, el mòdul de "xat", és el que té un funcionament més complex (més quantitat de crides a servidor).

Al entrar dintre del mòdul, l'aplicació web demana a l'API el llistat d'assessors disponibles i també torna a demanar la llista de missatges sense llegir (al iniciar l'aplicació, també ha demanat aquesta llista per poder mostrar la notificació).

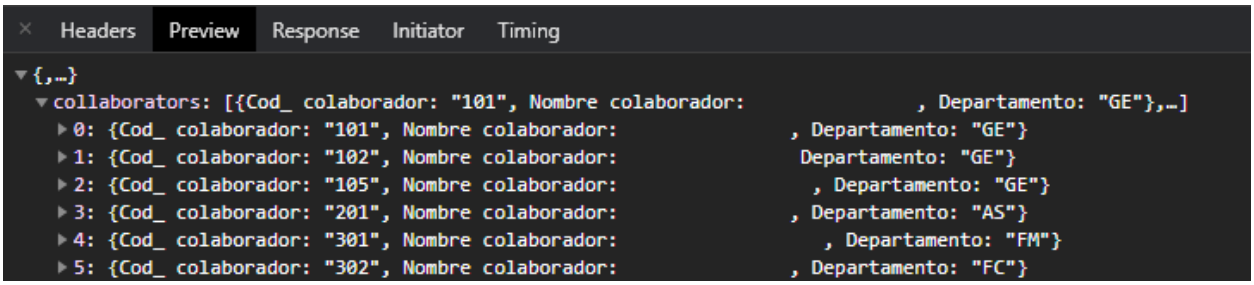


Figura 48 - Resposta http a petició llista col·laboradors

Quan l'usuari entra en aquest apartat, a la part superior pot seleccionar amb quin treballador de la gestoria vol parlar i també quin usuari vol utilitzar per la comunicació (en el cas que en tingui més d'un, per exemple, si té una empresa).

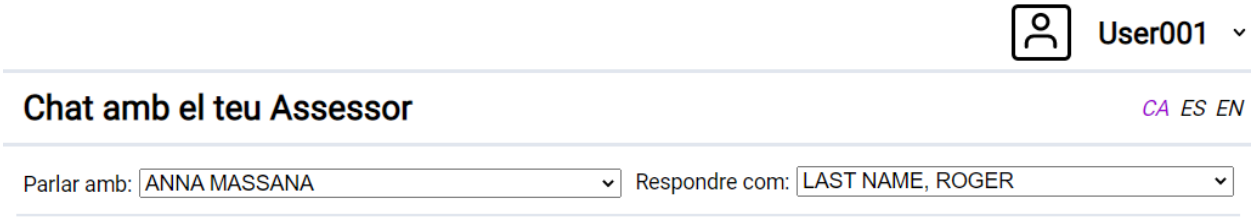


Figura 49 - Selectors de conversa

En el cas que l'usuari tingui missatges pendents en alguna de les diferents converses, apareixerà una notificació en el menú esquerra i en el selector de converses.

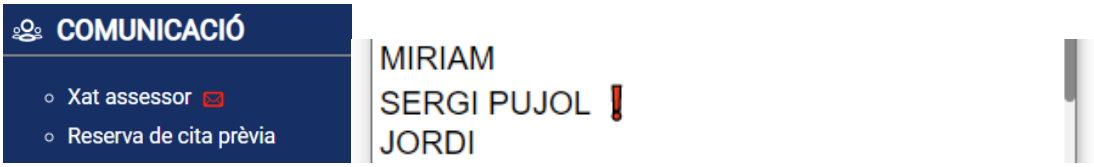


Figura 50 - Exemple de visualització de notificacions

Un cop decideix amb qui vol parlar, en el cas que tingui missatges amb aquest gestor, aquests es mostraran per pantalla (a nivell estètic, a petició del client, s'ha buscat certa similitud amb aplicacions de missatgeria populars per a facilitar l'adaptació dels usuaris). Si s'està dintre de la conversa, l'aplicació pregunta al servidor cada 5 minuts si hi ha missatges nous i en cas afirmatius els baixarà.

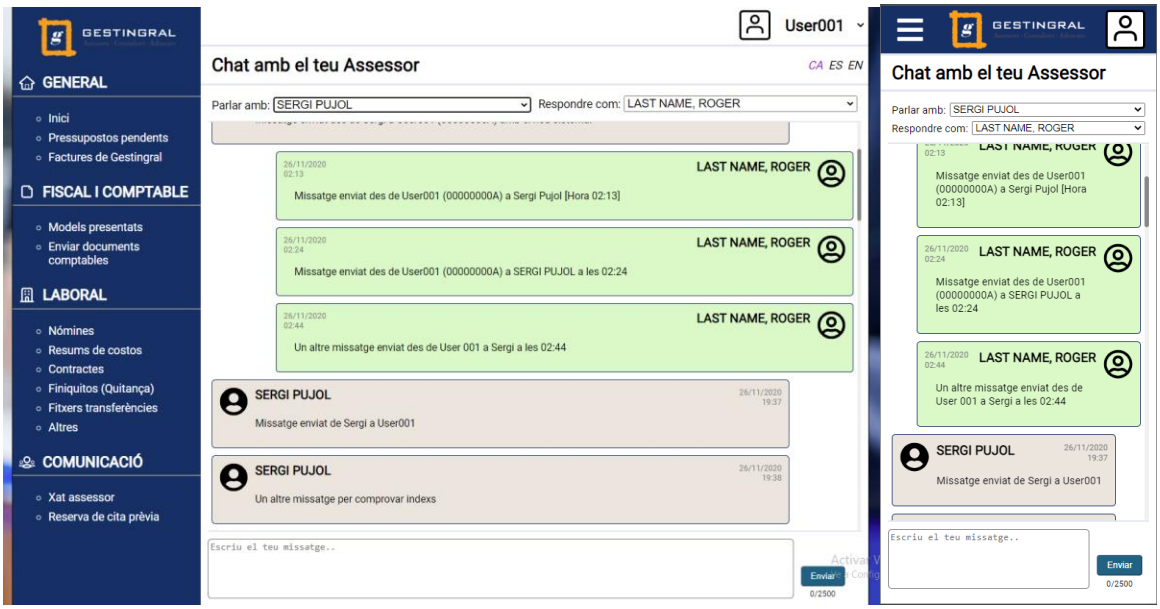


Figura 51 - Captura de pantalla escena Xat Assessor (Desktop/mobile)

Els missatges l'API els envia en forma d'*Array*, amb una longitud màxima de 30 missatges.

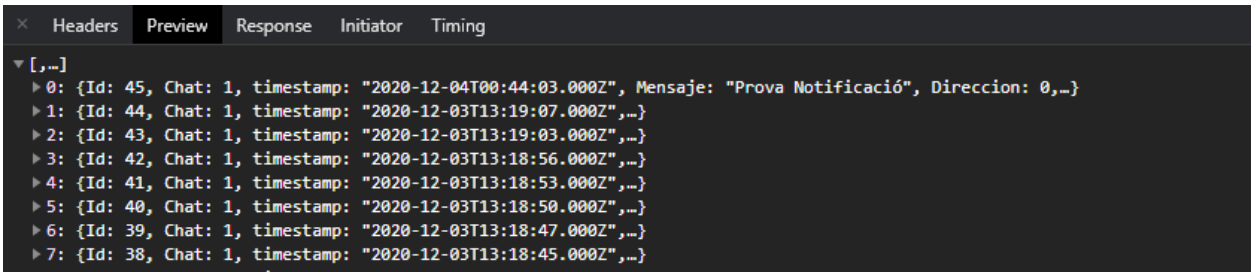


Figura 52 - Resposta http a petició missatges

En el cas que la conversa tingui més de 30 missatges, l'usuari pot anar al principi d'aquesta i clicar a "Carregar més" per a descarregar-se la conversa sencera (el qual demana a l'API la mateixa petició però sense la limitació de 30 elements).

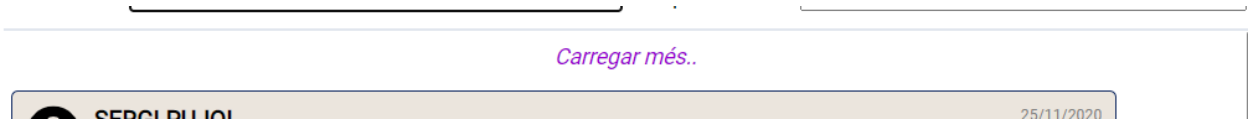


Figura 53 - Opció de Carregar més missatges

Per a enviar missatges, l'usuari ha d'estar dintre la conversa i a la part inferior d'ella pot escriure el missatge i enviar-lo premen el botó de la dreta. La llargada màxima del missatge és de 2500 caràcters, els quals s'indiquen a la part inferior dreta de la pantalla.

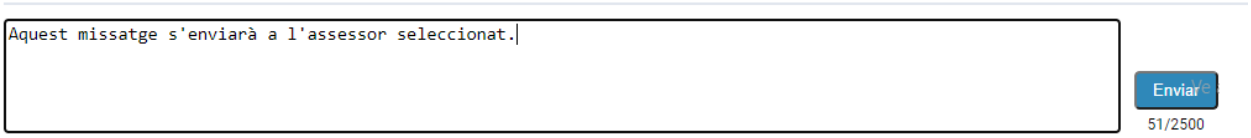


Figura 54 - Zona input dels missatges

Quan l'aplicació envia el missatge a l'*API*, aquesta l'escriu a la base de dades sota la *ID* de la conversa que coincideixi tant en Origen com a Destí.

46	58	2020-12-04 02:00:47	Aquest missatge s'enviarà a l'assessor seleccionat...	1	0	LAST NAME, ROGER

Figura 55 - Visualització de missatge en Base de dades

Per exemple, en el missatge que s'acaba d'enviar, el número de missatge és el 46 i la *ID* de la conversa és el 58 (tots els missatges entre aquest usuari i aquest assessor, tenen el mateix *ID*). Aquesta identificació de conversa, com s'ha vist en la part corresponent a aquest mòdul de la memòria, es troba guardat en una altra taula.

Id	Colaborador	Usuario
58	101	00000000A

Figura 56 - Visualització de ID de conversa en Base de dades

7. Eina de comunicació Gestor-Client

Per a facilitar la feina dels treballadors de la gestoria, s'ha desenvolupat una eina per a poder gestionar de manera externa les converses del mòdul de comunicació.

A nivell de lògica interna, és molt semblant, amb la principal diferència que es comprova que l'usuari sigui un col·laborador de l'empresa. També, aquests poden enviar missatges a qualsevol usuari (en comptes de només a treballadors).

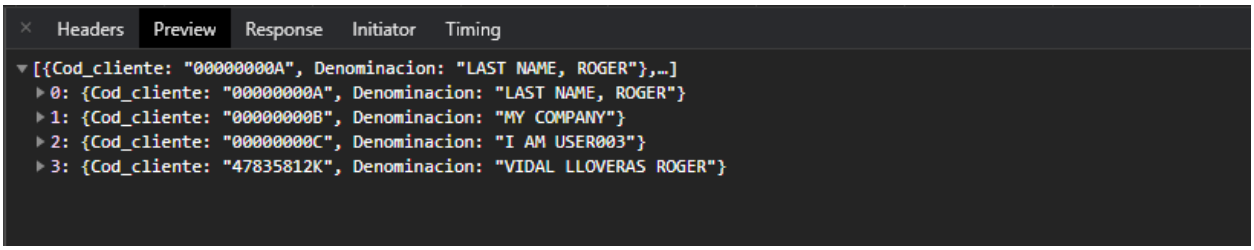


Figura 57 - Resposta http a petició llistat de clients

A la primera connexió del dia, l'eina demana l'*API* la llista actualitzada d'usuaris disponibles. Un cop ja la té, els gestors poden enviar missatges com si estiguessin a l'aplicació, però amb una *UI* més enfocada a aquest tipus de tasques. També, com que els treballadors han de passar moltes hores davant de l'ordinador, s'ha afegit l'opció per canviar de tema visual (clar o fosc).

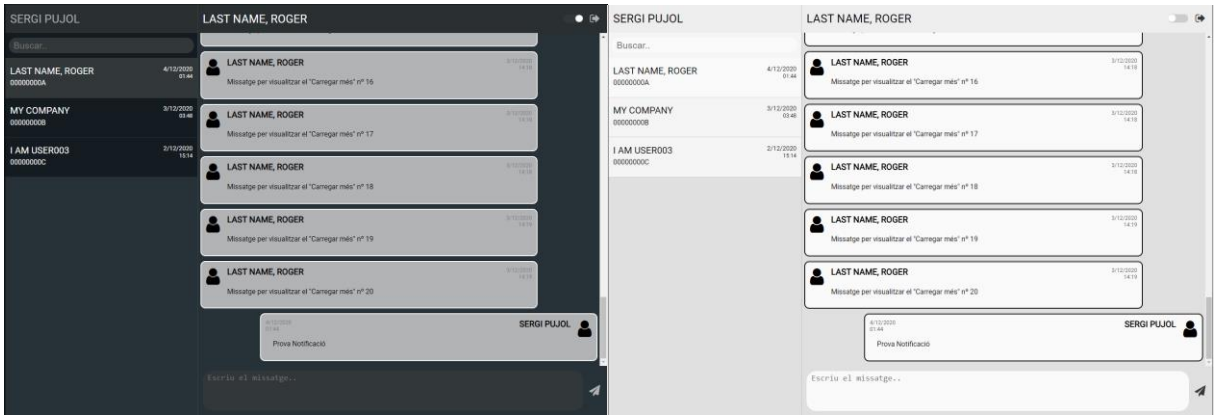


Figura 58 - Captura de pantalla eina missatgeria (Dark/Light)

CAPÍTOL 6: CONCLUSIONS I LÍNIES DE FUTUR

1. Conclusions

L'objectiu principal d'aquest projecte, era dissenyar i desenvolupar una plataforma web per a una gestoria a on els seus clients puguin accedir a diferents dades personals i realitzar diverses tasques. Tenint això en compte, aquest objectiu s'ha assolit de manera satisfactòria, doncs tot i que sempre es poden millorar aspectes del producte, s'ha aconseguit un projecte sòlid que compleix les expectatives.

Una de les etapes inicials del projecte, va ser realitzar un anàlisi tant de tecnologies disponibles com de solucions comercials semblants a el producte que es volia desenvolupar. En la part de tecnologies, m'agradaria destacar la gran quantitat d'opcions que hi ha al disponibles, doncs tot i que es va limitar la recerca a tres opcions, sense massa esforç es podria treure una llista d'unes altres vint amb característiques molt semblants. En aquest sentit, crec que la millor conclusió que es pot treure, es saber analitzar quines oportunitats t'ofereix cada opció a part de les estrictament tecnològiques (ofertes de treball, introducció a una altra tecnologia, etc..). Per altra banda, en la part d'anàlisi de competidors comercials, tot i que l'objectiu principal era determinar si era necessari realitzar el producte, va resultar ser molt interessant veure les característiques de tots ells. Considero que a l'hora de desenvolupar un projecte d'aquestes característiques, amb aquest primer anàlisi, podem veure quines son les característiques que considerem que no fan be la resta d'opcions i fer una versió pròpia millorada.

Durant el desenvolupament del projecte, estic realment satisfet de la quantitat de coneixements que he assolit amb tant poc temps. Per exemple, abans de començar el projecte mai havia treballat amb SQL. A hores d'ara, em veuria capaç de realitzar un projecte on SQL fos un dels pilars fonamentals. També, crec que és interessant reconèixer que si ara tornés a començar aquest mateix projecte faria moltes coses diferents. Des de el meu punt de vista, això significa que hi ha hagut un procés d'aprenentatge correcte.

Finalment, al començar a treballar amb la idea del projecte, una de les principals preocupacions que tenia en ment era el desconeixement de la majoria de tecnologies necessàries per a realitzar-lo i la falta d'objectivitat a l'hora de precisar si seria capaç de desenvolupar-lo. Ara que s'ha acabat el desenvolupament del projecte, la sensació és que tot i que s'ha hagut de treballar molt, tenint una bona base hi ha suficients recursos a l'abast de tothom per a poder assolir gairebé qualsevol objectiu a nivell tecnològic. Considero que a nivell personal, el fet d'haver treballat en aquest projecte i haver-me sortit de la meva zona de confort, m'ha ajudat molt a canviar la manera de pensar. Al començar el projecte, la meva primera resposta va ser que no tenia clar si ho podria fer. Ara que l'he acabat, considero que la primera resposta sempre ha de ser que sí, i esforçar-se per poder-ho complir.

2. Línies de futur

Un cop acabat el projecte, es pot veure que la majoria de punts plantejats durant tot el desenvolupament s'han assolit, no obstant, hi ha alguns que per certs motius s'han deixat de banda i d'altre que no s'han arribat a plantejar però podrien ser molt interessants per la plataforma.

- **Millorar SQL *Queries*:** Desprès d'haver treballat per primer cop amb SQL, considero que tot i que té una entrada molt senzilla, desprès té una corba d'aprenentatge bastant desequilibrada. Fer les primeres *queries* és molt senzill, però un cop entres en el camp dels "*joins*", "*unions*", etc.. la diferència amb la resta de llenguatges és important. És per aquest motiu, que tot i que estan desenvolupades totes les *queries*, en un treball futur es podrien unificar gran part d'elles per reduir-les aproximadament a la meitat.

- **Millorar i netejar codi client:** Durant tot el desenvolupament, s'ha procurat anar netejant el codi i aplicant nous conceptes apressos referents al desenvolupament amb React. Tot i això, desprès de treballar durant aproximadament tres mesos amb el projecte, ara considero que moltes coses de les que es van fer al principi es podrien millorar a nivell de codi.

- **Mòdul Seguretat Social:** Dintre el plantejament inicial del projecte, aquest mòdul estava inclòs, però degut a la necessitat d'incloure un altre mòdul i que el client vol treballar més amb la funcionalitat d'aquest, no s'ha acabat desenvolupant. La idea d'aquest mòdul, és permetre a autònoms i petits empresaris la possibilitat de gestionar altes i baixes de la seguretat social de manera fàcil i automatitzada.

- **Mòdul Enviar Documents:** Un dels mòduls que actualment es troba dintre el projecte però no està activat, és el de enviar documents. La idea inicial era treballar amb la mateixa API que hem creat per la resta del projecte, no obstant, durant el transcurs del projecte el client ha expressat la intenció d'integrar l'API d'un programari que utilitzen de manera interna actualment.

- **Estètica General:** Des de que es va començar el projecte, es va deixar clar que l'autor del projecte és de perfil programador, no dissenyador. És per aquest motiu, que una possible opció de cara al futur per a millorar l'estètica de l'aplicació web, passi per a contractar els serveis d'un dissenyador.

- **Eina Missatgeria:** En el transcurs d'aquest projecte, l'eina de missatgeria s'ha desenvolupat de manera ràpida per a que compleixi les necessitats, però no s'ha tingut massa en compte l'estètica ni la possibilitat d'afegir més funcionalitats. És per això, que un possible pas de cara el futur, sigui fer de nou aquesta eina i incloure més funcionalitats i una estètica més treballada.

- ***Chatbot*:** Des d'abans de començar a treballar amb el projecte, el client de la gestoria va mostrar interès en desenvolupar un *chatbot* que serveixi com a filtre per a preguntes bàsiques. Tot i que finalment s'hagi prioritzat el mòdul de missatgeria, en un futur pròxim es podria afegir aquesta funcionalitat a l'aplicació.

- **Generalitzar:** Una de les possibles opcions amb la plataforma creada, és convertir-la en un producte comercial que pugui servir per a qualsevol gestoria. Tenint això en compte, seria molt interessant adaptar el codi per que de manera molt senzilla es pugui personalitzar segons les necessitats del client (actualment, tot i que es pot fer sense cap problema, s'ha de canviar directament el codi font).

A part de totes aquestes possibles millores, al tractar-se d'un disseny modular, sempre queda oberta la possibilitat d'afegir nous mòduls a l'aplicació. També, segons el tracte final amb el client, hi ha la possibilitat d'estandarditzar la plataforma per tal de poder-la comercialitzar de manera personalitzada a diferents interessats.

Bibliografia

1. INE - IneBase [Internet]. España: Instituto Nacional de Estadística [data de l'última actualització 23/09/2020; data de consulta 28/09/2020]. Disponible a: https://www.ine.es/dyngs/INEbase/listaoperaciones.html

2. Diseño web adaptable [Internet]. España: Wikipedia, La enciclopedia libre; [data de l'última actualització 15/07/2020; data de consulta 28/09/2020]. Disponible a: https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable

3. La digitalització de l'empresa en un món post COVID-19 [Internet]. España: Gencat; [data de l'última actualització 23/07/2020; data de consulta 08/10/2020]. Disponible a: https://www.accio.gencat.cat/ca/serveis/banc-coneixement/cercador/BancConeixement/digitalitzacio-empresa-mon-post-covid-19

4. What You Need to Know About Chatbot Development [Internet]. US: Medium; [data de l'última actualització 12/03/2019; data de consulta 08/10/2020]. Disponible a: https://chatbotslife.com/what-you-need-to-know-about-chatbot-development-4900e9fbf702

5. Agile software development [Internet]. US: Wikipedia; [data de l'última actualització 17/11/2020; data de consulta 22/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Agile_software_development

6. Top 10 software de gestión para asesorías fiscales [Internet]. España: asesorias; [data de l'última actualització 26/02/2019; data de consulta 09/10/2020]. Disponible a: https://asesorias.com/empresas/programas-gratis/software-para-asesorias/

7. TOP 8 de software para asesorías, gestorías y consultorías [Internet]. España: SoftDoit; [data de l'última actualització 27/03/2020; data de consulta 09/10/2020]. Disponible a: https://www.softwaredoit.es/software-gestoria-asesoria-consultoria/index.html

8. Los 10 mejores programas para Asesorías y Despachos Profesionales [Internet]. España: SoftwarePara; [data de l'última actualització 25/02/2019; data de consulta 09/10/2020]. Disponible a: https://softwarepara.net/asesorias-despachos-profesionales/

9. Suasor ERP para despachos [Internet] España: Summar Tecnología y Gestión SA; [data de consulta 09/10/2020]. Disponible a: https://www.summar.es/software-asesorias/

10. Enterprise resource planning [Internet] US: Wikipedia, the free encyclopedia; [data de l'última actualització 22/09/2020; data de consulta 09/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Enterprise_resource_planning

11. Software de gestión para Asesorías y Despachos Profesionales [Internet] España: Sage Spain; [data de consulta 09/10/2020]. Disponible a: https://www.sage.com/es-es/asesorias-y-despachos/

12. Software de gestión en la nube para despachos [Internet] España: Sudespacho.net; [data de consulta 09/10/2020]. Disponible a: https://www.sudespacho.net/

13. The Progressive JavaScript Framework - Vue [Internet] US: Evan You; 2014 [data de consulta 10/10/2020]. Disponible a: https://vuejs.org/

14. Single-page application [Internet]. España: Wikipedia, La enciclopedia libre; [data de l'última actualització 29/05/2020; data de consulta 28/09/2020]. Disponible a: https://es.wikipedia.org/wiki/Single-page_application

15. Superheroic JavaScript MVW Framework - Angular [Internet]. US: Google; [data de consulta 10/10/2020]. Disponible a: https://angularjs.org/

16. Command-line interface [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 08/10/2020; data de consulta 10/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Command-line_interface

17. A JavaScript library for building user interfaces. [Internet] US: Facebook Inc; [data de consulta 10/10/2020]. Disponible en: https://reactjs.org/

18. JAVASCRIPT FRAMEWORKS VS LIBRARIES—WHAT'S THE DIFFERENCE? [Internet] US: Scott Morris; [data de consulta 10/10/2020]. Disponible a: https://skillcrush.com/blog/javascript-frameworks-vs-libraries/

19. Angular vs React vs Vue: Which Framework to Choose in 2020 [Internet] US: CodeinWp; [data de l'última actualització 09/08/2020; data de consulta 10/10/2020]. Disponible a: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

20. Representational state transfer [Internet]. US: Wikipedia, the free encyclopedia; 2001 [data de l'última actualització 07/10/2020; data de consulta 10/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Representational_state_transfer

21. Introducción a Express/Node [Internet]. US: Mozilla Firefox - MDN web docs; [data de consulta 10/10/2020]. Disponible a: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

22. Koa, next generation web framework [Internet]. US: Koa; [data de consulta 10/10/2020]. Disponible a: https://koajs.com/#

23. The Simple, Secure Framework Developers Trust [Internet]. US: Sideway INC; [data de consulta: 10/10/2020]. Disponible a: https://hapi.dev/

24. Mootools a compact javascript framework [Internet]. US: Mootools Developers ; [data de consulta: 21/11/2020]. Disponible a: https://mootools.net/

25. jQuery write less, do more. Javascript library [Internet]. US: The jQuery Foundation; [data de consulta: 21/11/2020]. Disponible a: https://jquery.com/

26.Dojo Toolkit [Internet]. US: OpenJS Foundation; [data de consulta: 21/11/2020]. Disponible a: https://dojotoolkit.org/

27. Software Architecture [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 07/10/2020; data de consulta 11/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Software_architecture

28. Model–view–controller [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 28/09/2020; data de consulta 12/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

29. MVC vs. MVVM: How a Website Communicates With Its Data Models [Internet]. US: Hackernoon; [data de l'última actualització 16/10/2017; data de consulta 12/10/2020]. Disponible a: https://hackernoon.com/mvc-vs-mvvm-how-a-website-communicates-with-its-data-models-18553877bf7d

30. ARQUITECTURAS DE APLICACIONES WEB DE 2, 3 Y N CAPAS [Internet]. ES: Mundo Android Web; [data de consulta 12/10/2020]. Disponible a: https://tec755.wordpress.com/infografia/

31. El árbol web: qué es y para qué sirve [Internet]. ES: Inbound Cycle; [data de l'última actualització 22/10/2014; data de consulta 18/10/2020]. Disponible a: https://www.inboundcycle.com/blog-de-inbound-marketing/bid/195257/el-rbol-web-qu-es-y-para-qu-sirve

32. ¿Qué es un feed RSS? [Internet]. ES: Microsiervos; [data de consulta 29/10/2020]. Disponible a: https://www.microsiervos.com/archivo/general/que-es-un-feed-rss.html

33. React (web framework) [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 06/11/2020; data de consulta 16/10/2020]. Disponible a: https://en.wikipedia.org/wiki/React_(web_framework)

34. Hypertext Markup Language [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 29/10/2020; data de consulta 16/10/2020]. Disponible a: https://en.wikipedia.org/wiki/HTML

35. JavaScript (JS). [Internet]. US: MDN web docs; [data de l'última actualització 01/11/2020; data de consulta 16/10/2020]. Disponible a: https://developer.mozilla.org/en-US/docs/Web/JavaScript

36. Cascading Style Sheets (CSS) [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 08/11/2020; data de consulta 17/10/2020]. Disponible a: https://en.wikipedia.org/wiki/CSS

37. CSS with superpowers [Internet]. US: Github team; [data de consulta 20/10/2020]. Disponible a: https://sass-lang.com/

38. Node.js NPM [Internet]. US: W3schools; [data de consulta 17/10/2020]. Disponible a: https://www.w3schools.com/nodejs/nodejs_npm.asp

39. Axios Promise based HTTP client for the browser and node.js [Internet]. US: Github; [data de l'última actualització 03/11/2020; data de consulta 18/10/2020]. Disponible a: https://github.com/axios/axios

40. jQuery - A javascript Library [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 16/11/2020; data de consulta 21/11/2020]. Disponible a: https://en.wikipedia.org/wiki/JQuery

41. Node.js - A Javascript Runtime [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 04/11/2020; data de consulta 16/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Node.js

42. Express. Fast, unopinionated, minimalist web framework for Node.js [Internet] US: OpenJS Foundation [data de consulta 16/10/2020]. Disponible a: https://expressjs.com/

43. JavaScript Object Notation [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 01/11/2020; data de consulta 16/10/2020]. Disponible a: https://en.wikipedia.org/wiki/JSON

44. JSON Web Token [Internet]. US: Auth0; [data de consulta 17/10/2020]. Disponible a: https://jwt.io/

45. NPM - Bcrypt for Javascript. [Internet] US: GitHub; [data de consulta 26/10/2020]. Disponible a: https://www.npmjs.com/package/bcryptjs

46. SQL - Structured Query Language [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 01/11/2020; data de consulta 24/10/2020]. Disponible a: https://en.wikipedia.org/wiki/SQL

47. MySQL Database Service [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 29/10/2020; data de consulta 23/10/2020]. Disponible a: https://en.wikipedia.org/wiki/MySQL

48. Visual Studio Code, Code editing. Redefined [Internet]. US: Microsoft; [data de consulta 18/10/2020]. Disponible a: https://code.visualstudio.com/

49. GIT [Internet]. US: Software Freedom Conservancy; [data de consulta 18/10/2020]. Disponible a: https://git-scm.com/

50. GitHub [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 06/11/2020; data de consulta 18/10/2020]. Disponible a: https://en.wikipedia.org/wiki/GitHub

51. Heroku. [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 26/08/2020; data de consulta 03/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Heroku

52. XAMPP Apache + MariaDB + PHP + Perl. [Internet]. US: Apache Friends; [data de consulta 19/10/2020]. Disponible a: https://www.apachefriends.org/es/index.html

53. Bringing MySQL to the web [Internet]. US: phpMyAdmin contributors; [data de consulta 23/10/2020]. Disponible a: https://www.phpmyadmin.net/

54. The Collaboration Platform for API Development [Internet]. US: Postman INC; [data de consulta 20/10/2020]. Disponible a: https://www.postman.com/

55. Adobe Photoshop. [Internet]. US: Wikipedia, the free encyclopedia; [data de l'última actualització 03/11/2020; data de consulta 17/10/2020]. Disponible a: https://en.wikipedia.org/wiki/Adobe_Photoshop

56. Zoom. [Internet]. US: Zoom Video Communications Inc.; [data de consulta 21/10/2020]. Disponible a: https://zoom.us/

57. Universal Database Tool. [Internet]. US: DBeaver Community; [data de consulta 01/11/2020]. Disponible a: https://dbeaver.io/

58. Responsive Web Design Patterns [Internet]. US: Google Developers; [data de l'última actualització 26/11/2019; data de consulta 14/10/2020]. Disponible a: https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns

59. Roboto [Internet] US: Wikipedia, the free encyclopedia; [data de l'última actualització 06/08/2020; data de consulta 24/11/2020]. Disponible a: https://es.wikipedia.org/wiki/Roboto

60. Roboto [Internet] US: Google Fonts; [data de consulta: 24/11/2020]. Disponible a: https://fonts.google.com/specimen/Roboto

61. Sans-serif [Internet] US: Wikipedia, the free encyclopedia; [data de l'última actualització 20/11/2020; data de consulta 24/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Sans-serif

62. APACHE LICENSE, VERSION 2.0 [Internet] US: The apache software foundation; [data de consulta: 24/11/2020]. Disponible a: https://www.apache.org/licenses/LICENSE-2.0

63. Font Awesome [Internet] US: Fonticons, INC; [data de consulta: 25/10/2020]. Disponible a: https://fontawesome.com/

64. TypIcons [Internet] US: Stephen Hutchings; [data de consulta: 25/10/2020]. Disponible a: https://www.s-ings.com/typicons/

65. HeroIcons [Internet] US: Hero Icons Team; [data de consulta: 25/10/2020] Disponible a: https://heroicons.dev/

66. React Icons [Internet] US: GitHub; [data de consulta 25/10/2020] Disponible a: https://react-icons.github.io/react-icons/

67. Qué es: UX y UI [Internet]. ES: Andrea Cantú - Intuitivamente; [data de la última actualització 30/03/2020; data de consulta 25/10/2020]. Disponible a: https://blog.acantu.com/que-es-ux-y-ui/

68. Create, read, update and delete. [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 27/10/2020; data de consulta 02/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

69. Utilización del middleware. [Internet] US: Github; [data de consulta 21/10/2020]. Disponible a: https://expressjs.com/es/guide/using-middleware.html

70. Bcrypt. [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 08/10/2020; data de consulta 07/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Bcrypt

71. Blowfish (cipher) [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 10/10/2020; data de consulta 07/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Blowfish_(cipher)

72. Rainbow table. [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 02/10/2020; data de consulta 07/11/2020]. Disponible a: https://en.wikipedia.org/wiki/Rainbow_table

73. Database that can handle >500 millions rows [Internet] US: Stackoverflow; [data de consulta 21/11/2020]. Disponible a: https://stackoverflow.com/questions/3779088/database-that-can-handle-500-millions-rows

74. InnoDB Limits [Internet] US: MySQL; [data de consulta 21/11/2020]. Disponible a: https://dev.mysql.com/doc/refman/8.0/en/innodb-limits.html

75. ReactJS Contexts [Internet] US: Facebook Inc; [data de consulta 13/11/2020]. Disponible a: https://reactjs.org/docs/context.html

76. REACT ROUTER [Internet] US: Facebook; [data de consulta 16/10/2020]. Disponible a: https://reactrouter.com/

77. Window.localStorage [Internet] US: Mozilla web Docs; [data de consulta 16/11/2020]. Disponible a: https://developer.mozilla.org/es/docs/Web/API/Window/localStorage

78. HTTP cookie [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 10/11/2020; data de consulta 16/11/2020]. Disponible a: https://en.wikipedia.org/wiki/HTTP_cookie

79. Cross-site scripting [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 29/11/2020; data de consulta 01/12/2020]. Disponible a: https://en.wikipedia.org/wiki/Cross-site_scripting

80. Cross-site request forgery [Internet] US: Wikipedia, the free encyclopedia; [data de la última actualització 23/11/2020; data de consulta 01/12/2020]. Disponible a: https://en.wikipedia.org/wiki/Cross-site_request_forgery

81. What does it mean to escape a string? [Internet] US: Stackoverflow; [data de la consulta 01/12/2020]. Disponible a: https://stackoverflow.com/questions/10646142/what-does-it-mean-to-escape-a-string

82. Anti CSRF Tokens ASP.NET [Internet] US: Owasp; [data de consulta 01/12/2020]. Disponible a: https://owasp.org/www-community/Anti_CRSF_Tokens_ASP-NET

83. Functional Programming VS Object Oriented Programming (OOP) Which is better….?. [Internet] US: Medium; 2019 [data de l'última actualització 09/07/2019; data de consulta 19/10/2020]. Disponible a: https://medium.com/@shaistha24/functional-programming-vs-object-oriented-programming-oop-which-is-better-82172e53a526

84. React Components vs. React Hooks. [Internet] US: Medium; 2019 [data de l'última actualització 11/10/2019; data de consulta 19/10/2020]. Disponible a: https://medium.com/better-programming/react-components-vs-react-hooks-52932d4ab6db

Annexos

Annex A: Glossari d'acrònims

API: Application Programming Interface

APP: Application

BBDD: Bases de Dades

CSS: Cascading StyleSheets

DB: DataBase

ERP: Enterprise Resource Planning

HTTP: Hypertext Transfer Protocol

JSON: JavaSript Object Notation

JWT: JSON Web Token

MVC: Model View Controller

NPM: Node Package Manager

PAC: Prova d'Avaluació Continua

REST: Representational State Transfer

SPA: Single Page Application

TFM: Treball Final de Màster

UI: User Interface

UOC: Universitat Oberta de Catalunya

UX: User Experience

Annex B: Lliurables del projecte

rogervidallloveras_informe-treball: Document en format "pdf" corresponent a l'informe de treball de la PAC5.

rogervidallloveras_memoria: Document en format "pdf" corresponent a la memòria del projecte (Aquest document).

rogervidallloveras_presentacio-academica: Arxiu de vídeo en format "mp4" corresponent a la presentació acadèmica. Aquesta va destinada al tribunal d'avaluació.

rogervidallloveras_presentacio-publica: Conjunt de diapositives en format "pptx" corresponent a la presentació pública. Aquesta presentació consta d'un conjunt de transparències automatitzades i amb pistes auditives, destinades al públic general.

rogervidallloveras_instruccions: Document en format "pdf" corresponent a les instruccions per a poder accedir a l'aplicació web.

rogervidallloveras_demostracions: Carpeta amb diferents arxius en format "mkv" corresponents a vídeos de demostració de funcions concretes de la plataforma.

rogervidallloveras_codi-font: Carpeta amb el codi font del client, de l'API i de l'eina de missatgeria.