**Xavier Romeu Martínez**

# SISTEMA DE GESTIÓ DE CONTINGUTS MULTILINGÜE: OPTIMITZANT LA COMUNICACIÓ I L'ACCESSIBILITAT A LA INFORMACIÓ

**TREBALL DE FI DE GRAU**

**dirigit per Jordi Massaguer Pla**

**Grau en Tècniques de Desenvolupament d'Aplicacions Web i Mòbils**

UNIVERSITAT ROVIRA I VIRGILI

**Tarragona**

**2024**

## Resum

Aquest projecte consisteix en la creació d'una pàgina web fullstack. Aquesta resol la necessitat de donar visibilitat a un projecte creat pel fons europeu, on hi han unes directius a seguir en quant a l'estructura, accessibilitat, usabilitat... De forma que qualsevol persona, inclús tenint qualsevol tipus de dificultat com ara ser no oient o presentar visibilitat reduïda, pugui usar-la de totes formes.

Per a realitzar aquest projecte s'ha emprat el framework de PHP: Laravel. Que ens crea tota una estructura amb tot el necessari per crear una aplicació web completa, tant per la part del frontend com la part del backend.

Durant el projecte s'han repassat i aprofundit diferents conceptes apresos durant el grau com pot ser l'anàlisi de requisits i el disseny d'una aplicació web des de zero.

## Resumen

Este proyecto consiste en la creación de una página web fullstack. Esta resuelve la necesidad de dar visibilidad a un proyecto creado por el fondo europeo, donde hay unas directrices a seguir en cuanto a la estructura, accesibilidad, usabilidad... De manera que cualquier persona, incluso teniendo cualquier tipo de dificultad como por ejemplo ser no oyente o presentar visibilidad reducida, pueda usarla de todas formas.

Para realizar este proyecto se ha empleado el framework de PHP: Laravel, que nos crea toda una estructura con todo lo necesario para crear una aplicación web completa, tanto para la parte del frontend como la parte del backend.

Durante el proyecto se han repasado y profundizado diferentes conceptos aprendidos durante el grado como puede ser el análisis de requisitos y el diseño de una aplicación web desde cero.

## Abstract

This project consists of creating a full-stack website. It addresses the need to provide visibility to a project created by the European fund, where there are guidelines to follow regarding structure, accessibility, usability... so that anyone, even if they have any type of difficulty such as being deaf or having reduced visibility, can use it anyway.

To carry out this project, the PHP framework Laravel was used. It creates an entire structure with everything necessary to build a complete web application, both for the frontend and the backend.

During the project, various concepts learned during the degree were reviewed and deepened, such as requirements analysis and designing a web application from scratch.

# Índex de taules

# Índex de figures

# 1 Introducció

Des del fons europeu s'estan duent a terme projectes de tota mena per promoure la inclusivitat entre totes les persones, sense importar el seu gènere, nacionalitat, religió...

Aquest projecte inclou la creació d'una pàgina web per donar visibilitat i informar a un d'aquests col·lectius com son les mares en situacions complicades, com ara tenir dificultats econòmiques o estar en risc d'aïllament per la seva religió o nacionalitat.

Això pel que fa a la part visible, però tanmateix, hi inclou un gestor de continguts, on es crearà, gestionarà i traduirà tota la informació visible a la pàgina als idiomes de totes les associacions participants, és a dir, amb català, castellà, anglès, italià, búlgar i grec.

Des d'aquest gestor també permet publicar notícies i vídeos rellevants. Dins la pàgina hi ha un PDF informatiu, un test per comprovar el nivell de consciència de l'usuari que el realitzi sobre el tema on les preguntes es crearan des del mateix gestor. La pàgina també compta amb un fòrum on tot usuari registrat podrà publicar contingut, però aquest abans haurà de ser aprovat per un usuari autoritzat del gestor, que revisarà que no es publiqui res ofensiu.

A més a més, s'ha realitzat una estratègia de testing per aconseguir validar el correcte funcionament del màxim de components i pàgines possibles del projecte, anomenant aquest tipus d'estratègia de testing coneguda com a coverage testing. Els testos realitzats son Feature test, més exactament indicats per cobrir casos d'ús a alt nivell i fluxos de treball de l'usuari sobre el programari.

# 2 Descripció general del projecte

## 2.1 Tecnologies i eines emprades

En aquest apartat s'expliquen les diverses tecnologies i aplicacions utilitzades per realitzar aquest projecte.

### *2.1.1 Visual Studio Code*

Visual Studio Code [1] és un editor de codi redefinit i optimitzat per a la creació i depuració d'aplicacions web, mòbils i cloud modernes.

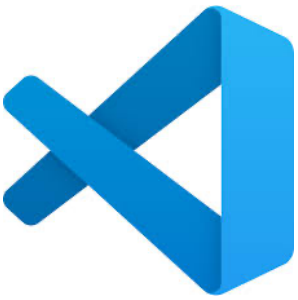

**Figura 1.** Visual Studio Code Logo

### *2.1.2 PHP*

PHP [2] (acrònim recursiu de PHP: Hypertext Preprocessor) és un llenguatge de codi obert molt popular especialment adequat per al desenvolupament web i que pot ser incrustat en HTML.

El que distingeix PHP de llenguatges del costat del client com Javascript és que el codi és executat en el servidor, generant HTML i enviant-lo al client. El client rebrà el resultat d'executar l'script, encara que no coneixerà el codi. El servidor web pot ser configurat fins i tot perquè processi tots els fitxers HTML amb PHP, de manera que els usuaris no puguin saber què es té sota la lògica.

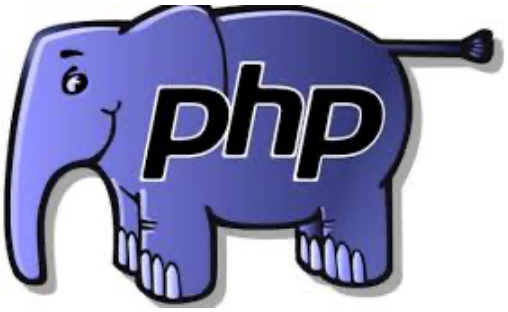


**Figura 2.** PHP Logo

### *2.1.3 Laravel*

Laravel [3] és un framework PHP per al desenvolupament d'aplicacions web. Es distingeix d'altres frameworks fràcies a la seva sintaxi elegant i concisa, el seu enfocament en el desenvolupament basat en proves i la seva rica col·lecció de paquets i eines.


**Figura 3.** Laravel Logo

#### 2.1.3.1 Artisan

Artisan, el seu sistema de comandaments, otorga al framework gran poder i grans facilitats i possibilitats als programadors per crear controladors, entitats o actualitzar la base de dades entre moltes altres coses.

#### 2.1.3.2 Blade

El seu motor de plantilles, anomenat Blade, ofereix nombroses possibilitats per crear pàgines visualment molt potents i eficaces, capaç d'utilitzar les seves pròpies variables i reutilitzar-les.

#### 2.1.3.3 Migrations

Permet actualitzar i migrar la base de dades una vegada que el desenvolupament ja està començat i ja s'han fet canvis en el codi conforme es requereixi, sense necessitat d'eliminar-la i tornar-la a crear. Gràcies a això, el risc de perdre dades, siguin del valor que siguin, és mínim. A més, gràcies al seu Schema Builder, no cal utilitzar SQL, ja que compta amb un sistema intuïtiu en PHP per fer-ho més fàcil.

### *2.1.4 Bootstrap*

Bootstrap [4] és un framework FOSS desenvolupat per Twitter, que conté components HTML, CSS, i JS per facilitar la implementació d'una interfície gràfica que sigui responsiva i visualment consistent.


**Figura 4.** Bootstrap Logo

### *2.1.5 Git i GitHub*

Git [5] és un sistema de control de versions distribuït que permet gestionar els canvis en els fitxers i el seguiment de l'historial de revisions d'un projecte de programació. Amb Git, diversos desenvolupadors poden treballar en un mateix projecte de manera col·laborativa, mantenint un seguiment dels canvis realitzats per cada membre de l'equip.


**Figura 5.** GitHub Logo

### *2.1.6 MySQL*

MySQL [6] és un sistema de gestió de bases de dades (DBMS, per les seves sigles en anglès) de codi obert desenvolupat per Oracle.

SQL i MySQL són dues eines diferents. SQL és el llenguatge universal de consulta de bases de dades, mentre que MySQL és un sistema de gestió de bases de dades específic que utilitza SQL per interactuar amb la base de dades.

MySQL utilitza un model de base de dades relacional, on les dades s'organitzen en taules amb relacions definides. Utilitza el llenguatge SQL per realitzar consultes i manipular dades. Els seus components principals inclouen un servidor de base de dades, motors d'emmagatzematge i clients que permeten la interacció amb la base de dades.


**Figura 6.** MySQL Logo

### *2.1.7 NPM*

NPM [7] és un gestor de paquets en l'ecosistema de JavaScript. Permet als desenvolupadors instal·lar, gestionar o compartir llibreries i mòduls de codi reutilitzables de JavaScript.

**Figura 7.** NPM Logo

Per altra banda, s'han utilitzat un conjunt d'eines per tal d'efectuar el projecte:

• L'eina MagicDraw [8] per a realitzar els esquemes UML del disseny de l'aplicació (diagrames de casos d'ús, classes, activitat i seqüències).


**Figura 8.** MagicDraw Logo

• L'eina draw.io de GoogleDrive per a realitzar els esquemes d'estructura de fitxers de l'aplicació, base de dades…


**Figura 9.** Draw.io Logo

## 2.2 Disseny de l'aplicació web

### *2.2.1 Identitat de l'aplicació web*

Pel que fa al disseny de l'aplicació web, l'empresa ha proporcionat una paleta de colors i una estructura que demana la unió europea per tal de complir amb els requisits i sigui apta per tota persona que vulgui accedir i navegar còmodament per aquesta.

Com podem observar ens proporcionen un logo per al projecte i quina gamma de colors s'han d'utilitzar al aplicatiu web.

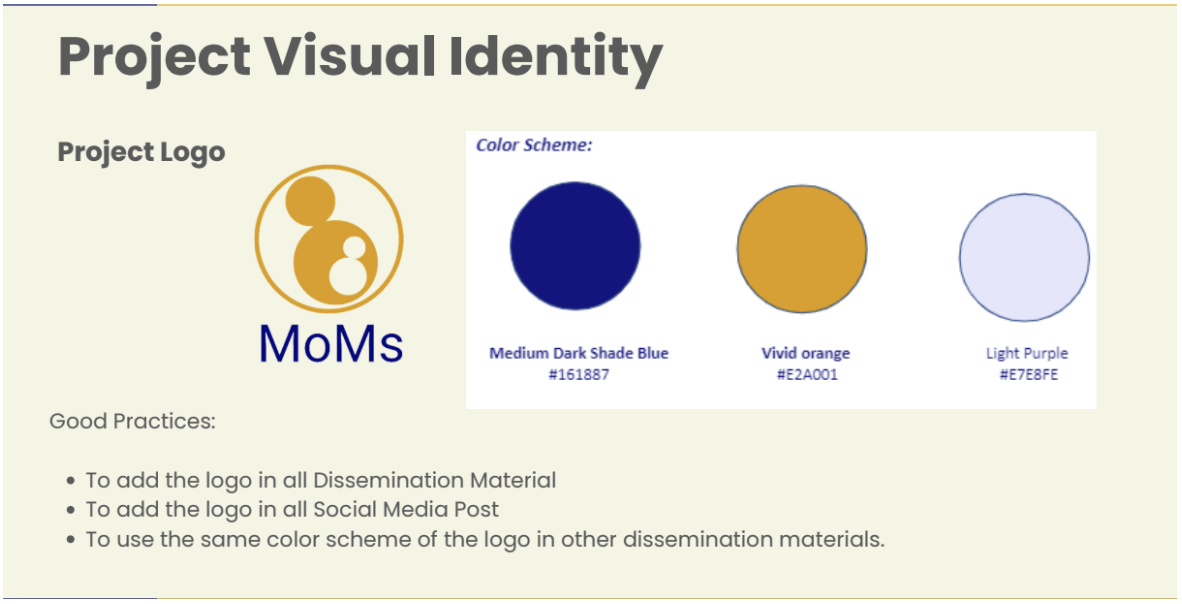


**Figura 10.** Identitat visual del projecte

L'empresa també ha adjuntat els estils que faran ús per a les newsletters i els flyers que crearan per la seva part, de manera que l'estil de la pàgina tingui una semblança i concordança amb aquest. Gràcies a això es proporciona una imatge unificada a tot el projecte.

## 2.3 Apartat legal

Dins de l'apartat legal de la pàgina s'ha de tenir en compte diversos aspectes, entre els quals es troben la llei de protecció de les dades personals, ja que es registren usuaris i proporcionen correus electrònics.

En aquest cas no hi han però en cas que n'hi hagués s'hauria d'especificar quines cookies s'utilitzen, i que emmagatzemen, i mostrar l'avís d'aquestes.

Es mostrarà l'avís amb el text RGPD (Reglament General de Protecció de Dades), qui és el propietari d'aquestes dades i quins son els drets de l'usuari sobre aquestes. Tot això serà mostrat dins l'apartat privacitat de la pàgina, accessible des del footer.

Dins el footer hi haurà un altre enllaç a l'apartat legal on s'informa qui és el propietari de l'aplicatiu web.

I per finalitzar amb l'apartat legal es tindrà en compte la llei de propietat intel·lectual, sobre la qual en sería propietar del dret d'autoria com a únic autor de la pàgina però declarant que el dret d'explotació sobre l'empresa que ha gestionat el projecte, en aquest cas Open Europe.

## 2.4 Disseny de la interfície gràfica

A l'hora d'implementar la interfície gràfica, primer s'ha dissenyat conjuntament amb l'empresa alguns mockups de diverses pantalles sobre com es volen implementar i, finalment, s'han anat modificant durant el desenvolupament per poder adaptar-ho al que realment es vol i es pot fer, així com petits canvis o decisions que s'han pres durant la implementació.

Per a aquesta interfície cal comentar diverses decisions de disseny preses. A l'hora de demanar dades a l'usuari (formularis amb HTML) es fa mitjançant els estils i formularis que ofereix Bootstrap de forma que tots tindran un disseny uniforme i visualment verificat pels creadors de la biblioteca.

Un altra decisió de disseny és basada en que està pensada per ser utilitzada en tot tipus de dispositius, des d'ordinadors, tablets i fins a mòbils.

### *2.4.1 Disseny de les pantalles de l'aplicació web*

2.4.1.1 Navbar o barra de navegació superior

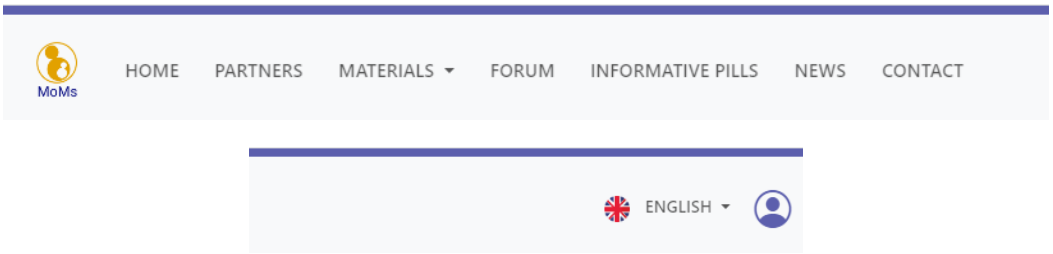


**Figura 11.** Barra de navegació superior

La navbar és un element que conté la navegació principal present en totes les pàgines. Aquesta presenta enllaços que indiquen la pàgina en la què és situat l'usuari, així com el menú per canviar d'idiomes i l'opció d'accedir amb l'usuari corresponent a la pàgina.

Posteriorment veiem com es mostrada aquesta navbar en pantalles més petites, on està oculta fins que no pitgem la icona d'estil "hamburger", on es mostren totes les opcions de navegació del menú d'igual manera que a pantalles amb més resolució.

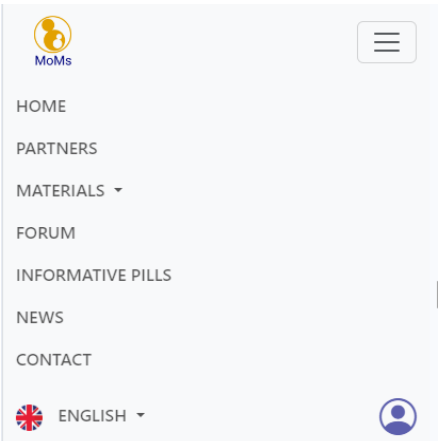


**Figura 12.** Barra de navegació mòbil

### 2.4.1.2 Perfil de l'usuari

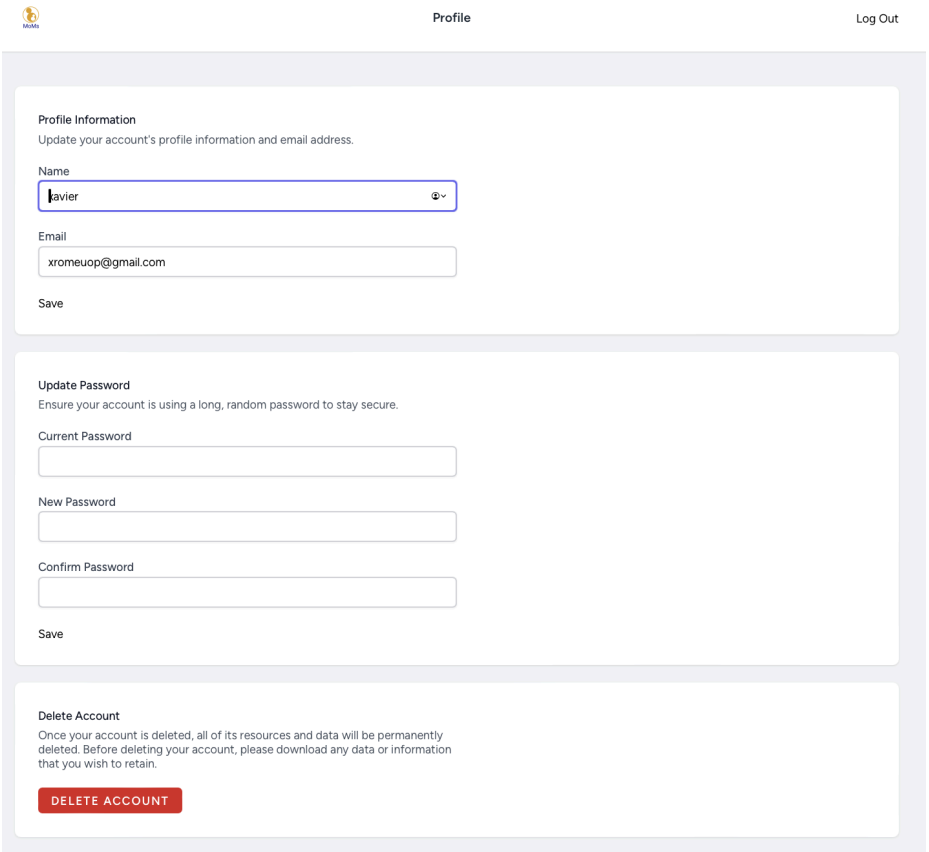


**Figura 13.** Pantalla de perfil d'usuari

Per a la pàgina de perfil de l'usuari s'ha fet servir la que proporciona el mateix Laravel per defecte, adaptant la barra superior a l'estil de la web, i amb la paleta de colors d'aquesta.

### 2.4.1.3 Pàgina principal de l'aplicació web

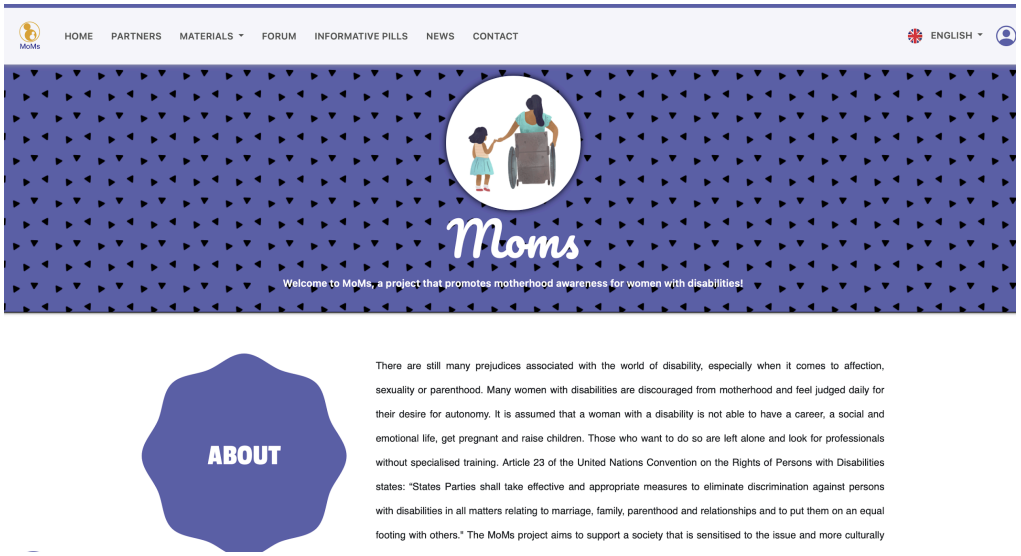


**Figura 14.** Vista de la pàgina principal

A la pàgina principal es trobarà l'usuari cada cop que s'entra a la web. Es pot observar com a totes i cadascuna de les pàgines es trobarà la mateixa barra superior amb el menú que ens permet navegar per cadascun dels apartats de la web, canviar l'idioma o accedir a la secció del perfil en cas d'haver iniciat sessió, si l'usuari encara no ha iniciat sessió ens redigirà a la pàgina d'inici. Des d'aquesta també se'ns permetrà arribar a la pàgina de registre.


**Figura 15.** Vista de la timeline del projecte

Dins la pàgina principal ens trobem la presentació de que tracta el projecte. Com podem visualitzar sota del timeline hi ha imatges de diversos events, degut a que es tracta d'un projecte que dura dos anys, actualment només ha tingut lloc un d'aquests, per aquesta raó hem inclòs momentàniament imatges del lloc on es realitzaran els futurs.

#### 2.4.1.4 Partners del projecte

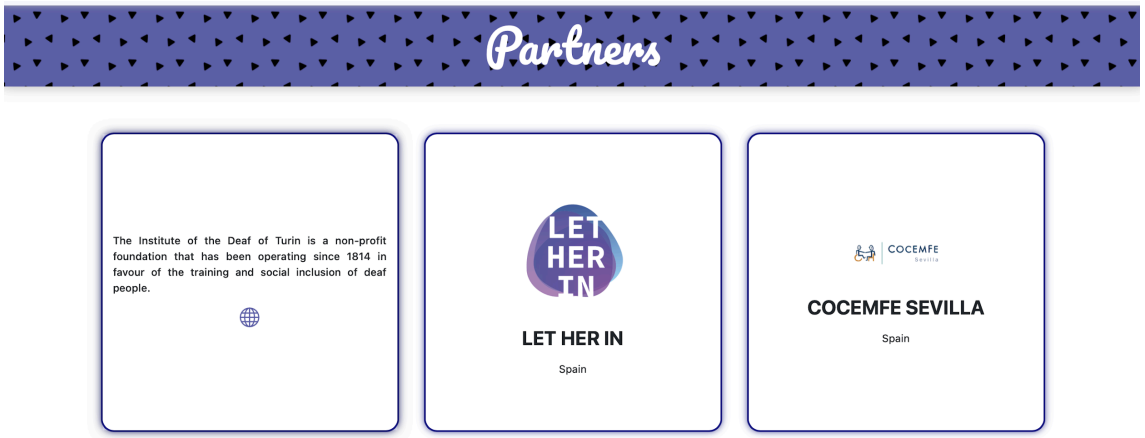


**Figura 16.** Vista de la pàgina partners

A continuació, com podem observar a la part superior es pot veure una especie de header que conté a quin apartat de la web ens trobem. Aquests headers els trobarem al llarg de tota la pàgina i per tant tenen tots el mateix estil, ja que són un element que es reutilitza canviant el text que correspon a cada pàgina.

Dins de la pàgina partners tenim unes cards on es mostra el logotip de les entitats que hi participen i el seu país, al fer hover per damunt es mostrarà la descripció d'aquesta i un icona que ens redigirà al seu lloc web.

Aquestes cards apareixen al carregar amb una animació que les farà sorgir des de l'esquerra de la pàgina.

#### 2.4.1.5 Research book

La pàgina de research book ens mostrarà una descripció del perquè es fa aquesta mena de llibre, un botó per descarregar el pdf i aquest pdf visualitzat a la pàgina directament, de manera que ens permetrà navegar com si el tinguéssim descarregat localment.

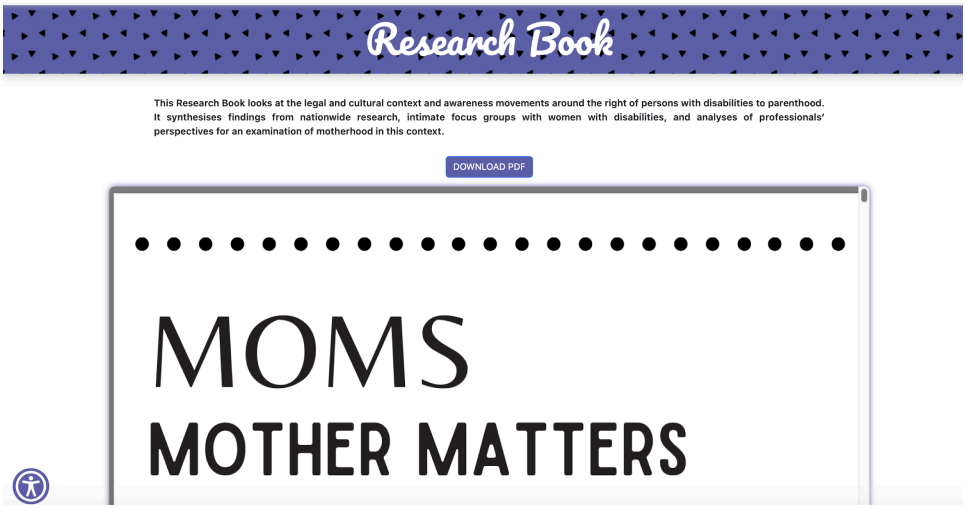


**Figura 17.** Vista de la pàgina research book

#### 2.4.1.6 Self assessment

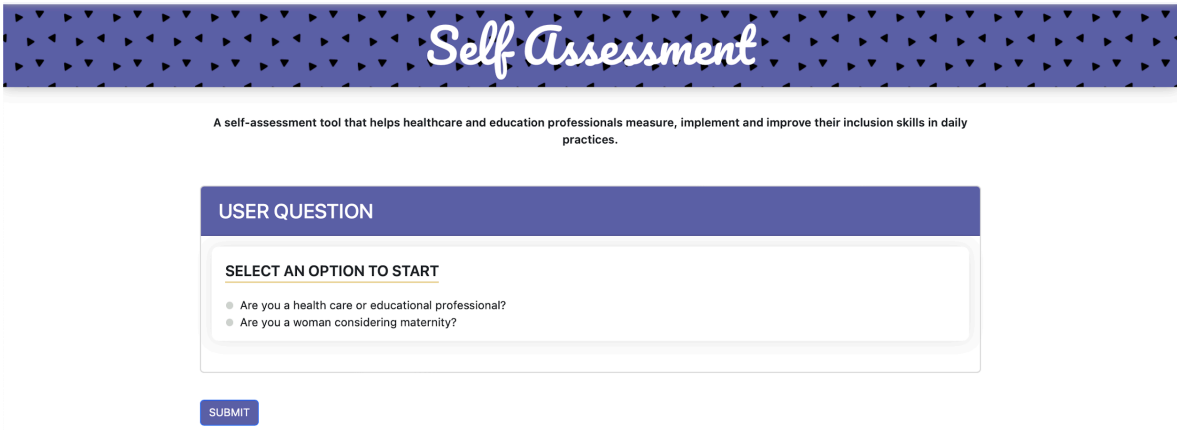


**Figura 18.** Vista de la pàgina self assessment

Pel que fa a la pàgina de self assessment ens trobem amb un qüestionari amb diverses opcions. Posteriorment ens mostrarà unes preguntes acord amb la primera opció seleccionada.

#### 2.4.1.7 Fòrum

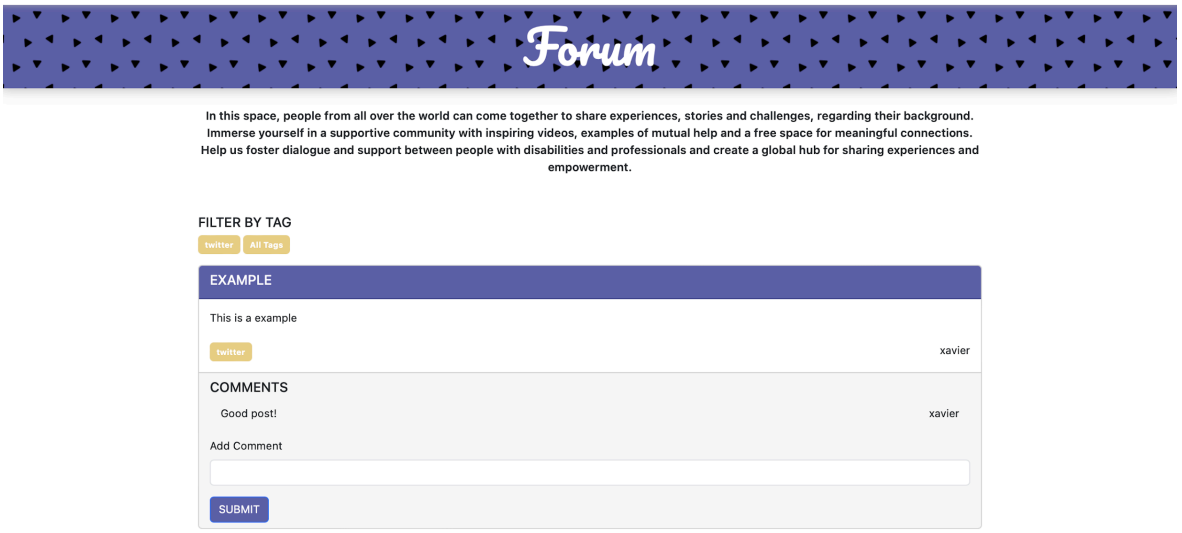


**Figura 19.** Vista de la pàgina del fòrum

A la pàgina del fòrum tenim una descripció de per a que s'ha de fer servir aquest fòrum i que publicar, un filtre que funciona amb les tags assignades als posts, els posts i els comentaris que hi hagin en aquest post.

2.4.1.8 News

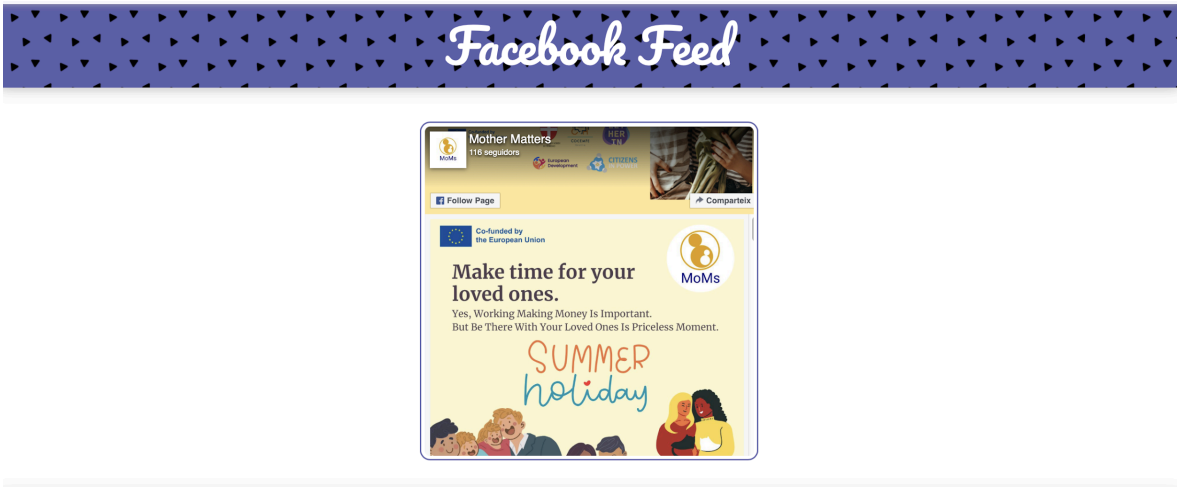


**Figura 20.** Vista de la pàgina news, apartat facebook feed

A la pàgina de news tenim diversos apartats, primerament un widget amb el grup de facebook. Aquest ens permetrà adherir-nos i redirigir-nos per veure el contingut.


**Figura 21.** Vista de la pàgina news, apartat newsletter

El següent apartat dins la pàgina de news es el de les newsletters que s'han publicat sobre aquest projecte, són unes cards que ens redigeix a una pàgina amb aquesta newsletter en format pdf.

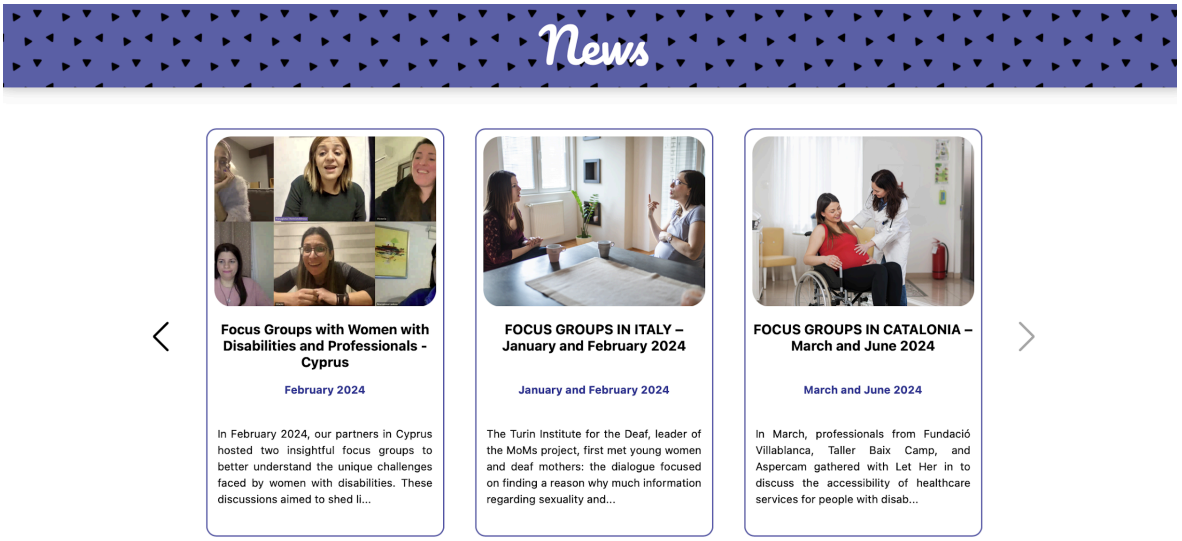


**Figura 22.** Vista de la pàgina news, apartat news

I per finalitzar amb aquesta pàgina de news tindrem les notícies en si sobre el projecte, unes cards amb una imatge sobre l'event a explicar, un títol, data en que va ocórrer aquest event i una descripció del que va succeir.

### 2.4.1.9 Contacte

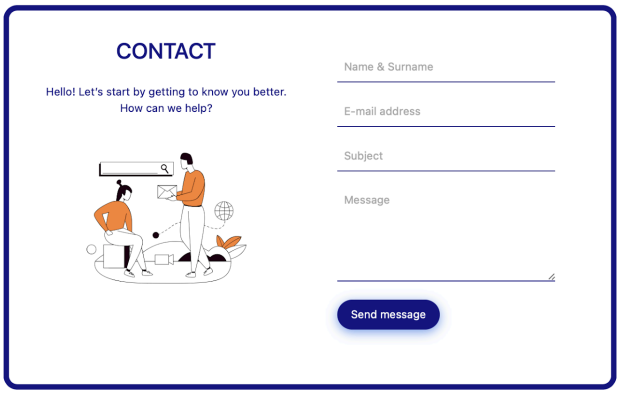


**Figura 23.** Vista de la pàgina contact

La pàgina disposa d'un apartat de contacte on es poden enviar suggerències que fan cap al correu del gestor del projecte on serà respostes per aquest mateix mitjà.

#### 2.4.1.10 Footer

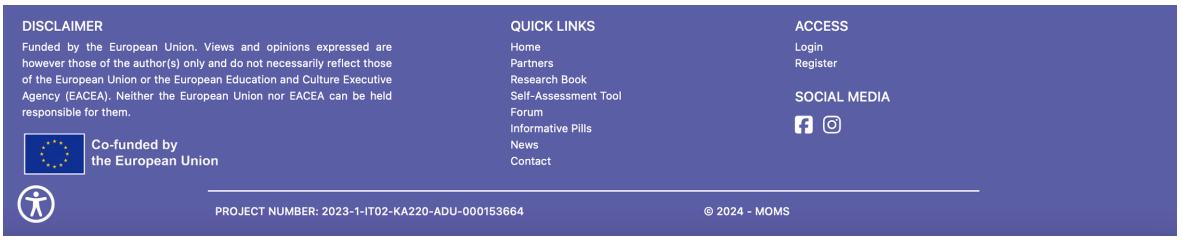


**Figura 24.** Vista del footer de la pàgina

Pel que fa a la part inferior de cada pàgina ens trobem amb el footer, amb un disclaimer sobre el projecte, indicacions de que es un projecte cofundat per la unió europea, un menú de navegació ràpida i enllaços a les diferents xarxes socials que té el projecte.

### *2.4.2 Disseny de les pantalles del gestor de l'aplicació web*

#### 2.4.2.1 Sidebar

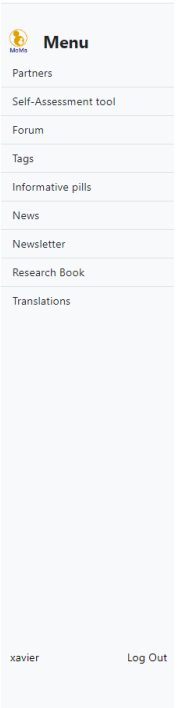


**Figura 25.** Menú lateral del gestor

Pel que fa al gestor de la pàgina disposarem d'un menú lateral a la part esquerra que ens permetrà navegar entre tots els apartats d'aquest gestor, accedir a la pàgina del perfil de l'usuari, tancar sessió o fent clic al logotip del projecte ens redigirà a fora del backoffice de la pàgina.

### 2.4.2.2 Consulta d'instàncies

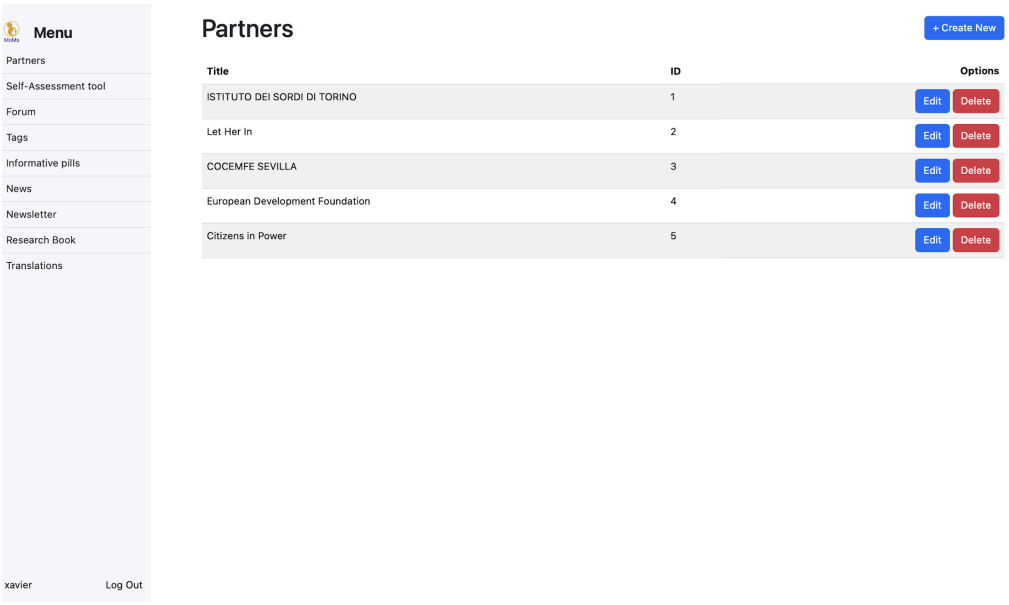


**Figura 26.** Vista d'un taulell d'instàncies

Dins cada apartat tindrem el títol, un botó per crear noves instàncies, baix apareixerà una taula que mostra totes les instàncies d'aquesta categoria, mostrant les dades rellevants d'aquesta per identificar-les, dins aquesta taula tindrem a cada línia instància dos botons, un per editar aquesta i un altre per eliminar-la.

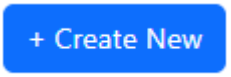


**Figura 27.** Botó per afegir una nova entrada


**Figura 28.** Botó per editar una instància

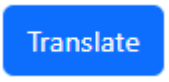


**Figura 29.** Botó per traduir una instància

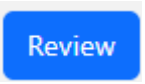


**Figura 30.** Botó per revisar un post del fòrum


**Figura 31.** Botons de revisió d'un post del fòrum

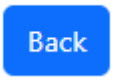


**Figura 32.** Botó per tirar enrere

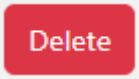


**Figura 33.** Botó per eliminar una instància

### 2.4.2.3 Creació de noves instàncies

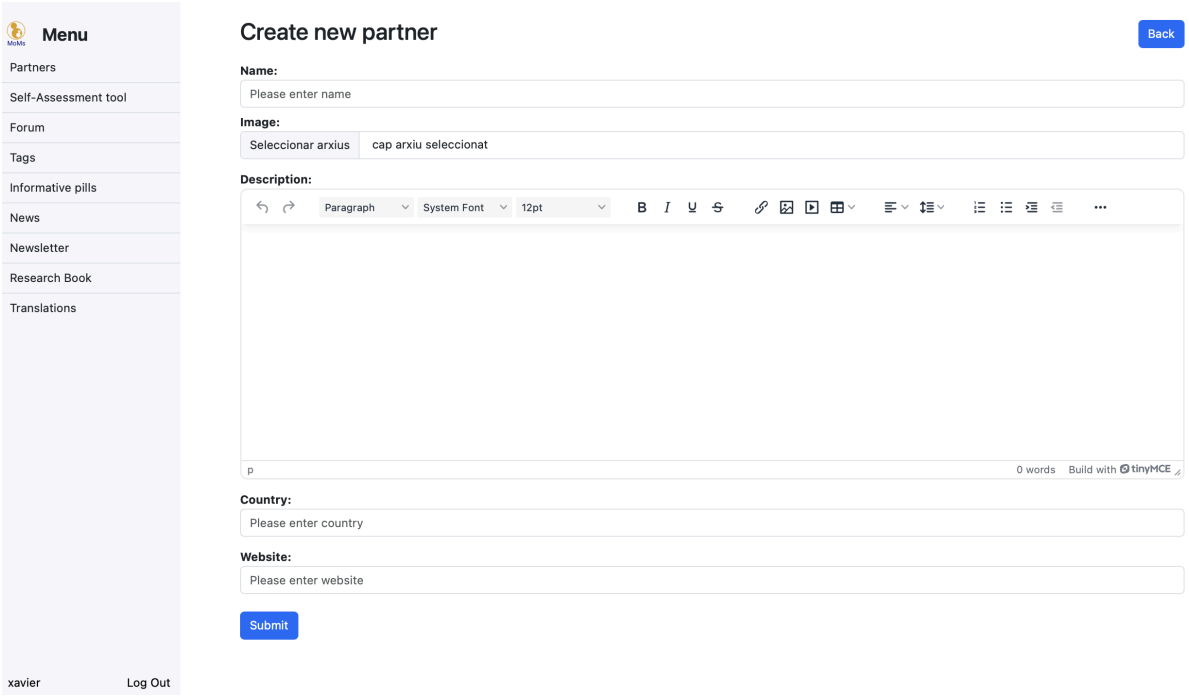


**Figura 34.** Vista d'un formulari per crear noves instàncies

Pel que fa a la funcionalitat de crear noves instàncies d'una categoria tindrem totes les pàgines amb una disposició similar, un títol indicatiu de l'acció que estem realitzant i un botó per tirar enrere a la part superior de la pàgina, i a la part inferior un formulari on emplenarem totes les dades necessàries per crear aquesta nova instància, i finalment el botó per finalitzar, en cas d'haver algun error ens remarcarà amb roig un missatge baix del camp que no estigui ja bé omplert o omplert incorrectament.

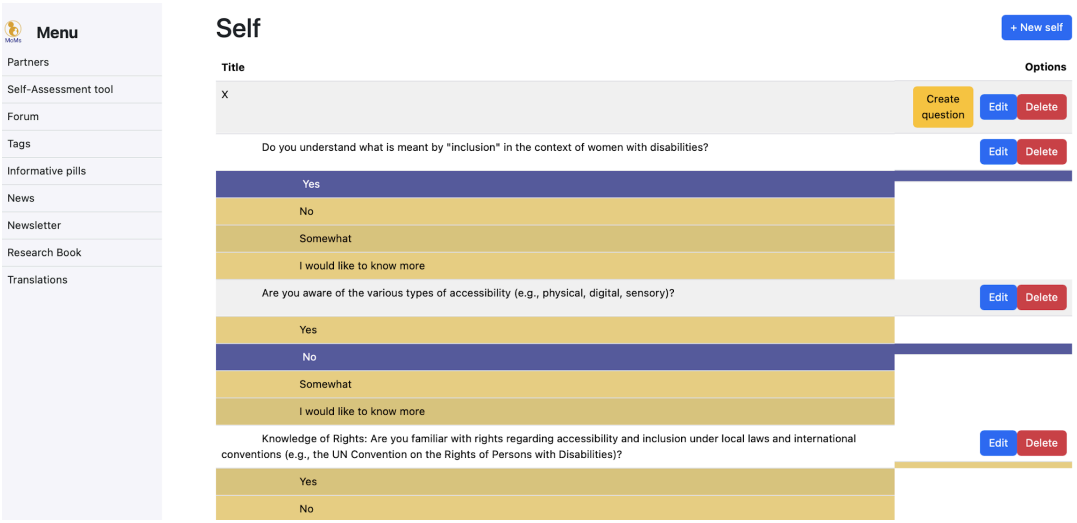


**Figura 35.** Vista per crear noves qüestions i respostes

L'apartat de self assessment és l'únic que tindrà un aparença diferent, ja que crearem qüestions i dins cada qüestió també crearem possibles respostes per marcar sobre aquestes. Tindrem un botó per crear noves qüestions i a la línia on ens mostra aquesta qüestió tindrem el botó per crear les respostes per aquesta, també disposem de les opcions d'editar i eliminar cada instància.

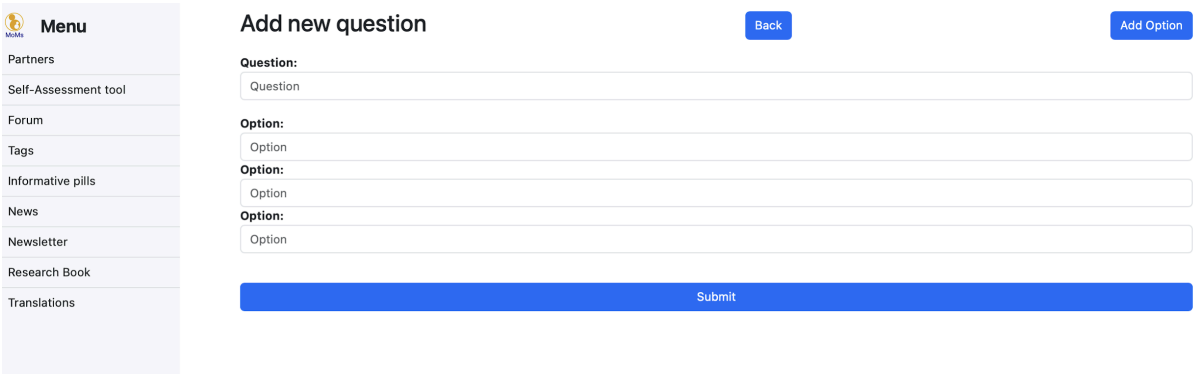


**Figura 36.** Vista per crear noves qüestions

Aquí podem veure el formulari de afegir noves qüestions amb les seves opcions.


**Figura 37.** Vista del filtre de categories de la pàgina

Pel que fa a les traduccions dels textos, disposem d'un filtre per mostrar per categories o apartats de la pàgina dels textos disponibles per traduïr.

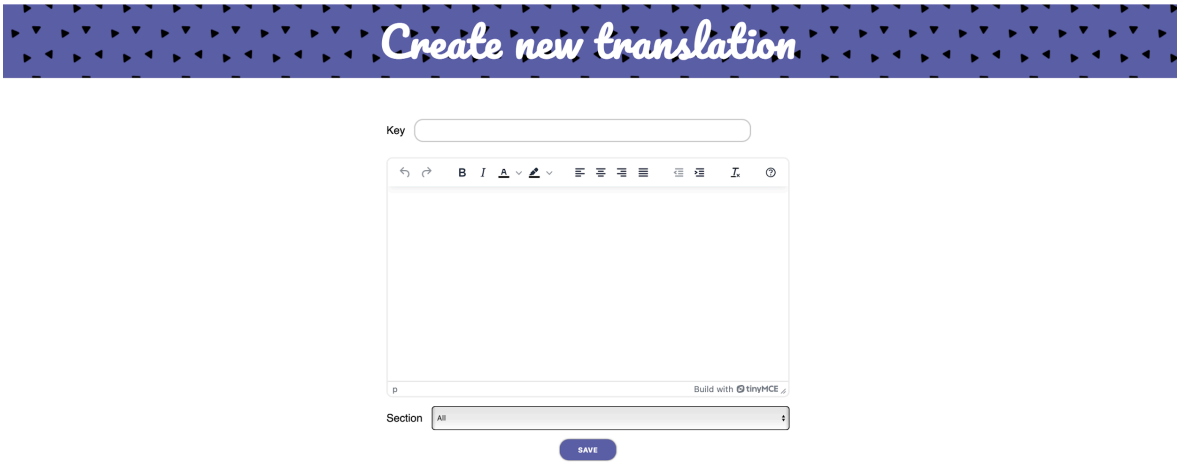


**Figura 38.** Vista de la creació de noves traduccions

El formulari per crear noves traduccions disposarà d'un selector de categoria on s'associa, una key o clau identificativa que serà utilitzada com a identificador dins de la pàgina que la volem incloure, i el text a mostrar que s'ha d'introduir inicialment amb anglès per després traduir-lo a la resta d'idiomes.

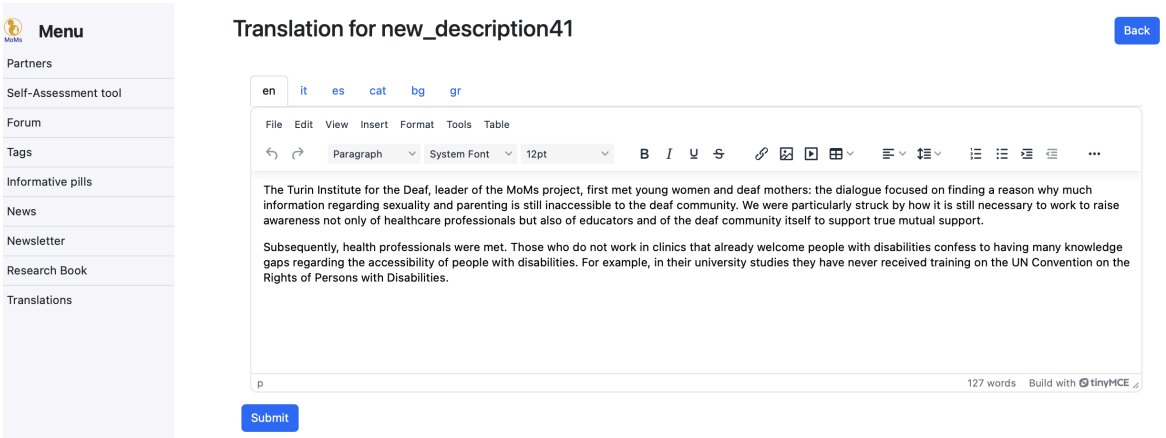


**Figura 39.** Vista d'addició d'idiomes per traduir

A dins de les traduccions de cada text tindrem un textarea amb múltiples pestanyes on inclourem a cadascuna el text amb la traducció a mostrar d'aquest.

### 2.5 Avaluació de costos

Com a costos tindrem associats uns que seran fixos i d'altres variables.

Costos fixos:

- Servidor de hosting: el projecte està allotjat al servei de hosting de ionos on s'està pagant una cuota de 12€ al mes, el qual ens inclou:
  - 200GB d'emmagatzematge.
  - 9GB de RAM.
  - Fins a 100 bases de dades.
  - Ample de banda il·limitat.
  - Direcció de correu professional.
  - Suport online 24/7.
  - SSL inclòs.
- Domini: el preu del domini és gratuït de fer el registre i després té un cost de 10€ / any.

Costos variables:

- Personal: el cost per hora establert pel tutor del projecte ha estat de 50€ / hora.

En conclusió els cost total de fer el projecte per al primer any, tenint en compte que s'ha dedicat 300 hores a la realització completa desde l'estudi, disseny i implementació suposarà un total de 15144€. Els següents anys només es tindran 154€ fixos en costos i en cas de requerir algun canvi es pagarà l'hora sobre la mateixa tarifa al programador destinat a fer els canvis.

# 3 Requisits

## 3.1 Requisits funcionals

Dins dels requisits funcionals de l'aplicació del projecte a continuació es detallen les funcionalitats per pantalla, el diagrama de casos d'ús i especificacions d'aquests casos d'ús.

### *3.1.1 Login*

- Pantalla on l'usuari iniciarà sessió
- Entrada del correu electrònic de l'usuari
- Entrada de la contrasenya de l'usuari
- Opció de canviar a la pantalla de registre
- Opció de canviar a la pantalla de recuperar contrasenya

### *3.1.2 Forgot Password*

- Pantalla on l'usuari introduirà el seu correu electrònic on rebrà indicacions per recuperar la contrasenya
- Entrada del correu electrònic
- Opció de canviar a la pantalla de iniciar sessió

### *3.1.3 Registrar usuari*

- Pantalla on l'usuari es registrarà a l'aplicació
- Entrada del correu electrònic de l'usuari
- Entrada de la contrasenya de l'usuari
- Entrada de la confirmació de la contrasenya de l'usuari
- Opció de canviar a la pantalla de iniciar sessió

### *3.1.4 Crear instància*

- Pantalla on l'usuari crearà una nova instància; una nova notícia, pregunta per al formulari de self assessment, pujada d'un nou video...
- Entrada de les dades requerides
- Pujada d'imatge en cas de que sigui considerat necessària
- Pujada d'enllaç extern

### *3.1.5 Editar instància*

- Pantalla on l'usuari actualitzarà les dades que es necessiti canviar
- Entrada de les noves dades
- Pujada de la nova imatge
- Pujada del nou enllaç

### *3.1.6 Traduir instància*

- Pantalla on l'usuari inclou les traduccions amb els idiomes que esta disponible la web
- Entrada dels textos amb cadascun dels idiomes

### *3.1.7 Revisió dels fòrum*

- Pantalla on es revisarà si el contingut de l'entrada creada per un usuari es adient per ser publicada o no
- Acceptació o declinació del nou post

### *3.1.8 Eliminar instància*

- Botó per eliminar instàncies de cada entitat diferent

### *3.1.9 Diagrama de casos d'ús*

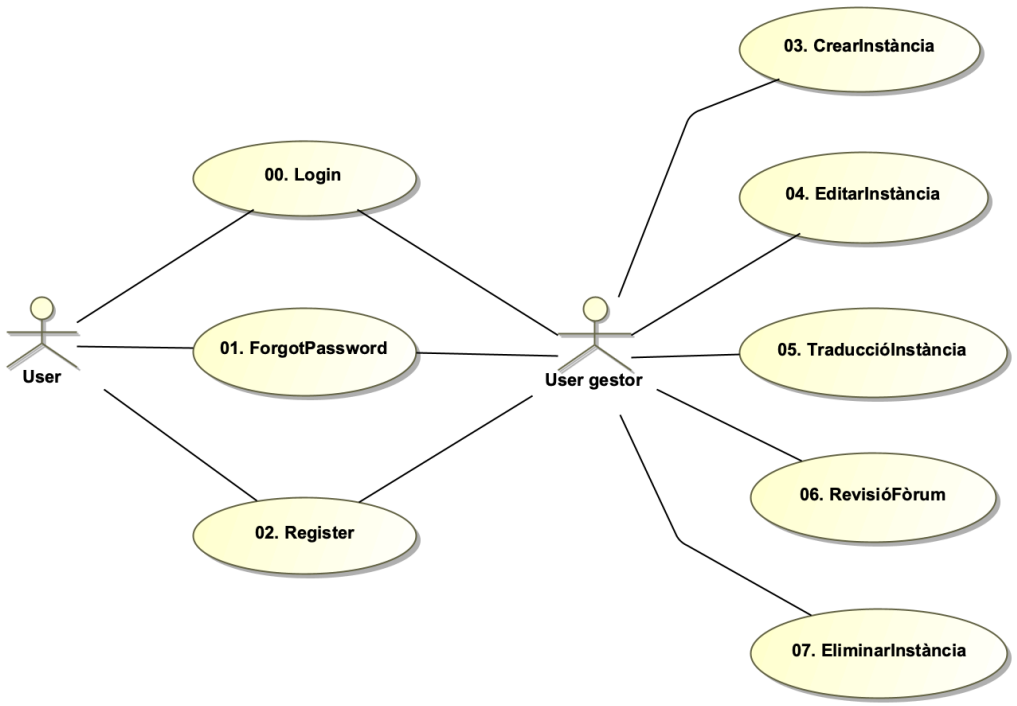


**Figura 40.** Diagrama de casos d'ús de l'aplicació

## 3.2 Requisits no funcionals

Referent als requisits no funcionals s'ha decidit dividir aquests en dues categories: requisits no funcionals del producte i del procés. En la primera s'indiquen algunes puntualitzacions dels principals trets (usabilitat, portabilitat, rendiment i eficiència) i en la segona categoria una breu descripció dels requisits addicionals que no tenen a veure amb el producte final.

### *3.2.1 Requisits no funcionals del producte*

• Usabilitat: L'aplicació serà utilitzada per tot tipus d'usuaris de forma que ha de ser intuïtiva i senzilla d'utilitzar.

• Portabilitat: Ha de ser una aplicació totalment compatibles a tot tipus de dispositius, tant mòbils com tauletes i ordinadors, sense que hi hagi cap problema ni variació entre aquestes dispositius.

• Rendiment i eficiència: S'espera un rendiment i eficiència òptim per a garantir a l'usuari final la millor de les experiències durant l´ús de l'aplicatiu web.

### *3.2.2 Requisits no funcionals del procés*

El llenguatge de programació principal ha de ser PHP fent ús del framework de Laravel, ja que és el llenguatge amb el que es treballa majoritàriament a l'empresa de forma estàndard per a la codificació de totes les aplicacions web.

El codi ha d'estar comentat i acompanyat d'un manual d'ús per als usuaris gestors de continguts, de forma que en puguin consultar en cas d'algun dubte.

# 4 Arquitectura i disseny

## 4.1 Arquitectura i organització de fitxers de l'aplicació

L'aplicació web ha estat implementat amb el framework Laravel, basat en PHP. Laravel proporciona una estructura clara i ben definida que facilita el desenvolupament d'aplicacions web mantenibles i escalables. La separació de lògica en models, vistes i controladors ajuda a mantenir el codi net i organitzat. L'ús de serveis, middleware i altres components integrats augmenta la flexibilitat i la potència del framework.

Laravel és un framework PHP molt popular que segueix el patró Model-View-Controller (MVC) [9]. Això facilita la separació de les preocupacions, millorant la mantenibilitat i escalabilitat de l'aplicació. A continuació, s'explica l'arquitectura i l'organització de fitxers d'una aplicació Laravel.

### *4.1.1 Estructura Bàsica de Directoris*

- app/:
  - Conté la lògica de l'aplicació.
- Subdirectoris més importants:
  - Console/: Comandes de consola definides per l'usuari.
  - Exceptions/: Gestió d'excepcions.
  - Http/:
    - Controllers/: Controladors que gestionen les peticions HTTP.
    - Middleware/: Middleware per filtrar les peticions HTTP.
    - Requests/: Classes per gestionar les peticions HTTP validades.
  - Models/: Models Eloquent que representen les taules de la base de dades.
  - Policies/: Classes per autoritzar accions d'usuari.
  - Providers/: Proveïdors de serveis que inicialitzen diferents components de l'aplicació.

- bootstrap/:
  - Conté el fitxer app.php que inicialitza el framework.
  - cache/: Fitxers de cache generats per optimitzar l'inici de l'aplicació.
- config/:
  - Conté fitxers de configuració per diferents components i serveis de l'aplicació.
- database/:
  - Conté migracions, fàbriques i seeder per la base de dades.
  - migrations/: Fitxers que defineixen l'estructura de la base de dades.
  - factories/: Definicions de fàbriques per generar models de prova.
  - seeders/: Fitxers per inserir dades inicials a la base de dades.
- public/:
  - Conté l'entrada pública de l'aplicació (index.php).
  - Fitxers accessibles públicament com CSS, JavaScript, i imatges.
- resources/:
  - Conté recursos de l'aplicació com vistes i fitxers de llenguatge.
  - views/: Vistes Blade que renderitzen el HTML.
  - lang/: Fitxers de traducció.
  - sass/: Fitxers SASS per estilitzar l'aplicació.
  - js/: Fitxers JavaScript per la part del client.

- routes/:
  - Conté fitxers de definició de rutes.
  - web.php: Rutes per l'aplicació web.
  - api.php: Rutes per l'API.
  - console.php: Rutes per comandes Artisan.
  - channels.php: Canals de broadcast.
- storage/:
  - Conté fitxers generats per l'aplicació.
  - app/: Fitxers de l'aplicació.
  - framework/: Fitxers de framework com sessions i vistes en cache.
  - logs/: Fitxers de log.
- tests/:
  - Conté tests de l'aplicació.
  - Feature/: Tests de funcionalitats completes.
  - Unit/: Tests unitaris per classes i mètodes individuals.
- vendor/:
  - Conté dependències de Composer.

### *4.1.2 Flux de Treball*

- Petició HTTP:
  - Una petició entra a través del fitxer public/index.php.
  - La petició és gestionada pel middleware definit en app/Http/Middleware.
- Rutes i Controladors:
  - Les rutes definides en routes/web.php o routes/api.php determinen quin controlador gestiona la petició.
  - Els controladors es troben en app/Http/Controllers.
- Models i Base de Dades:
  - Els controladors utilitzen models (a app/Models) per interactuar amb la base de dades.
  - Els models utilitzen Eloquent ORM per facilitar aquesta interacció.
- Vistes:
  - Els controladors poden retornar vistes (a resources/views), que són renderitzades amb Blade, el motor de plantilles de Laravel.

Aquí podem veure com quedaria aquesta estructura de fitxers.

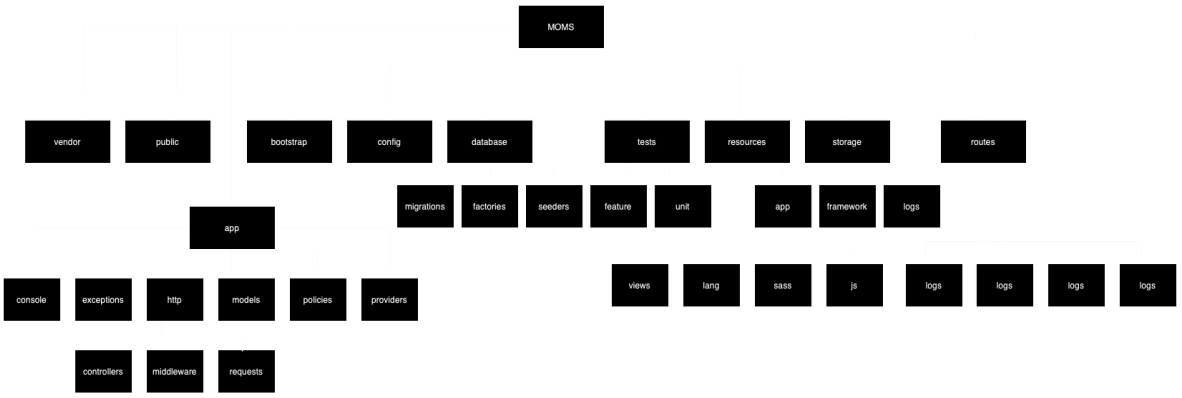


**Figura 41.** Esquema de l'estructura de fitxers de l'aplicació

A continuació es defineixen, a grans trets, els fitxers i directoris que inclou l'aplicació per tal de poder complir els requisits funcionals i no funcionals definits en la secció de Requisits.

Posteriorment es comenta amb més detall en la secció d'implementació algunes de les seves característiques.

Controllers: fitxers que contenen la lògica de les funcions per gestionar les instàncies de l'aplicació web.

- CommentController: conté la lògica per gestionar els comentaris del fòrum.
- ForumController: conté la lògica per gestionar els posts del fòrum.
- NewController: conté la lògica per gestionar les notícies de la web.
- NewsletterController: conté la lògica per gestionar la newsletter de la web.
- PartnerController: conté la lògica per gestionar els partners de la web.
- ProfileController: conté la lògica per gestionar les dades de l'usuari de la web.
- ResearchBookController: conté la lògica per gestionar el research book de la web.
- SelfController: conté la lògica per gestionar el qüestionari de self assessment de la web.
- TranslationController: conté la lògica per gestionar les traduccions de cada text de la web.
- VideoController: conté la lògica per gestionar les informative pills de la web.

Middleware: fitxers que gestionen la lògica que s'executa en fer crides a les rutes de l'API.

- Authenticate: envia l'usuari a la ruta definida en cas de que no estigui autenticat.
- RedirectIfAuthenticate: envia l'usuari a la ruta definida en cas de que estigui autenticat correctament.
- SetLocale: defineix a la path de la ruta l'idioma per gestionar les traduccions de la web.

Models: són els fitxers que defineixen el Model tal com el seu nom indica de les classes de cada instància per a la base de dades, definint quines columnes tindrà cada taula i lògica entre aquestes taules.

- Comments: conté l'estructura de columnes de la taula comments del fòrum.
- News: conté l'estructura de columnes de la taula news.
- Newsletter: conté l'estructura de columnes de la taula newsletter.
- Partner: conté l'estructura de columnes de la taula partner.
- Post: conté l'estructura de columnes de la taula post.
- ResearchBook: conté l'estructura de columnes de la taula research book.
- SelfAssessment: conté l'estructura de columnes de la taula self assessment.
- SelfQuestions: conté l'estructura de columnes de la taula self questions.
- Tag: conté l'estructura de columnes de la taula tag.
- Translations: conté l'estructura de columnes de la taula translations.
- User: conté l'estructura de columnes de la taula user.
- Video: conté l'estructura de columnes de la taula video.

Config: són els fitxers on s'especifica quina configuració s'ha de fer servir per cada apartat de l'aplicatiu web.

- Database: conté la configuració de quina base de dades ha d'utilitzar l'aplicació i les dades per autenticar-se a aquesta.
- Cache: conté la configuració que ha de fer servir l'aplicació web per al caché.

Database: fitxers que gestionen la base de dades, per crear dades de proves, i per crear les taules en si d'aquesta.

- Factories:
  - CommentFactory: conté la lògica de creació de dades de prova per a la taula comment.
  - NewsFactory: conté la lògica de creació de dades de prova per a la taula news.
  - UserFactory: conté la lògica de creació de dades de prova per a la taula user.
- Migrations:
  - Create_partner_table: conté quines columnes i de quin tipus son cadascuna, relacions, primary key de la taula partner.
  - Create_translation_table: conté quines columnes i de quin tipus son cadascuna, relacions, primary key de la taula translation.
  - Create_newsletter_table: conté quines columnes i de quin tipus son cadascuna, relacions, primary key de la taula newsletter.

Resources: són els recursos de l'aplicació, com ara fitxers amb traduccions, les mateixes plantilles que es visualitzaran a l'aplicatiu web, imatges, pdfs...

- Lang:
  - Es: conté les traduccions que s'agafen localment amb castellà.
  - En: conté les traduccions que s'agafen localment amb anglès.
  - It: conté les traduccions que s'agafen localment amb italià.
- Views:
  - Home: conté l'estructura i estils de la pàgina home.
  - Forum: conté l'estructura i estils de la pàgina del fòrum.
  - News: conté l'estructura i estils de la pàgina news.
  - Login: conté l'estructura i estils de la pàgina per iniciar sessió.
  - Register: conté l'estructura i estils de la pàgina per registrar-se.

Routes: són els fitxers que gestionaran les rutes de l'API de l'aplicatiu web.

- Web: gestiona les rutes de l'API de la web i del gestor.
- Auth: gestiona les rutes de l'API [10] relacionades amb l'autenticació, registre, canvi de contrasenyes de l'usuari de la web.

## 4.2 Disseny de la base de dades

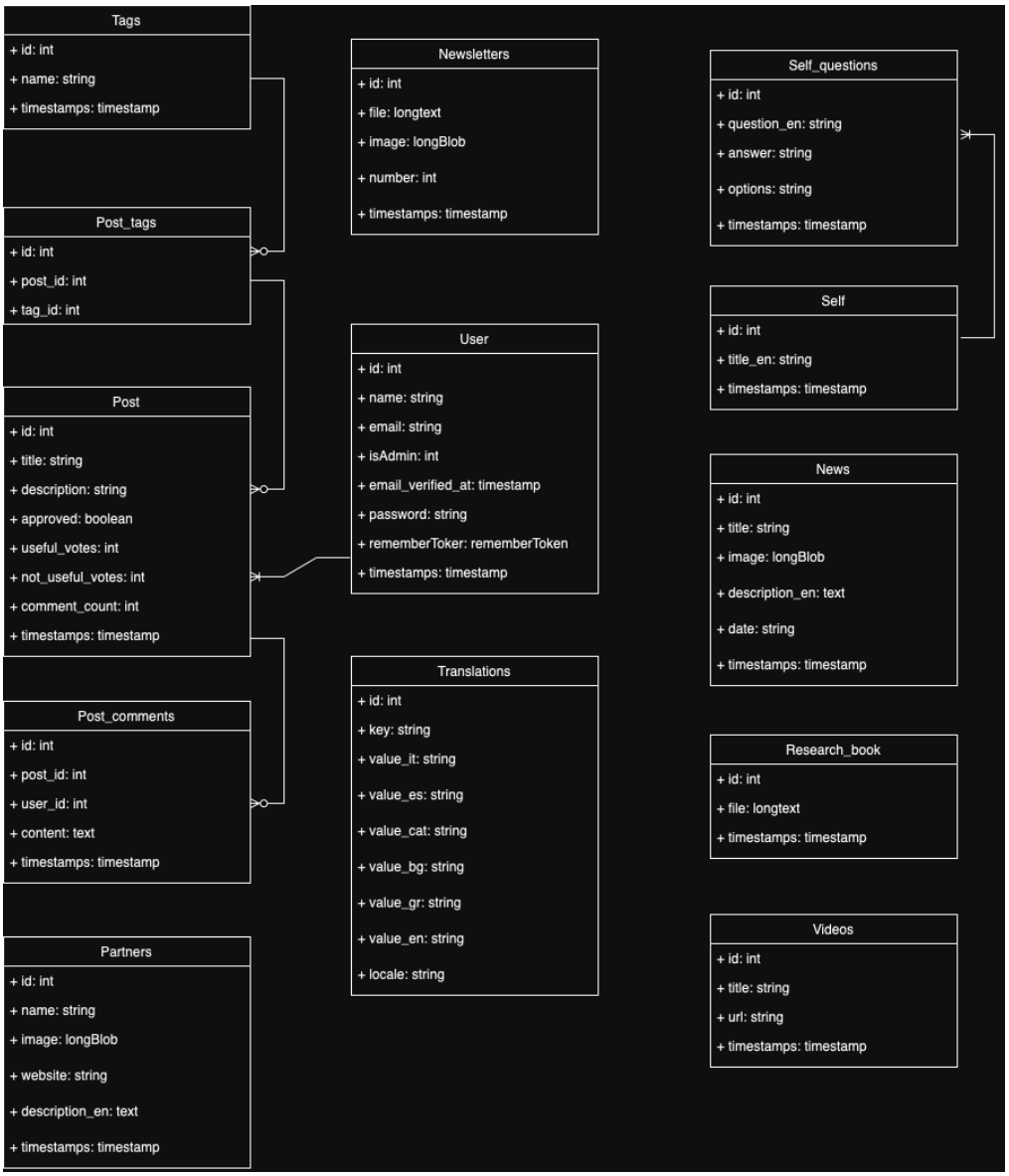


**Figura 42.** Diagrama de classes de la base de dades

Seguidament es detallen les taules, les relacions, i el propòsit dels camps que inclouen.

### *4.2.1 Especificació de les taules*

User: taula que emmagatzema les dades dels usuaris que es registren a la pàgina web.

- Id: identificador únic de la instància dins la taula.
- Name: nom de l'usuari.
- Email: email de l'usuari, ha de ser únic.
- IsAdmin: camp que gestiona l'accés al gestor de la pàgina web.
- Email_verified_at: moment de verificació de l'usuari a la pàgina.
- Password: contrasenya emmagatzemada amb un hash a la base de dades.
- RememberToken: camp per emmagatzemar si l'usuari vol que sigui recordat per no tenir que iniciar sessió cada cop que accedeixi a la pàgina.
- Timestampts: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Post: taula que emmagatzema els posts del fòrum.

- Id: identificador únic de la instància dins la taula.
- Title: títol del post del fòrum.
- Description: contingut del post.
- Approved: camp que determina si el post està aprovat es mostrarà a la pàgina o restarà a l'espera fins ser-ho.
- Useful_votes: número de vots donats per diferent usuaris marcant com útil el post.
- Not_useful_votes: número de vots donats per diferent usuaris marcant com no útil el post.
- Comment_count: número de comentaris que té el post
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Post_comments: taula que relaciona els comentaris amb els posts del fòrum, guardant el posts al que pertanyen i quin usuari l'ha fet.

- Id: identificador únic de la instància dins la taula.
- Post_id: id del post on s'ha posat aquest comentari.
- User_id: id del usuari que ha posat el comentari.
- Content: contingut del comentari.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Translations: taula que conté les traduccions les textos de l'aplicatiu web.

- Id: identificador únic de la instància dins la taula.
- Key: clau identificativa per a mostrar el valor dels textos amb els diferents idiomes disponibles.
- Value_it: text traduït al italià.
- Value_es: text traduït al castellà.
- Value_cat: text traduït al català.
- Value_bg: text traduït al búlgar.
- Value_gr: text traduït al grec.
- Value_en: text traduït a l'anglès
- Locale: camp que determina el valor del idioma que ha de coincidir amb el del path per mostrar la traducció adient a l'idioma seleccionat.

Tags: taula que emmagatzema els tags disponibles per utilitzar als posts del fòrum.

- Id: identificador únic de la instància dins la taula.
- Name: nom de l'etiqueta.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Post_tags: taula que relaciona els tags amb els posts del fòrum, guardant el posts al que pertanyen.

- Id: identificador únic de la instància dins la taula.
- Post_id: id del post on està assignada aquesta etiqueta.
- Tag_id: id de l'etiqueta assignada al post.

Videos: s'emmagatzema els vídeos que es mostraran posteriorment a l'apartat d'informative pills.

- Id: identificador únic de la instància dins la taula.
- Title: títol del vídeo per ser mostrat a la pàgina.
- Url: url del vídeo que serà mostrat a la pàgina.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Self: taula que emmagatzema els formularis per al self assessment.

- Id: identificador únic de la instància dins la taula.
- Title_en: títol del formulari amb anglès.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Self_questions: taula que emmagatzema les preguntes per als formularis del self assessment.

- Id: identificador únic de la instància dins la taula.
- Question_en: títol de la qüestió anglès.
- Answer: resposta de la qüestió.
- Options: opcions per a la qüestió.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

News: taula que guarda les notícies que es mostraran a l'apartat de news de la pàgina.

- Id: identificador únic de la instància dins la taula.
- Title: títol de la noticia.
- Image: imatge per mostrar a la noticia.
- Description_en: descripció de la notícia amb anglès.
- Date: data en que va ocórrer aquest event anunciat a la noticia.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Partners: taula que emmagatzemarà les entitats col·laboradores del projecte per mostrar-les a l'apartat de partners de la web.

- Id: identificador únic de la instància dins la taula.
- Name: nom de l'entitat col·laboradora.
- Image: logo de l'entitat col·laboradora.
- Website: lloc web de l'entitat col·laboradora.
- Description_en: descripció amb anglès de l'entitat col·laboradora.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Newsletters: conté els fitxers de newsletter que es creen per al projecte i es mostraran al apartat de news.

- Id: identificador únic de la instància dins la taula.
- File: fitxer amb el contingut de la newsletter.
- Image: imatge de portada per a la newsletter.
- Number: número comptador de newsletter.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

Research_book: taula que conté el llibre informatiu sobre el projecte.

- Id: identificador únic de la instància dins la taula.
- File: fitxer amb el contingut del research book.
- Timestamps: camps per emmagatzemar el moment de creació i el moment d'última actualització.

# 5 Implementació

En aquest apartat s'esmenten els detalls més destacables de la implementació.

## 5.1 Enrutament de l'aplicació web

### 5.1.1 *Com funcionen les rutes i API Calls*

Mecanisme de Rutes

Les rutes GET i POST serveixen per mostrar formularis i processar dades, respectivament. En el cas de les rutes de registre i inici de sessió, la ruta GET mostra un formulari, i la ruta POST envia les dades per ser processades.

Controladors

Cada ruta apunta a un mètode específic d'un controlador que gestiona la lògica associada. Els controladors processen les sol·licituds i, sovint, redirigeixen l'usuari o tornen una resposta basada en l'acció realitzada (com l'inici de sessió, el registre o el restabliment de contrasenya).

Rutes d'Accés Públic

Algunes rutes son accessibles sense autenticar-se, com les pàgines d'inici, sobre, i altres pàgines informatives. Aquestes rutes poden carregar vistes o dirigir l'usuari a altres pàgines de l'aplicació.

Rutes de CRUD

Utilitzen el mètode resource per generar automàticament rutes per a operacions de CRUD. Laravel crea les rutes per mostrar, crear, editar i eliminar recursos segons les necessitats del controlador associat.

Rutes del Dashboard

Organitzades sota un prefix de dashboard per gestionar diferents aspectes de l'administració dins de l'aplicació. Aquestes rutes utilitzen el middleware auth per garantir que només els usuaris amb permisos adequats puguin accedir a aquestes funcionalitats.

### *5.1.2 Rutes de l'aplicatiu*

Pel que fa a l'enrutament de l'aplicatiu tenim diverses configuracions implementades a explicar, ja que hem fet ús del framework Laravel per crear la nostra aplicació web hem configurat dos fitxers diferents.

Auth.php

Fitxer on configurarem les rutes que necessiten d'un middleware per fer verificacions on necessitem que hi hagi un usuari registrat o amb la sessió iniciada.

Rutes amb el Middleware guest

Aquest middleware s'assegura que les rutes només siguin accessibles per als usuaris no autenticats. Si un usuari autenticat intenta accedir a aquestes rutes, serà redirigit a una pàgina diferent.

Registre i Inici de Sessió

- GET /{lang?}/register i POST /register: Aquesta ruta permet als usuaris registrar-se en l'aplicació. La vista create del controlador RegisteredUserController mostra el formulari de registre, i el mètode store processa la sol·licitud de registre.
- GET /{lang?}/login i POST /login: Aquesta ruta permet als usuaris iniciar sessió. La vista create del controlador AuthenticatedSessionController mostra el formulari d'inici de sessió, mentre que el mètode store autentica l'usuari.

Recuperació de Contrasenya

GET /{lang?}/forgot-password i POST /forgot-password: Aquestes rutes permeten als usuaris sol·licitar la recuperació de la contrasenya. La vista create del controlador PasswordResetLinkController mostra el formulari per sol·licitar un enllaç de restabliment de contrasenya, i el mètode store envia l'enllaç de restabliment.

Restabliment de Contrasenya

GET /{lang?}/reset-password/{token} i POST /reset-password: Aquestes rutes permeten als usuaris restablir la contrasenya. La vista create del controlador NewPasswordController mostra el formulari de restabliment, i el mètode store processa el nou valor de la contrasenya.

Rutes amb el Middleware auth

Aquest middleware assegura que les rutes només siguin accessibles per a usuaris autenticats. Si un usuari no autenticat intenta accedir a aquestes rutes, serà redirigit.

Verificació d'Email

- GET /{lang?}/verify-email: Aquesta ruta mostra la vista per a la verificació d'email. Utilitza el controlador EmailVerificationPromptController.
- GET /verify-email/{id}/{hash}: Aquesta ruta verifica l'email de l'usuari. Utilitza el controlador VerifyEmailController, amb els middleware signed (per verificar l'enllaç de verificació) i throttle (per limitar la freqüència de les sol·licituds).

Notificacions de Verificació d'Email

POST /email/verification-notification: Aquesta ruta permet als usuaris sol·licitar una nova notificació de verificació d'email. Utilitza el controlador EmailVerificationNotificationController i el middleware throttle.

Confirmació de Contrasenya

GET /confirm-password i POST /confirm-password: Aquestes rutes permeten als usuaris confirmar la seva contrasenya abans de realitzar accions sensibles. Utilitzen el controlador ConfirmablePasswordController.

Actualització de Contrasenya

PUT /password: Aquesta ruta permet als usuaris actualitzar la seva contrasenya. Utilitza el controlador PasswordController.

Tancament de Sessió

POST /logout: Aquesta ruta permet als usuaris tancar la sessió. Utilitza el mètode destroy del controlador AuthenticatedSessionController.

Aquí podem veure com funciona la integració dels middlewares a les rutes per al tractament dels usuaris, determinant si han iniciat sessió o no.

```php
Route::middleware('guest')->group(function () {
    Route::get('/{lang?}/register', [RegisteredUserController::class, 'create'])
                ->name('register');

    Route::post('/register', [RegisteredUserController::class, 'store']);

    Route::get('/{lang?}/login', [AuthenticatedSessionController::class, 'create'])
                ->name('login');

    Route::post('/login', [AuthenticatedSessionController::class, 'store']);

    Route::get('/{lang?}/forgot-password', [PasswordResetLinkController::class, 'create'])
                ->name('password.request');

    Route::post('/forgot-password', [PasswordResetLinkController::class, 'store'])
                ->name('password.email');

    Route::get('/{lang?}/reset-password/{token}', [NewPasswordController::class, 'create'])
                ->name('password.reset');

    Route::post('/reset-password', [NewPasswordController::class, 'store'])
                ->name('password.store');
});
```

**Figura 43.** Rutes d'usuaris no autenticats

```php
Route::middleware('auth')->group(function () {

    Route::get('/{lang?}/verify-email', EmailVerificationPromptController::class)->name('verification.notice');

    Route::get('/verify-email/{id}/{hash}', VerifyEmailController::class)
                ->middleware(['signed', 'throttle:6,1'])
                ->name('verification.verify');

    Route::post('/email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
                ->middleware('throttle:6,1')
                ->name('verification.send');

    Route::get('/confirm-password', [ConfirmablePasswordController::class, 'show'])
                ->name('password.confirm');

    Route::post('/confirm-password', [ConfirmablePasswordController::class, 'store']);

    Route::put('/password', [PasswordController::class, 'update'])->name('password.update');

    Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])
                ->name('logout');
});
```

**Figura 44.** Rutes d'usuaris autenticats

Per exemple podem veure com verifica si el primer cop que ha iniciat sessió ja ha verificat el seu email o no, fins que no el verifiqui no podrà actuar com a un usuari verificat i realitzar les tasques disponibles, com ara interactuar al fòrum.

Web.php:

Fitxer on configurarem les diverses rutes que ens portaran a les diferents pàgines amb les seves respectives configuracions o controllers que necessitin utilitzar.

Pàgines Estàtiques

- /{lang?}/home: Mostra la vista de la pàgina d'inici.
- /{lang?}/informative-pills: Mostra la vista de les píndoles informatives.

Rutes de Recursos i Controladors

Aquestes rutes estan associades a controladors específics que gestionen les operacions CRUD per diversos recursos. Utilitzant el mètode resource, Laravel crea automàticament les rutes necessàries per a operacions com la visualització, creació, edició i eliminació de recursos.

Exemples de com usar controladors a l'arxiu d'enrutament:

- Route::resource('partners', PartnerController::class);
- Route::resource('forum', ForumController::class);
- Route::resource('tags', TagController::class);
- Route::resource('pills', VideoController::class);
- Route::resource('new', NewController::class);
- Route::resource('self', SelfController::class);

Rutes Protegides per Autenticació

Algunes rutes requereixen que l'usuari estigui autenticat per accedir-hi. Aquestes rutes utilitzen el middleware auth per protegir-se:

- **Perfil d'Usuari**

GET /{lang?}/profile: Mostra el formulari per editar el perfil de l'usuari.

PATCH /profile: Actualitza les dades del perfil de l'usuari.

DELETE /profile: Elimina el perfil de l'usuari.

- **Dashboard**

/{lang?}/dashboard: Mostra el tauler de control. Accessible només per usuaris autenticats i amb verificació d'email.

- **Rutes de CRUD per Recursos del Dashboard**

Les rutes del dashboard permeten gestionar recursos específics amb operacions CRUD. Aquestes rutes utilitzen el prefix {lang?}/dashboard per separar les funcionalitats d'administració de les pàgines públiques.

Agrupació de rutes

Un altra de les coses a tenir en compte amb les rutes i integracions es l'ús dels controlador per a cridar funcions d'un controlador al fer una crida d'una ruta de l'API de l'aplicació.

```php
Route::resource('partners', PartnerController::class);
Route::resource('forum', ForumController::class);
Route::resource('tags', TagController::class);
Route::resource('pills', VideoController::class);
Route::resource('new', NewController::class);
Route::resource('self', SelfController::class);
```

**Figura 45.** Addició de controllers per referència

```php
Route::group(['prefix' => '{lang?}/dashboard'], function () {
    //PARTNERS
    Route::get('partners', [PartnerController::class, 'index'])->name('dashboard.partners.index');
    Route::get('partners/create', [PartnerController::class, 'create'])->name('dashboard.partners.create');
    Route::post('partners', [PartnerController::class, 'store'])->name('dashboard.partners.store');
    Route::get('partners/{partner}/edit', [PartnerController::class, 'edit'])->name('dashboard.partners.edit');
    Route::patch('partners/{partner}/edit', [PartnerController::class, 'update'])->name('dashboard.partners.update');
    Route::delete('partners/{partner}', [PartnerController::class, 'destroy'])->name('dashboard.partners.destroy');
```

**Figura 46.** Creació del grup de rutes amb el prefix d'idioma

Ara un cop tenim disponibles els controlador mitjançant les rutes podem fer les crides a aquestes funcions. Podem veure com declarem un "grup" de rutes que tindran totes el mateix prefix ja que seran totes les rutes amb les que treballaràn els usuaris del gestor. Aquí podem veure la part de la gestió de les entitats col·laborades del projecte, com fem les crides per llistar aquestes, crear-ne de noves, editar-les, eliminar-les.

## 5.2 Connexió a la base de dades

Pel que fa a la configuració i connexió de la base de dades i d'altres serveis com el del mailing hem creat un fitxer .env comunment utilitzat en el desenvolupament web per emmagatzemar les claus i contrasenyes de forma segura, aquest fitxer ens permet cridar les claus necessàries amb el nom de les variables i que no siguin accessibles desde cap lloc del nostre aplicatiu web, també l'hem inclós al fitxer .gitignore de forma que ni tenint accés al nostre repositori poden extraure aquestes claus.

Pel que fa a la configuració necessària per a dur a terme la connexió amb la base de dades requerim de 6 paràmetres diferents:

- DB_CONNECTION: quin tipus de base de dades utilitzarem, en aquest cas es una MySQL.
- DB_HOST: on està emmagatzemada la base de dades, per fer proves disposem d'una versió local, finalment l'aplicatiu està al núvol,
- DB_PORT: el port pel qual s'accedeix al recurs de la base de dades, per a les bases de dades sql acostuma a ser el 3306, llevat que vulguem definir nosaltres un port específic i realitzem tota la configuració necessària.
- DB_DATABASE: el nom de la base de dades que utilitzarem.
- DB_USERNAME: el nom de l'usuari que es connectarà a la base de dades.
- DB_PASSWORD: la contrasenya de l'usuari que es connectarà a la base de dades.

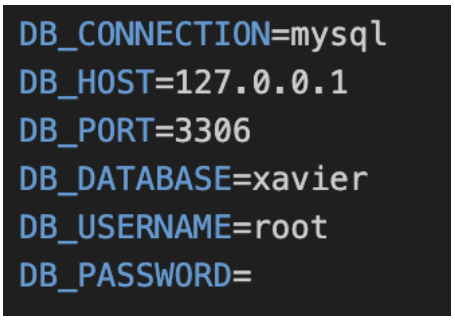


**Figura 47.** Variables per connectar a la base de dades

Laravel disposa d'un fitxer anomenat "database.php" on definirem quins paràmetres ha d'utilitzar del fitxer .env i realitza la connexió sense necessitat de codificar-la.

## 5.3 Configuració d'idiomes

Per a gestionar amb quin idioma han de sortir els textos hem optat per dedicar una part de l'url de la pàgina per a especificar-ho, de forma que al canviar d'idioma fent ús del selector d'idiomes de la pròpia pàgina canviarà aquest part de l'url.

Per a gestionar les traduccions ho hem fet des de el gestor intern al que tenen accés els usuaris administrador de l'aplicatiu web, on s'han creat tots els textos necessaris amb un nom identificatiu i traduint els textos amb tots els idiomes disponibles a la pàgina web.

Pel que fa a la configuració del codi inclourem aquest codi a cada pàgina on mostrarem textos en diversos idiomes:

```
@php
    $locale = app()->getLocale();
    $translations  = \App\Models\Translations::where('locale', $locale)->get()->select('key', 'value')->pluck('value', 'key')->toArray();
    $translations_en  = \App\Models\Translations::where('locale', "en")->get()->select('key', 'value')->pluck('value', 'key')->toArray();
@endphp
```

**Figura 48.** Lectura de variables per mostrar traduccions

I els textos els llegirem de la base de dades fent ús del seu nom assignat desde el gestor de traduccions:

```
<div class="about">
    <div class="about_pic">
        <h2 class="about_title">{!!strip_tags($translations['about_title'] ?? $translations_en['about_title'])!!}</h2>
        <img class="image-circle" src="/images/Circle.svg" alt="Circle image">
    </div>
    <p class="about_text">{!! strip_tags($translations['about_text'] ?? $translations_en['about_text']) !!}</p>
</div>
```

**Figura 49.** Mostreig de variable per nom i idioma

Afegim la funció de strip_tags ja que s'està fent ús de l'editor de textos TinyMCE i s'emmagatzemen les tags del text usades, com ara <p> o <h1>...

També pel que fa a alguns textos s'ha fet ús d'una forma local de mostrar les traduccions, fent ús també de l'url i creant una carpeta anomenada "lang" dins la carpeta resources, amb Laravel es detecta automàticament de forma que afegint aquest fragment de codi mostra el text corresponent.

Farem ús en casos de textos estàtics que no requereixin canvis i que volguéssim carregar de forma més ràpida, com ara els títols del menú, però també s'ha d'anar en compte perquè si fem un ús excessiu d'aquest mètode de càrrega local de traduccions pot derivar en un temps de càrrega superior de pàgina que si es carreguen textos des de la base de dades.

```
<h4 class="top">
    <i class="bi bi-caret-right-fill"></i>
    <i class="bi bi-caret-right-fill"></i>
    <i class="bi bi-caret-right-fill"></i>
    {{ __('tl_title') }}
    <i class="bi bi-caret-left-fill"></i>
    <i class="bi bi-caret-left-fill"></i>
    <i class="bi bi-caret-left-fill"></i>
```

**Figura 50.** Mostreig de variable amb traduccions locals

Lectura de variables locals per obtenir els valors de les traduccions adients al text.

## 5.4 Gestió de les dades

Pel que fa a la gestió de les dades es fa mitjançant el gestor de la pròpia pàgina on té assignades diferents pàgines per gestionar les diverses entitats de l'aplicació web. Aquesta gestió es fa mitjançant els controllers, on gestionarem tota la part lògica per manipular les instàncies d'aquestes entitats.

Posteriorment s'explica les funcionalitats del controller més complet, des d'on podem partir com a base i agafar les principals funcions comuns a totes les entitats com son index, crear, store, editar, update i eliminar, que com els seus noms indiquen conformen el comunment anomenat CRUD de les aplicacions.

Els controllers es declaren com a classes i estenen una classe controller per a que proporciona la lògica de maneig d'entitats durant les crides de les API, permetent agafar i passar paràmetres mitjançant aquestes.

### *5.4.1 Visualització de dades*

Fem crida d'aquesta funció on utilitzem la funció de login del backend implementat, en cas d'haver algun error mirem es mostrarà per pantalla que les dades introduïdes no son correctes. També gestiona la ruta de redirecció en cas de que tingui permisos d'administrador o no, el que permetrà que accedeixi al gestor del lloc web.

```php
/**
 * Handle an incoming authentication request.
 */
public function store(LoginRequest $request): RedirectResponse
{
    $request->authenticate();

    $request->session()->regenerate();

    if (Auth::user()->isAdmin == 0) {
        return redirect(RouteServiceProvider::GUEST);
    }else{
        return redirect(RouteServiceProvider::HOME);
    }
}
```

**Figura 51.** Codi per iniciar sessió a l'aplicació

```php
/**
 * Attempt to authenticate the request's credentials.
 *
 * @throws \Illuminate\Validation\ValidationException
 */
public function authenticate(): void
{
    $this->ensureIsNotRateLimited();

    if (! Auth::attempt($this->only('email', 'password'), $this->boolean('remember'))) {
        RateLimiter::hit($this->throttleKey());

        throw ValidationException::withMessages([
            'email' => trans('auth.failed'),
        ]);
    }

    RateLimiter::clear($this->throttleKey());
}
```

**Figura 52.** Codi per iniciar sessió a l'aplicació

```php
class ForumController extends Controller
{
    public function index(Request $request)
    {
        $tagFilter = $request->input('tag');

        if ($tagFilter) {
            $posts = Post::whereHas('tags', function ($query) use ($tagFilter) {
                $query->where('id', $tagFilter);
            })->get();
        }
        else {
            $posts = Post::all();
        }

        $availableTags = Tag::all();

        return view('dashboard.forum.posts-list', [
            'posts' => $posts,
            'availableTags' => $availableTags,
            'selectedTag' => $tagFilter
        ]);
    }
```

**Figura 53.** Codi per obtenir dades sobre una taula específica

En aquest cas de d'obtenció de dades per visualitzar-les sobre part del gestor de la pàgina web podem veure com apart ofereix l'opció de filtrar per les Tags en cas d'haver-n'hi seleccionades, fa una query a la base de dades amb els paràmetres que s'han passat i posteriorment retorna la vista on es mostraràn aquestes dades, on es passa com a paràmetres en forma d'array el resultat de la query.

```
@if ($posts->isEmpty())
    <p>{{ __('No posts available.') }}</p>
@else
    @foreach ($posts as $post)
        <div class="card mb-3">
            <div class="card-header">
                <h5 class="card-title">{{ $post->title }}</h5>
            </div>
            <div class="card-body">
                <p class="card-text mb-4">{{ $post->description }}</p>
                <div class="d-flex justify-content-between mb-0">
                    <p class="card-text mb-0">
                        @foreach ($post->tags as $tag)
                            <span class="badge">{{ $tag->name }}</span>
                        @endforeach
                    </p>
                    <p class="card-text">{{ $post->user->name }}</p>
                </div>
```

**Figura 54.** Codi de comprovació de valors d'una variable

Al codi de la vista podem veure com primerament comprovem els valors que conté l'array, si esta buit mostrem que no hi ha contingut disponible, i en cas de que si hi hagi recorrem les instàncies de l'array mostrant les dades que hem definit com rellevants.

```
public function showApproved(Request $request)
{
    $query = Post::where('approved', 2)->with('comments.user')->orderByDesc('id');

    if ($request->has('tag')) {
        $tagId = $request->tag;
        $query->whereHas('tags', function ($query) use ($tagId) {
            $query->where('id', $tagId);
        });
    }

    $posts = $query->get();
    $availableTags = Tag::all();

    return view('forum', [
        'posts' => $posts,
        'availableTags' => $availableTags,
        'selectedTag' => $request->tag ?? null,
    ]);
}
```

**Figura 55.** Codi per fer una query amb filtres

Una de les funcions específiques d'aquest controller es que és necessària l'aprovació d'un post per ser mostrat al fòrum, de forma que executem una query filtrant pel camp approved i passem el contingut de la mateixa forma que al gestor per a poder ser visualitzat pels usuaris que accedeixin al fòrum.

```php
public function create()
{
    $availableTags = Tag::all();
    return view('dashboard.forum.post-create', compact('availableTags'));
}
```

**Figura 56.** Codi per mostrar una vista de creació d'instàncies

Pel que fa a la funció de crear noves entrades passarem com a paràmetre les tags que poden ser utilitzades per a donar informació extra al post del fòrum i retornarem la vista on es mostrarà el formulari per crear-ne un de nou.

```php
public function store(Request $request)
{
    $request->validate([
        'title' => 'required',
        'description' => 'required',
        'tags' => 'required|array'
    ]);

    $post = Post::create([
                'user_id' => auth()->id(),
                'title' => $request->title,
                'description' => strip_tags($request->description),
                'approved' => false,
            ]);

    $tags = [];
    foreach ($request->tags as $tagName) {
        $tag = Tag::firstOrCreate(['name' => $tagName]);
        $tags[] = $tag->id;
    }

    $post->tags()->sync($tags);

    return redirect()->route('dashboard.forum.index', ['lang' => app()->getLocale()])->with('status', 'Post created successfully. Waiting for approval');
}
```

**Figura 57.** Codi per emmagatzemar una nova instància a la base de dades

Un cop fem submit al formulari mostrat per l'anterior funció farem la crida a la funció store, que rep totes les dades del formulari i executa una query d'inserció de dades a la base de dades, un cop feta la inserció ens redigirà a la pàgina del gestor on es visualitzarà aquest nou post.

```php
public function reviewPost($lang, Post $post){
    return view('dashboard.forum.post-review', compact('post'));
}
```

**Figura 58.** Codi per mostrar la vista de revisió d'un post

Aquest mètode ens retornarà una pantalla on veure el contingut del post que volem revisar amb dos botons que ens donen l'opció d'aprovar o denegar la publicació final del post dins al fòrum.

```php
public function approveOrDeny(Request $request, $lang, Post $post)
{
    $request->validate([
        'action' => 'required|in:approve,deny',
    ]);

    $updateData = [];

    if ($request->action === 'approve') {
        $updateData['approved'] = 2;
    } elseif ($request->action === 'deny') {
        redirect()->route('dashboard.forum.index', ['lang' => $lang])
                  ->with('status', 'Post denied');
    }

    if (!empty($updateData)) {
        DB::table('posts')
            ->where('id', $post->id)
            ->update($updateData);
    }

    return redirect()->route('dashboard.forum.index', ['lang' => $lang])
                     ->with('status', 'Post updated successfully');
}
```

**Figura 59.** Codi per modificar l'estat d'un post a acceptat o denegat

Aquesta funció serà executada al fer clic a qualsevol dels dos botons de l'anterior vista, on modificarà el valor de si està aprovat per ser visible o no i finalment ens retornarà a la pantalla de gestió de posts del fòrum.

```php
public function edit($lang, Post $post)
{
    $availableTags = Tag::all();
    return view('dashboard.forum.post-edit', compact('post', 'availableTags'));
}
```

**Figura 60.** Codi per mostrar la vista d'edició d'una instància

Aquesta funció ens portarà a una vista on es carregarà el formulari d'edició del post del fòrum seleccionat amb les dades d'aquest i les tags disponibles a la base de dades.

```php
public function update (Request $request, $lang, Post $post){
    $request->validate([
        'title' => 'required',
        'description' => 'required',
        'tags' => 'required|array'
    ]);

    $post->update([
        'title' => $request->title,
        'description' => strip_tags($request->description),
    ]);

    $tags = [];
    foreach ($request->tags as $tagName) {
        $tag = Tag::firstOrCreate(['name' => $tagName]);
        $tags[] = $tag->id;
    }

    $post->tags()->sync($tags);

    return redirect()->route('dashboard.forum.index', ['lang' => app()->getLocale()])->with('status', 'Post updated successfully');
}
```

**Figura 61.** Codi per actualitzar les dades d'una instància

Un cop es fa clic al botó de guardar edició es crida aquesta funció que actualitzarà tots els valors nous, si la tag no existeix a la base de dades la crearà i finalment ens retornarà a la vista que conté la llista amb tots els posts del fòrum.

```php
public function delete($lang, Post $post, Comment $comments){
    $comments->where('post_id', $post->id)->delete();
    $post->delete();
    return redirect()->route('dashboard.forum.index', ['lang' => app()->getLocale()])->with('status', 'Post deleted successfully');
}
```

**Figura 62.** Codi per eliminar una instància específica

Des del llistat de posts del fòrum al fer clic al botó de delete es cridarà aquest mètode amb la id del post a eliminar, s'executarà la query fent ús d'aquest identificador únic i un cop acabada l'execució s'actualitzarà la vista automàticament sense aquest post.

Tenint en compte que dels posts del fòrum pengen els comentaris, l'eliminació d'aquests suposarà l'eliminació en cascada dels comentaris associats.

```php
public function store(Request $request)
{
    $files = [];
    if ($request->has('file')) {
        foreach ($request->file('file') as $file) {
            $filename = $file->getClientOriginalName();
            $file->move(public_path('pdf/Research Book/'), $filename);
            $files[] = 'pdf/Research Book/' . $filename;
        }
    }

    $validatedData['file'] = implode(',', $files);

    $research = new ResearchBook();
    $research->file = $validatedData['file'];
    $research->created_at = now();
    $research->updated_at = now();

    $research->save();

    return redirect()->route('dashboard.research-book.index', ['lang' => app()->getLocale()])
        ->with('status', 'Research Book created successfully.');
}
```

**Figura 63.** Codi per emmagatzemar una instància amb fitxers

Un altre mètode rellevant de l'aplicació sería aquest per emmagatzemar la creació del research book, ja que gestiona la pujada d'un fitxer, on realitzarem comprovacions de tipus per a que sigui un PDF i poder mostrar-lo sense problemes des de la vista.

```php
public function store(Request $request)
{
    // Validate the request data
    $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string|max:500',
        'url' => 'required|url',
    ]);

    // Extract YouTube video ID from the URL
    $url = $request->url;
    $videoId = $this->extractYoutubeVideoId($url);

    if (!$videoId) {
        return redirect()->back()->with('error', 'Invalid YouTube URL.');
    }

    // Create a new video instance
    $video = new Video([
        'title' => $request->title,
        'description' => $request->description,
        'url' => "https://www.youtube.com/embed/$videoId",
    ]);

    // Save the video to the database
    $video->save();

    return redirect()->route('dashboard.pills.index', ['lang' => app()->getLocale()])
                     ->with('status', 'Tag created successfully.');
}
```

**Figura 64.** Codi per emmagatzemar una instància amb crida a mètode externs

També tenim altres funcions rellevant com per emmagatzemar les informative pills, on es demana una url d'un vídeo de youtube per poder mostrar-lo després a la vista.

```php
private function extractYoutubeVideoId($url)
{
    parse_str(parse_url($url, PHP_URL_QUERY), $queryParams);
    return $queryParams['v'] ?? null;
}
```

**Figura 65.** Codi per formatar una url de youtube

Però per poder mostrar-lo necessitarem d'aquesta funció addicional que extreu la ID del video que a l'anterior funció afegirem a la url transformada del vídeo per guardar-la amb l'estructura necessària.

# 6 Joc de proves

Pel que fa al joc de proves com a punt addicional al treball he inclòs tests unitaris a l'aplicatiu web. A continuació detallo la recerca feta per a tenir un punt de partida i saber quins son els tipus de test més adients per al projecte i com dur a terme aquests de forma eficient i que sigui útil per al projecte.

## 6.1 Estudi tipus de jocs de proves

Per a tenir en compte quins tests son els més adients primer tenim que saber quin tipus de tests existeixen i quina funció tenen cadascun. Existeixen els següents:

UNITS TESTS:

Les proves unitàries [10] són un procés que consisteix a provar la unitat funcional de codi més petita. Aquestes proves de programari ajuden a garantir la qualitat del codi i són una part essencial del desenvolupament de programari. Una pràctica recomanada en el desenvolupament de programari és escriure el codi en unitats petites i funcionals, i després crear una prova unitària per a cada unitat de codi. És possible escriure primer les proves unitàries com a codi i, a continuació, executar-les de manera automàtica cada vegada que es facin canvis en el codi del programari. D'aquesta manera, si una prova falla, es pot aïllar ràpidament l'àrea del codi que conté l'error. Les proves unitàries promouen el pensament modular i milloren la cobertura i la qualitat de les proves. A més, les proves unitàries automatitzades permeten que els desenvolupadors disposin de més temps per centrar-se en la programació.

Com a possibles unit tests tindríem aquests exemples:

- Prova d'un Model (User):

Descripció: Comprova que la funció getFullName() del model User retorna correctament el nom complet de l'usuari.

Utilitat: Assegura que les funcions del model treballin correctament amb les dades i retornin els resultats esperats.

- Prova d'un Controlador (UserController):

Descripció: Verifica que la funció index() del controlador UserController retorna una vista amb la llista d'usuaris correctament.

Utilitat: Garanteix que els controladors gestionin les peticions correctament i retornin les vistes adequades amb les dades necessàries.

- Prova d'un Servei (DiscountService):

Descripció: Comprova que la funció calculate() del servei DiscountService calcula correctament el descompte sobre un preu donat.

Utilitat: Verifica que la lògica de negoci encapsulada en serveis funcioni com s'espera, especialment en càlculs crítics com els descomptes.

- Prova d'un Helper (StringHelper):

Descripció: Assegura que la funció toUpper() del helper StringHelper converteix correctament un string a majúscules.

Utilitat: Garanteix que les funcions auxiliars o helpers funcionin correctament, especialment aquelles que poden ser reutilitzades en diverses parts de l'aplicació.

FEATURE TESTS:

Els feature tests són proves utilitzades en el desenvolupament de programari per validar i optimitzar noves funcionalitats abans de llançar-les completament. Aquestes proves avaluen diferents versions d'una funcionalitat per determinar quina ofereix el millor rendiment o experiència d'usuari. Mitjançant experiments controlats, com els tests A/B, es poden detectar errors potencials i identificar la millor configuració per a una funció específica. Els feature tests es combinen sovint amb el lliurament continu, permetent millores iteratives sense necessitat de redeploys complets, facilitant així un llançament més segur i eficient.

Com a possibles feature tests tindríem aquests exemples:

- Test de Personalització d'una Pàgina:

Descripció: Provar diferents versions d'una pàgina web, com ara una amb una barra de cerca personalitzada i una altra sense.

Utilitat: Identificar quina versió proporciona una millor experiència d'usuari i augmenta la interacció o les conversions.

- Test d'una Nova Funció en una Aplicació:

Descripció: Avaluar una nova funcionalitat, com una opció de pagament amb un clic, en comparació amb el procés tradicional.

Utilitat: Mesurar si la nova funció redueix el temps de transacció i millora la satisfacció de l'usuari.

- Test d'una Interfície d'Usuari Modificada:

Descripció: Comparar una nova interfície d'usuari amb l'antiga per veure quina és més intuïtiva i eficient.

Utilitat: Determinar quins elements visuals i disposicions milloren la navegació i faciliten l'ús de l'aplicació.

- Test d'una Nova Funcionalitat d'Autocompletat:

Descripció: Avaluar si una nova funció d'autocompletat en una barra de cerca ajuda els usuaris a trobar el que busquen més ràpidament.

Utilitat: Millorar la precisió i la velocitat de les cerques, augmentant la satisfacció dels usuaris.

BROWSER TESTS:

Els browser tests són proves realitzades en aplicacions web per assegurar que funcionin correctament en diferents navegadors i dispositius. Això és essencial perquè cada navegador pot interpretar el codi de manera diferent, cosa que pot afectar el rendiment i l'experiència de l'usuari.

Aquestes proves són fonamentals per garantir una experiència de navegació consistent i de qualitat, independentment del navegador o dispositiu utilitzat per l'usuari. Realitzar browser tests ajuda a identificar i corregir problemes abans que els usuaris finals els experimentin, millorant així la satisfacció general i l'accessibilitat del lloc web.

Com a possibles browser tests tindríem aquests exemples:

- Test de Compatibilitat de CSS:

Descripció: Verificar que els estils CSS es mostrin de manera consistent en navegadors com Chrome, Firefox, Safari i Edge.

Utilitat: Assegurar-se que el disseny visual del lloc es mantingui homogeni independentment del navegador.

- Test de Compatibilitat de Funcionalitats JavaScript:

Descripció: Provar que funcions interactives, com ara menús desplegables o formularis dinàmics, funcionalitat igual en tots els navegadors.

Utilitat: Garantir una experiència d'usuari coherent, evitant errors o comportaments inesperats en navegadors menys comuns.

- Test de Responsivitat:

Descripció: Comprovar que el lloc web s'adapti correctament a diferents mides de pantalla, des de mòbils fins a monitors d'escriptori.

Utilitat: Assegurar una experiència d'usuari òptima independentment del dispositiu utilitzat.

- Test de Càrrega de Pàgina:

Descripció: Avaluar el temps de càrrega d'una pàgina web en diferents navegadors i connexions de xarxa.

Utilitat: Millorar la velocitat i l'eficiència del lloc web, evitant que els usuaris abandonin la pàgina per temps de càrrega lents.

API TESTS:

Les API tests són proves dissenyades per validar la funcionalitat, seguretat, rendiment i fiabilitat d'una API (Application Programming Interface). Aquestes proves asseguren que les API responen correctament a les sol·licituds, gestionen adequadament les dades i compleixen amb les especificacions.

Aquestes proves són essencials per assegurar la qualitat i la seguretat dels serveis que es basen en API.

Com a possibles API tests tindríem aquests exemples:

- Test de Funcionalitat:

Descripció: Verifica si les respostes de l'API són correctes segons els requisits establerts.

Utilitat: Garanteix que les funcions de l'API treballen segons el previst.

- Test de Rendiment:

Descripció: Avalua la resposta de l'API sota diverses càrregues.

Utilitat: Assegura que l'API mantingui el rendiment i la velocitat, fins i tot en condicions de trànsit elevat.

- Test de Seguretat:

Descripció: Prova les vulnerabilitats de l'API, com l'autenticació i l'autorització.

Utilitat: Protegeix contra amenaces externes, garantint que l'API sigui segura.

### *6.1.1 Tests aplicat*

Un cop tenim coneixements sobre quins tipus de tests existeixen i per a quins casos utilitzar cadascun arribem a la conclusió de fer servir els Feature Tests per a realitzar les proves de funcionament correcte per al nostre aplicatiu web, ja que ens donaria avantatges com ara:

- Verificació de la funcionalitat completa: Avaluen la funcionalitat d'una característica des de la perspectiva de l'usuari, assegurant que el flux complet d'una característica funciona correctament.
- Detecció primerenca d'errors: Identifiquen errors en la integració de diverses parts del sistema, no només en unitats individuals.
- Millora de la qualitat del codi: Asseguren que les noves característiques no trenquin les existents, millorant la confiança en els desplegaments.
- Facilitat en el refactoring: Permeten canviar el codi intern amb la seguretat que les funcionalitats no es veuen afectades.
- Documentació: Actuen com a documentació sobre com haurien de funcionar les característiques, millorant la mantenibilitat del projecte.

Un punt a tenir en compte a l'hora de fer tests per una aplicació web sobre laravel és l'ús de les factories ja que ens ajuden a crear instàncies de models amb dades falses, però realistes, que es poden utilitzar durant els tests.

## 6.2 Fonts de dades per als tests

Les factories ens proporcionen:

- Dades Consistents i Realistes: Les factories permeten crear dades que imiten les condicions reals del teu entorn de producció. Això ajuda a assegurar que els teus tests es realitzin amb dades que simulen el comportament esperat del teu model en situacions reals.
- Reducció de Codi Repetitiu: Si necessites crear diversos registres de models amb característiques específiques en diferents tests, les factories poden simplificar aquest procés. En lloc de definir les dades manualment en cada test, pots utilitzar factories per generar les dades de manera concisa i reutilitzable.
- Configuració Ràpida de l'Entorn de Test: Les factories faciliten la configuració ràpida de l'entorn de test. Pots crear múltiples registres de models amb una sola línia de codi, cosa que redueix el temps i l'esforç necessari per preparar l'entorn de test.
- Separació de la Lògica de Test i la Dada: Utilitzar factories permet separar la lògica de test de la creació de dades. Això facilita la comprensió i manteniment del codi de test, ja que les factories es poden definir i actualitzar de manera independent dels tests que utilitzin aquestes dades.
- Flexibilitat en la Generació de Dades: Les factories poden ser personalitzades i ampliades per crear una gran varietat de conjunts de dades. Això et permet adaptar les dades generades a les necessitats específiques de cada test.

```php
use Illuminate\Database\Eloquent\Factories\Factory;

class PostFactory extends Factory
{
    protected $model = Post::class;

    public function definition()
    {
        return [
            'title' => $this->faker->sentence,
            'description' => $this->faker->paragraph,
            'user_id' => User::factory(),
            'approved' => $this->faker->boolean,
            'useful_votes' => $this->faker->numberBetween(0, 100),
            'not_useful_votes' => $this->faker->numberBetween(0, 100),
        ];
    }
}
```

**Figura 66.** Codi d'una factoria de dades

## 6.3 Joc de proves manual

Un cop tenim clar ja quins tests fer i les millors pràctiques per a dur a terme aquests passem amb el joc de proves sobre l'aplicatiu web.

Primer farem un joc de proves manual on buscarem trobar algún error detectable a simple vista, com ara desajustos d'estils en diferents tipus de dispositius, comprovació del correcte funcionament del menú d'accessibilitat per a gent amb dificultats, errors a l'hora de crear noves instàncies fent ús de les pantalles del gestor, redireccions errònies...

Provem a visualitzar la pàgina web des d'una tableta movil i veiem com s'adapta correctament i ens permet veure tot el contingut sense cap overflow d'elements, també podem veure com el menú enlloc de veure's a la barra superior ens apareix un menú de tipus hamburguesa a la dreta del menú superior i al fer click se'ns obrirà amb un desplegable cap abaix.


**Figura 67.** Vista de la pàgina des d'un mòbil

Un altra de les proves que hem realitzat ha estat dins l'apartat de research book provar a visualitzar per pantalla i descarregar aquest amb format pdf, veiem com se'ns descarrega correctament.

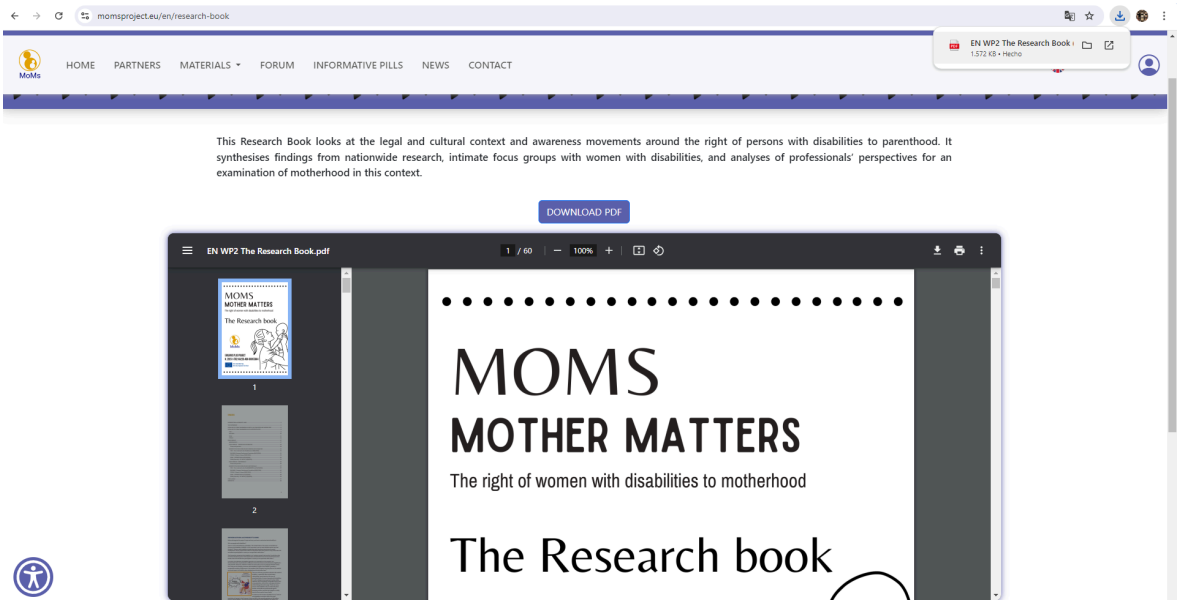


**Figura 68.** Prova de descàrrega del research book

Com a projecte fundat per a la unió europea un dels requisits es que sigui totalment accessible per a tota persona que pugui accedir a la pàgina web, com a proves despleguem el menú i marquem diverses opcions com ara remarcar els títols, els enllaços, alinear tots els textos a la dreta i augmentar el tamany del text de la pàgina.

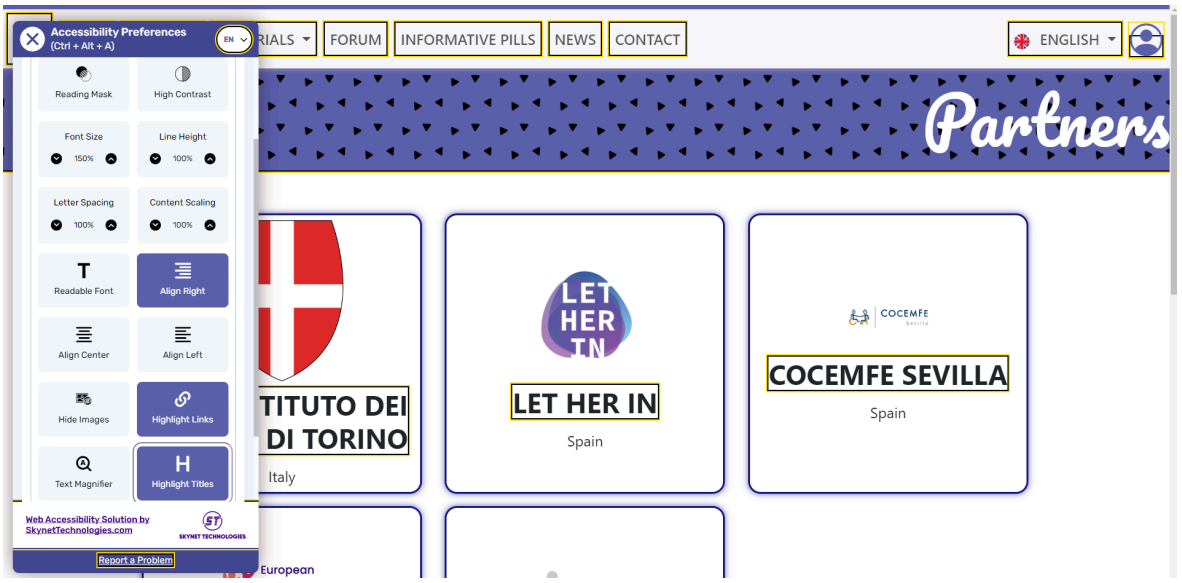


**Figura 69.** Prova de l'eina d'accessibilitat

Provem de crear un nou partner des de el menú del gestor reomplint correctament cadascun dels camps per a fer possible la creació d'aquest.

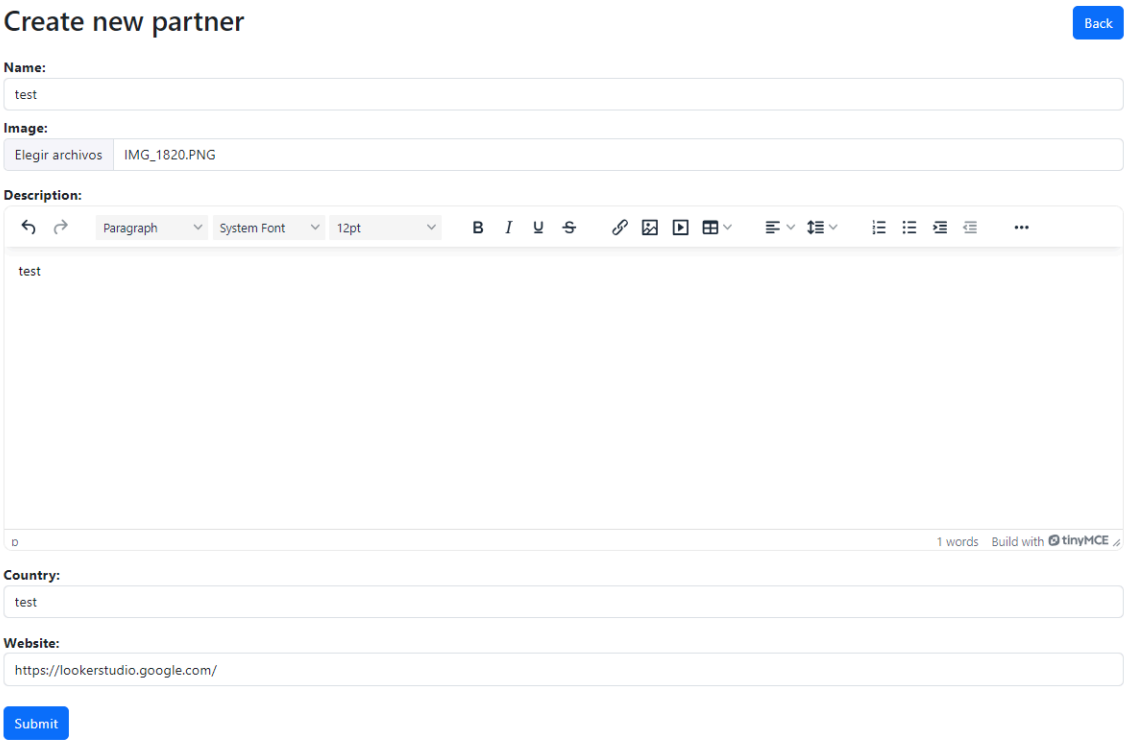


**Figura 70.** Creació d'un nou partner

Un cop creat correctament podem veure com es pot visualitzar correctament aquest nou partner al respectiu apartat de la web.

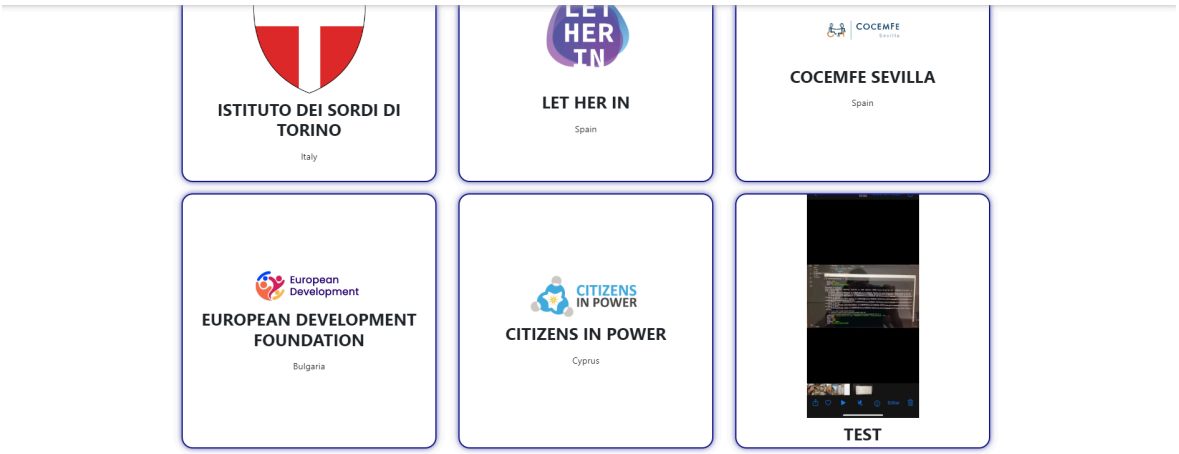


**Figura 71.** Visualització del nou partner creat

Provem de crear una nova entrada al fòrum de forma que aprofitarem per comprovar també el funcionament de filtratge per tags. Per comprovar-ho visualitzarem totes les entrades del fòrum sense aplicar cap filtre d'etiqueta i podem veure com només n'existeix un.

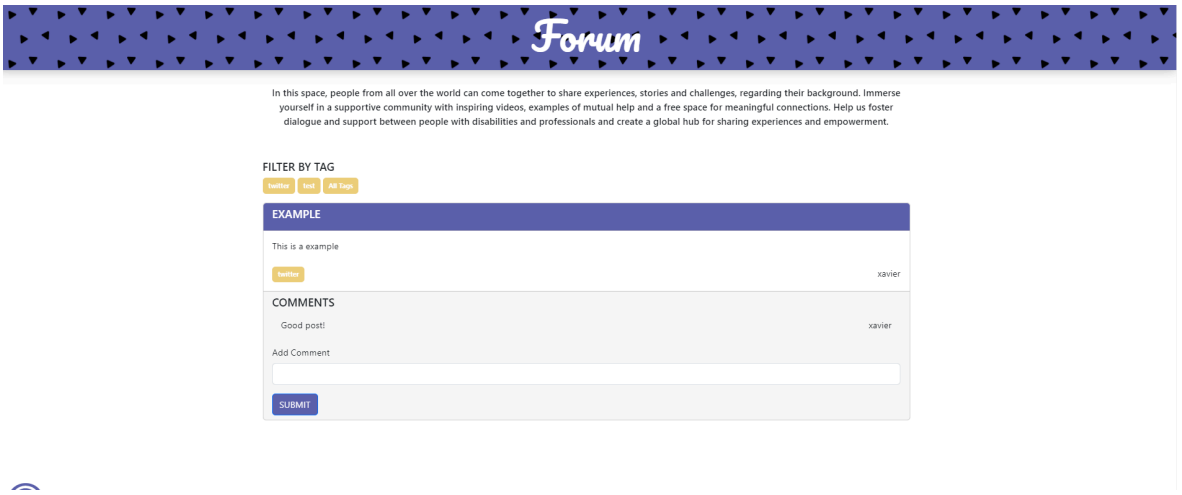


**Figura 72.** Visualització dels posts del fòrum

Creem el nou post i veiem com està pendent de revisió, l'acceptarem.

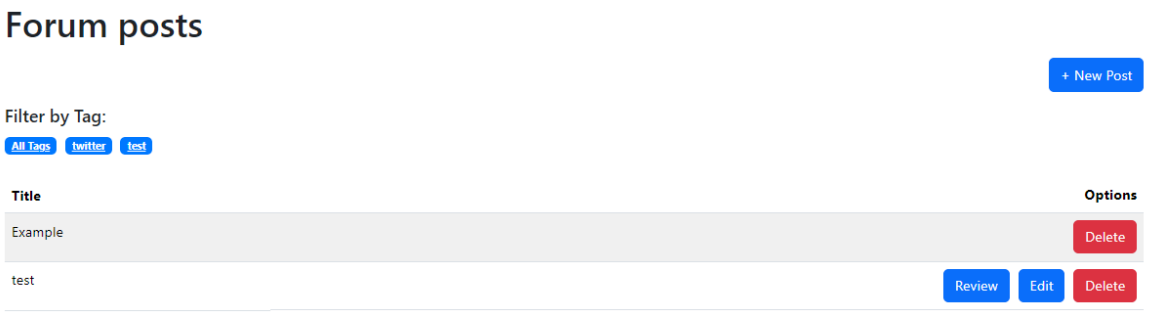


**Figura 73.** Review d'un post del fòrum

I un cop acceptat apliquem el filtre de l'etiqueta que conté el nou post i podem veure com ens apareix aquesta nova entrada.

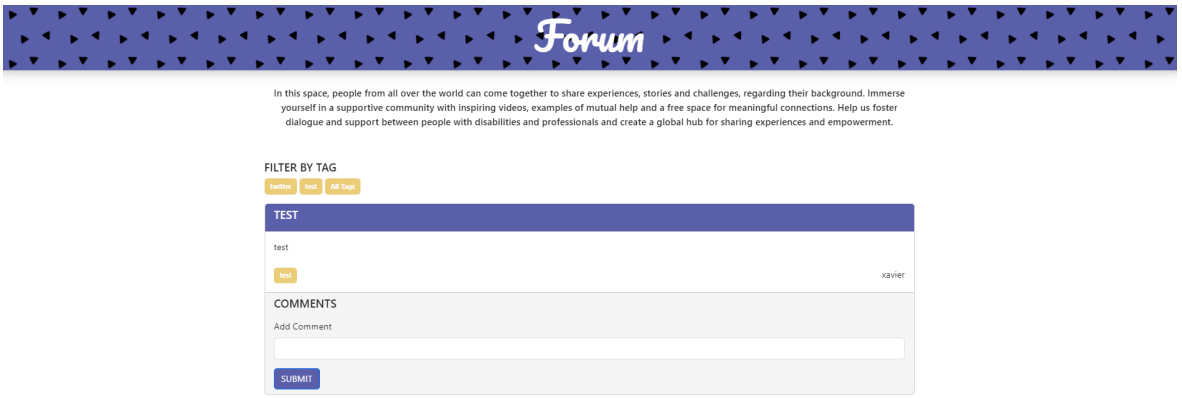


**Figura 74.** Visualització del nou post del fòrum

Des de l'apartat de news farem diverses comprovacions, la primera sera veure si el widget per a compartir el grup de Facebook del projecte s'obri correctament i ens deixa fer-ho sense cap problema, ens obrirà una nova finestra on iniciarem sessió i ens permetrà compartir-lo correctament.

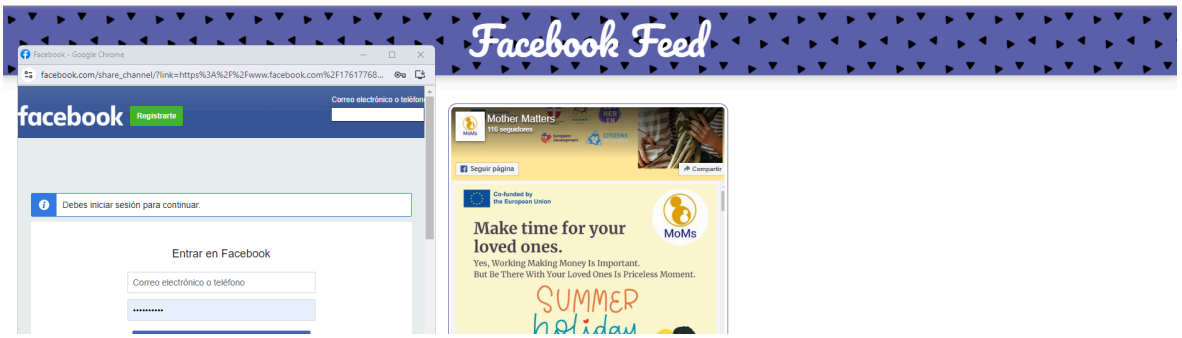


**Figura 75.** Prova del widget de facebook

Ara dins el mateix apartat comprovem les newsletters, inicialment tenim una única newsletter creada.


**Figura 76.** Visualització de les newsletter existents

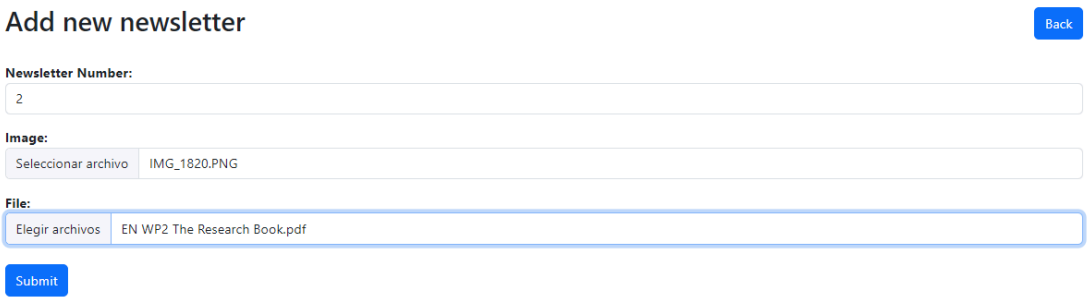


**Figura 77.** Creació d'una nova newsletter

Creem una nova newsletter omplint totes les dades correctament.


**Figura 78.** Prova de visualització de la nova newsletter

I podem veure com ens apareix aquesta nova newsletter disponible per ser visualitzada.

Ara un cop realitzades les comprovacions sense forçar cap tipus d'errada intentarem crear una nova instància sense omplir correctament algun camp o deixant-ne algun en blanc.

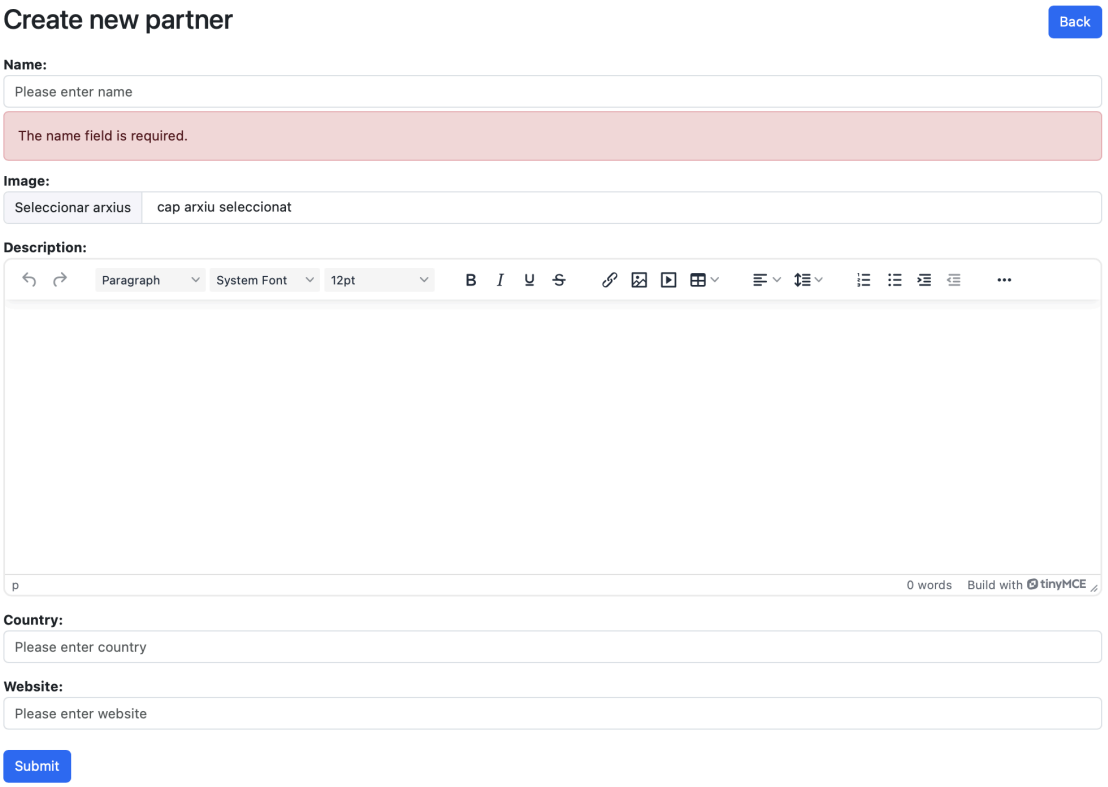


**Figura 79.** Visió d'un missatge d'error

Com podem veure ens mostra un requadre de diàleg amb un missatge indicant que ens hem deixat el camp X que es mandatari per crear aquesta nova instància.

## 6.4 Joc de proves automàtic

Després d'haver fet la recerca de quin tipus de tests són els més adients per al projecte i també haver fet tests manuals com a possible usuari tant sigui gestor com usuari simple que pugui entrar a l'aplicatiu web he creat la següent estructura de tests.

Tests per a comprovar cadascuna de les funcions del controlador i verificar que s'executen individualment sense retornar errors en cas de ser correctament utilitzades, o retornant errors en cas d'haver-n'hi.

També he fet tests per comprovar la visualització correcta de les dades que es llegeixes sobre la base de dades creades prèviament per aquests controladors i posteriorment visualitzades a la pàgina per qualsevol usuari que hi accedeixi.

Primer hem de saber com executar aquests tests ja que laravel incorpora una estructura ja definida amb comandes per ser executats aquests. Es troben dins la carpeta tests a la carpeta arrel del projecte.

Hi ha dues formes d'executar-los, individualment cada classe amb la seva comanda o tots a la vegada amb una comanda global.

**Execució individual:**

*php artisan test tests/Feature/ProfileTest.php*

**Execució global:**

*php artisan test*

```php
public function test_login_form_submits_with_valid_data()
{
    $user = User::factory()->create([
        'password' => bcrypt($password = 'password123'),
    ]);

    $response = $this->post(route('login'), [
        'email' => $user->email,
        'password' => $password,
    ]);

    $response->assertRedirect('/en/home');
    $this->assertAuthenticatedAs($user);
}
```

```php
public function test_login_form_fails_with_invalid_credentials()
{
    $user = User::factory()->create([
        'password' => bcrypt('password123'),
    ]);

    $response = $this->post(route('login'), [
        'email' => $user->email,
        'password' => 'invalid-password',
    ]);

    $response->assertStatus(302);
    $response->assertSessionHasErrors('email');
    $this->assertGuest();
}
```

**Figura 80.** Funcions per fer testing de la vista Login

Per a crear els test fem ús dels Modals de les classes i les factories per crear dades de prova, cada test conté la seva funcionalitat i compara l'execució dels mètodes a provar amb el resultat esperat si funciona correctament.

```php
public function test_comment_can_be_stored(): void
{
    $user = User::factory()->create();
    $post = Post::factory()->create();

    $this->actingAs($user);

    $response = $this->post(route('comments.store', $post->id), [
        'content' => 'This is a comment.',
    ]);

    $response->assertRedirect(route('forum.showApproved', ['lang' => app()->getLocale()]));
    $this->assertDatabaseHas('post_comments', [
        'post_id' => $post->id,
        'user_id' => $user->id,
        'content' => 'This is a comment.',
    ]);
}
```

**Figura 81.** Funció per fer testing del controller Comments

Comprovem una de les funcions del fòrum per afegir comentaris correctament sobre un post.

```
PS C:\Users\TERMINAL4\Documents\GitHub\Moms> php artisan test tests/Feature/Views/auth/LoginPageTest.php

  PASS  Tests\Feature\LoginPageTest
  ✓ login form submits with valid data
  ✓ login form validation
  ✓ login form fails with invalid credentials

  Tests:    3 passed (12 assertions)
  Duration: 1.98s
```

**Figura 82.** Execució dels test de la vista de Login

Aquí podem veure l'execució individual d'una classe de tests que comprova que la vista de la pàgina de login funciona correctament, tant la seva lògica com aparença.

```
PS C:\Users\TERMINAL4\Documents\GitHub\Moms> php artisan test tests/Feature/Dashboard/forum/comments/CommentsControllerTest.php

  PASS  Tests\Feature\CommentsControllerTest
  ✓ comment can be stored
  ✓ store comment requires content

  Tests:    2 passed (6 assertions)
  Duration: 1.05s
```

**Figura 83.** Execució dels test del controller per als comentaris del fòrum

Per altra part podem veure com es comprova amb aquests test la part lògica del controller que tot s'executi i funcioni tal com s'espera que faci la funcionalitat definida prèviament amb els requisits determinats.

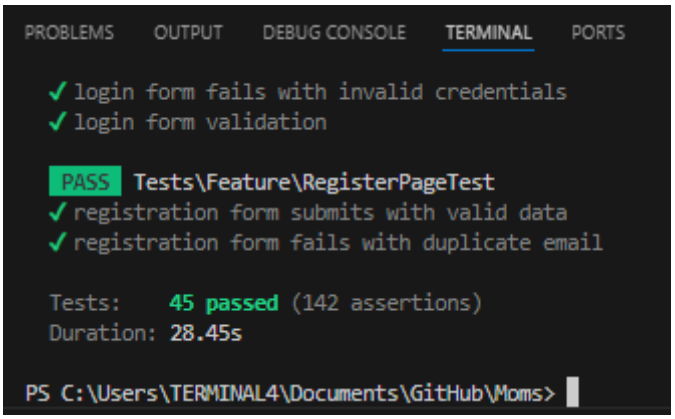


**Figura 84.** Execució global dels test de l'aplicatiu

Finalment executem la comanda per testejar l'aplicació sencera i podem veure com tots els tests definits passem correctament cadascun d'aquests.

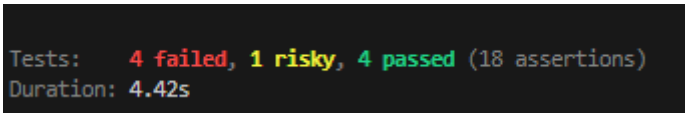


**Figura 85.** Execució de test amb errors

En cas d'haver-hi errors es mostrà d'aquesta forma per a què el programador en sigui conscient. A l'informació que ens proporciona l'execució dels test sempre defineix el nombre de test errats

# 7 Conclusions

## 7.1 Avaluació personal

Aquest projecte ha estat molt beneficiós per ampliar els meus coneixements i posar en pràctica l'après durant el grau a la URV. Per una part, existeix la complexitat d'analitzar els requisits i dissenyar una BD coherent que s'acommodi a les dades necessàries. Per altra part, hi ha la dificultat d'haver fet un treball com a full stack developer i, per tant, haver desenvolupat no només un backend, sinó també la part del front end que consumeix d'aquest backend.

La principal dificultat ha estat aprendre un llenguatge com es PHP de nou i el seu framework amb el que s'ha desenvolupat essent Laravel, integrant la part de disseny amb CSS i algunes funcionalitats extres amb Javascript. Al principi després de fer un estudi i un petit projecte per entendre el funcionament en conjunt d'aquest framework, tenint en compte la codificació de l'API, la configuració i crida d'aquestes rutes... Però finalment ha estat relativament fàcil un cop comprés el funcionament.

Aquest projecte m'ajudarà professionalment per demostrar que no tinc gaire dificultat en aprendre i adaptar-me a un nou entorn de desenvolupament, així com per tenir unes nocions bàsiques de PHP, que, donat el mercat laboral per desenvolupadors de software, són coneixements bastant valuosos.

A més a més de la feina sol·licitada per l'unió europea, he decidit afegir ja que tenia temps i volia aprofundir és amb el tema del testing, ja que al grau s'explica però molt breument, de forma que he realitzat un estudi sobre quin tipus de tests i parlant amb el tutor he decidit quin seria el més adient per al projecte.

## 7.2 Coneixements aplicats al TFG

A continuació s'exposa una llista d'assignatures i els seus coneixements adquirits al GTDAWIM i aplicats al TFG:

- Anàlisi i Disseny d'Aplicacions: la primera etapa del desenvolupament és realitzar una anàlisi dels requisits i, sobretot, les classes necessàries per mantenir les dades de la forma més coherent possible.
- Bases de Dades: aquesta assignatura m'ha ajudat principalment a entendre el llenguatge SQL i com transformar correctament un diagrama de classes a un model relacional i identificar claus i taules intermèdies necessàries.

# 8 Referències

[1] Visual Studio Code – URL https://code.visualstudio.com/ – Online
[2] PHP – URL https://www.php.net/docs.php – Online
[3] Laravel – URL https://laravel.com/docs/11.x – Online
[4] Bootstrap – URL https://getbootstrap.com/ – Online
[5] Git i GitHub Git – URL https://www.simplilearn.com/tutorials/git-tutorial/what-is-git– Online
[6] MySQL – URL https://www.mysql.com/ – Online
[7] NPM – URL https://www.npmjs.com/– Online
[8] MagicDraw – URL https://www.magicdraw.com/main.php/– Online
[9] MVC – URL https://www.geeksforgeeks.org/mvc-design-pattern/ – Online
[10] Unit Testing – URL https://aws.amazon.com/es/what-is/unit-testing/ – Online
[11] Feature Testing – URL https://www.optimizely.com/optimization-glossary/feature-test/ – Online
[12] Cross Browser Testing – URL https://developer.mozilla.org/es/docs/Learn/Tools_and_testing/Cross_browser_testing – Online
[13] API Testing – URL https://qalified.com/es/blog/api-testing/ – Online

# 9 Annex

## 9.1 Instal·lació i configuració

Per realitzar canvis sempre es recomanat provar-los localment i després desplegar-los a producció, la forma en que podem preparar canvis requerits per al projecte o afegir noves funcionalitats, el projecte està penjat a un repositori de github privat al domini de l'empresa, de forma que ens clonaríem el projecte amb la següent comanda i ja estarà llest per treballar localment:

*"git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY"*

## 9.2 Requisits

Per treballar localment amb aquest repositori necessitarem tenir el paquet xampp instal·lat de forma que puguem iniciar una base de dades local MySQL i un servidor web com és Apache, que ens permetrà accedir i visualitzar el projecte.

En quant als requisits per accedir al lloc web només en necessitaríem un dispositiu de qualsevol tipus amb accés a internet.

## 9.3 Treball pendent de fer

La funcionalitat de la pàgina ha quedat completament definida i finalitzada però com a extra es podria automatitzar la part del testing, configurant el respositori de Github sobre el que treballem amb les eines de CI/CD que proporciona de forma que ens executi els test creats cada cop que es faci un push d'un commit a aquest.

Un altre aspecte a millorar a l'apartat de testing sería calcular el percentatge de coverage del testing de l'app.