# **Informe i documentació de la utilització de PHP, JS, CSS, templates y SQL en el TFM**

**Alumne**: Salvador Peiró Palmer

**Assignatura**: Treball Final de Màster

# **Índex**

1. [Justificació utilització PHP pàg.](#_1._Justificació_utilització) 
   1. [Justificació utilització arxiu .htaccess pàg.](#_1.1._Justificació_utilització)
2. [Creació del model MVC pàg.](#_2._Creació_del) 
   1. [Explicació carpeta Controlers. pàg.](#_2.1._Explicació_carpeta) 
      1. [Arxiu Autoload.php pàg.](#_2.1.1._Arxiu_Autoload.php)
      2. [Conexion.php pàg.](#_2.1.2._Conexion.php)
      3. [Controlers.php pàg.](#_2.1.3._Controlers.php)
      4. [Load.php pàg.](#_2.1.4._Load.php)
      5. [Mysql.php pàg.](#_2.1.5._Mysql.php)
      6. [Views.php pàg.](#_2.1.6._Views.php)
3. [Connexió a la base de dades. pàg.](#_3._Connexió_a) 
   1. [Creació Base de Dades y relació entre elles pàg.](#_Creació_Base_de)
4. [Utilització template per a la gestió de rols i usuaris. pàg.](#_Utilització_template_per)
5. [Configuració del Template ValiADmin pàg.](#_5._Configuració_del)
6. [Sistemes de Rol i permisos d’usuari pàg.](#_6._Sistemes_de) 
   1. [Creació dels botons dels rols pàg.](#_6.1._Creació_)
   2. [Funcionalitat botons pàg.](#_6.2._Funcionalitat_botons) 
      1. [Funcionalitat boto modificar pàg.](#_6.2.1._Funcionalitat_boto)
      2. [Funcionalitat boto eliminar pàg.](#_6.2.2._Funcionalitat_boto)
      3. [Funcionalitat boto permisos pàg.](#_6.2.3._Funcionalitat_boto)
7. Sistema d’ususaris

# **1. Justificació utilització PHP i model MVC**

Una vegada analitzada la possibilitat d’utilitzar els llenguatges de prograamció PHP i JavaScript, per accedir a las base de dades, he optat per PHP. Tot i que ambdues son bones opcions, i que per exemple amb la informació cercada a les xarxes, JavaScript es més ràpid que PHP, he optat per PHP per ser un llenguatge de servidor, la qual cosa, fa que proporciones una capa addicional de seguretat al no ser visible el codi per als usuaris, no com passa amb JS. Això és fonamental per garantir la seguretat de les dades dels clients i la integritat de la plataforma en línia, atorgant una base sòlida per a la realització del teu TFM de desenvolupament web.

En relació a la seguretat, PHP ofereix protocols de connexió a la base de dades, mitjançant les llibreries integrades, com PDO que ajuda a previndre atacs com SQL injection com es va estudiar en l’assignatura desenvolupament de PHP.

Altra avantatge que te PHP es que demostra ser escalable en un lloc web de diferents pàgines, gràcies a la possibilitat d’utilitzar funcions i classes per gestionar la connexió a la base de dades i altres tasques relacionades al manteniment i la gestió.

Per tant, al ser una opció robusta i segura per al accés de la base de dades, i que aixina en el meu TFM serà més ric en varietat en llenguatges al utilitzar-lo.

I en relació a la justificació de la utilització del patró d’arquitectura Model-View-Controller (MVC), comentar que es el model aprés en la assignatura Desenvolupament front-end abm frameworks JavaScript. En aquesta assignatura vaig apendre que és una forma d’organitzar la estructura necessària en diferents carpetes, de tal forma que siga més sencill detectar on esta cada part del codi per si necessite modificar alguna cosa, i també que permet la reutilització del codi.

Tal i com senyala la web possibleapp “*MVC és un patró de disseny enfocat a separar les responsabilitats dins de la nostra aplicació i és molt utilitzat a la web pel seu enfocament i els avantatges que ofereix pel que fa a algunes altres formes o patrons de desenvolupament d'aplicacions web.*” (*font:* [*https://possibleapp.com/*](https://possibleapp.com/) *i el link* [*https://acortar.link/xg8BFq*](https://acortar.link/xg8BFq))

En el meu projecte he creat les següents carpetes:

* **Views** 🡺 És on tindrem totes les vistes
* **Model** 🡺 On estaran els models
* **Controller** 🡺 que es on estaran tots els controladors
* **Assets** 🡺 tindrem els arxius de JS, CSS, fonts...
* **Config** 🡺 Tindrem alguns arxius de configuració
* **Helpers** 🡺 Arxius que contindran la configuració per estructurar determinats objectes.
* **Libraires** 🡺 On contindrem les llibreries.

# **1.1. Justificació utilització arxiu .htaccess**

L'ús d'un arxiu .htaccess esdevé fonamental per a la gestió eficient d'una pàgina web que utilitza la tecnologia Apache, com és el cas del meu projecte de botiga en línia de jocs de taula. A través d'aquest arxiu, seré capaç de proporcionar funcionalitats clau per millorar la seguretat, l'experiència de l'usuari i la gestió global del lloc web.

Un dels principals avantatges de l'ús d'un arxiu .htaccess és la capacitat de gestionar l'accés als directoris i arxius de la pàgina web. Això permet implementar restriccions d'accés específiques, com ara limitar l'accés a certes àrees del lloc web només a l'administrador. Per exemple, suposem que la botiga en línia de jocs de taula té una àrea d'administració on es gestionen els productes, les comandes i els clients. A través de l'arxiu .htaccess, es pot configurar una restricció d'accés per a aquesta àrea, de manera que només els administradors amb les credencials correctes puguin accedir-hi. Això garanteix la protecció de la informació sensible i evita manipulacions no autoritzades.

Una altra funcionalitat clau que proporciona l'arxiu .htaccess és la capacitat de gestionar les redireccions d'URL de manera eficaç. A través de redireccions amigables, podré assegurar que els usuaris que accedeixin a URL antigues, que potser ja no estiguin en ús, siguin redirigits de forma transparent a pàgines actives. Això millora significativament l'experiència de l'usuari i evita que es trobin amb errors 404, millorant la usabilitat general del lloc web.

***Funcionalitats***:

**Prevenció d'intrusions**: La configuració .htaccess evita la navegació entre directoris, protegint carpetes sensibles com ara "admin" o "uploads" d'accessos no autoritzats. Això blinda la teva ecomerce contra possibles intrusions i vulnerabilitats.

**Ocultació de carpetes**: El .htaccess pot ocultar les carpetes que contenen fitxers de configuració o scripts sensibles, protegint la teva propietat intel·lectual i els secrets del teu projecte.

**Navegació fluida**: Elimina la llista d'index, evitant la visualització d'un llistat poc atractiu dels fitxers del directori si no hi ha un index.php present. Això contribueix a una experiència d'usuari més fluida i professional.

**Implementació de URLs amigables**: El .htaccess permet reescriure les URLs complexes per transformar-les en URLs fàcils de llegir i recordar. Això beneficia tant la usabilitat per als usuaris com la indexació per part dels motors de cerca. Això ens permet navegar entre les diferents pàgines del meu projecte.

A continuació es visualitza el codi utilitzat amb els seus comentaris:

# Aquestes configuració evita la navegació entre directoris.

# font de configuració https://www.youtube.com/watch?v=y9KEgh6yXME&list=PLH\_tVOsiVGzkGD2sB-cmSkxKzvoMYZrHL

# Desactiva l'índex de directoris per evitar mostrar un llistat del contingut del directori si no hi ha un index.

# Protegeix la visualització de les carpetes i aixina sols poden accedir els que tenen permisos.

Options ALL -Indexes

# Habilita el motor de reescriptura.

RewriteEngine on

# Verifica si la sol·licitud no fa referència a un directori existent.

RewriteCond %{REQUEST\_FILENAME} !-d

# Verifica si la sol·licitud no fa referència a un fitxer existent.

# Fa que una condició ha de ser verdadera per a que procesada la url

RewriteCond %{REQUEST\_FILENAME} !-f

# Redirigeix totes les sol·licituds que no són fitxers ni directoris existents a l'index.php amb la URL com a paràmetre 'url'.

# [QSA] - Afegeix els paràmetres de la cadena de consulta a la URL reescrita.

# [L] - Finalitza el procés de reescriptura i no executa altres regles si aquesta és aplicada.

RewriteRule ^(.\*)$ index.php?url=$1 [QSA,L]

Fonts:

<https://lc.cx/aHJ99g> ¿Qué es el fichero .htaccess? Para que se usa y como se depura.

<https://lc.cx/KNTxRW> 01 - 💻 HTACCESS - [Que es htaccess] Video YT

# **2. Creació del model MVC**

En primer lloc, explicaré com he preparat la base del meu model MVC carpeta per carpeta. Tot i que el codi està comentat explicaré la motivació de la utilització de cada carpeta. La utilització del model MVC en aquest projecte, te com a finalitat facilitar l’organització dels fitxers que contenen el codi, com vaig apendre en l’assignatura dessenvolupament front-end amb frameworks JavaScript.

Aquest patrò de disseny de software m’ajudarà a millorar la lógica del meu disseny. Amb aquest patro dividiré el Model, el View i el controller. Les carpetes que utilitzaré serán:

Els següents subapartats, es referiran a les diferents carpetes que he utilitzat per fer la base del model.

Organització de carpetes:

* **Assets**: Conté imatges, CSS i JavaScript. Això permet centralitzar el disseny CSS i dividir el JavaScript en arxius separats, evitant la càrrega innecessària de totes les funcions en cada pàgina.
* **Config**: Conté les rutes del projecte, dades de connexió a la base de dades, configuració horària i altres variables. Això redueix la duplicació de codi i facilita la lectura i el manteniment.
* **Controllers**: Conté els controladors de cada pàgina del projecte. Cada controlador s'encarrega de gestionar la lògica específica d'una pàgina.
* **Helpers**: Conté funcions i classes que s'utilitzen per a simplificar i agilitzar el codi. Per exemple, la funció headerAdmin carrega l'arxiu header\_admin.php.
* **Libraries**: Conté llibreries externes que s'utilitzen en el projecte.

Model: Conté la lògica d'accés a la base de dades SQL.

* **Views**: Conté les vistes del projecte, que són els arxius HTML que es mostren a l'usuari.

## **2.1. Explicació carpeta Controlers.**

Aquesta carpeta conte una codificació realitzada per gestionar de millor forma la nostra aplicació.

### **2.1.1. Arxiu Autoload.php**

Aquest arxiu te com a funció descarregar les classes que necessitem en cada moment, segons el nom i la direcció de la pàgina que estem utilitzan.

<?php

//spl\_autoload\_register és una funció de PHP que permet registrar una o diverses funcions d'autocarregament de classes. La seva finalitat és carregar automàticament les classes

spl\_autoload\_register(function ($class) {

//Ara busque si existeix la classe en la carpeta Libraires/Core i si existeix el requereix

if (file\_exists('Libraires/Core/'.$class.'.php')) {

require\_once('Libraires/Core/'.$class.'.php');

}

});

?>

### **2.1.2. Conexion.php**

Aquest arxiu utilitza la técnica PDO apresa en l’assignatura de PHP, per evitar injeccions, tal i com s’explica en l’apartat corresponent.

### **2.1.3. Controlers.php**

Es l’arxiu on es realitza al carrega dels controladors i la trucada als models. Amb la funció loadModel, el que aconseguiré es capturar la classe que utilitza la nostra web. Per això és important la concordancia entre el nom de la classe, el del arxiu del controlador i el de la View, ja que si son diferents, no funcionarà la nostra web. Ja que la variable models, buscarà el nom de l’arxiu denominat amb el nom de la web dins la carpeta Models.

### **2.1.4. Load.php**

El primer que fa aquest codi es convertir la primera lletra del nostre controlador en majúscula amb ucwords. Això ho faig, per mantenir la concordancia en els noms, ja que hi ha servidors sensibles a les majúscules i minúscules (https://www.arsys.es/blog/mayusculas-minusculas-gestion-servidores)

El que fa es buscar el controlador dins la capreta Controllers i sinó la troba, el que retonra es la pàgina d’error, per evitar el error 404, que es pàgina no trobada. Amb aquesta configuració quedarà més estètica la pàgina web.

### **2.1.5. Mysql.php**

Aquest arxiu gestiona tot el que te que vore en la gestió de bases de dades i les ordres que donarem. Per exemple, es crea la funció per eliminar registres (delete() o la de seleccionar tots els usuars amb select\_all()

### **2.1.6. Views.php**

Gestiona la vista de cada apartat de la nostra web. Això ho aconseguim mitjançant la captura del controlador(que seria la primera subcarpeta de Views en el meu MVC i de la vista, que es l’arxiu que es troba dins de la nostra subcarpeta)

# **3. Connexió a la base de dades.**

Dins del model MVC, la connexió a la base de dades, l’he incorporat a la subcarpeta Core de la carpeta Libraries.

He optat per utilitzar el programa XAMPP per crear la base de dades en el meu entorn local. Aquesta decisió es fonamenta en dues raons principals. En primer lloc, durant el desenvolupament de l'assignatura de PHP, ja havia utilitzat XAMPP, la qual cosa em va proporcionar familiaritat amb el seu funcionament. En segon lloc, XAMPP és una ferramenta àmpliament utilitzada i compta amb una comunitat activa, la qual cosa facilita l'obtenció de suport i la resolució de problemes.

El primer pas va ser crear una base de dades a la qual he anomenat 'db\_TFM'. Per establir la connexió amb la base de dades, he utilitzat PDO a la classe 'Conexion'. Donat que el codi està degudament comentat, estime que no es necessari afegir més explicacions addicionals en aquest punt.

class Conexion{

    private $host = "localhost"; // Nom del nostre servidor

    private $user = "root"; //Nom del nostre usuari

    private $password = ""; //Contrasenya per conectar

    private $db= "db\_TFM"; //Nom de la base de dades

    private $conect; //S'utilitza per a les instruccions de SQL

//Inicie el meu métode constructor per operar amb el nosre SQL

    public function \_\_construct(){

        $connectionString = "mysql:host=".$this->host.";dbname=".$this->db.";charset=utf8";

        try{

            //PDO(conexio al servidor, usuari, password)

            $this->conect = new PDO($connectionString, $this->user, $this->password);

            //a continuació intentem capturar els errors al conectar-me a la base de dades, aixina, detectaré més rapidament els problemes

            $this->conect->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

        }

        catch(Exception $e){

            $this->$conect = "Error de conexió";

            echo "Error". $e->getMessage();

        }

    }

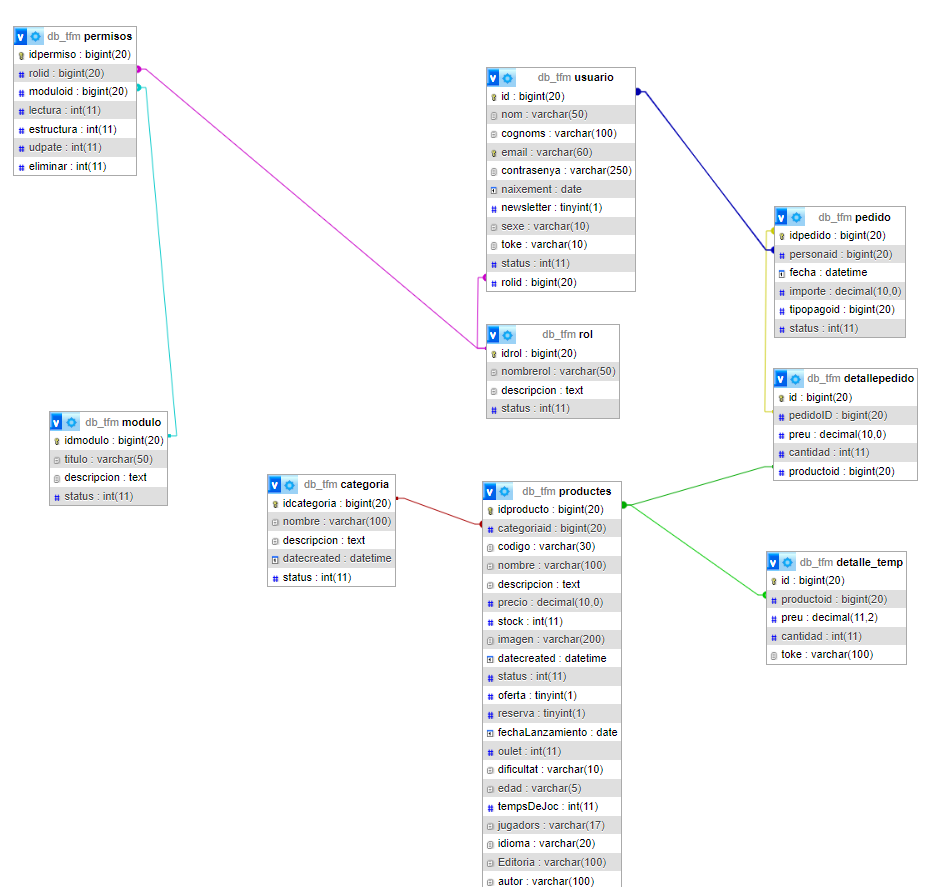
}

## **3.1. Creació Base de Dades i relació entre elles.**

Per a la gestió de la meua tenda virutal, he creat les següents bases de dades:

* Categoria
* Productes
* Detalle\_temp
* Detallepedido
* Pedido
* Usuario
* Rol
* Permisos
* Modulo

La relació de les bases de dades pot veure a continuació:



Per al desenvolupament del meu TFM he creat 9 bases de dades, per a gestionar tota la informació del meu e-comerce. Ja que una sola base de dades no seria operatiu. Com es pot vore en la següent captura d’imatge, les bases de dades, de forma directa o indirecta, estàn relacionades, pel mètode CASCADE, ja que això permet actualitzar la informació de forma automàtica. No obstant, pot ser canvie l’estructura de categories, ja que al poder tenir diferents categories els jocs de taula no sé com d’operatiu es tenir-ho per separat.

Per a no fer una explicació extensa i innecessària de la motivació de cadascun dels caps de totes les bases de dades, ja que son camps similars, a continuació explicaré únicament la justificació de la utilització dels camps de la base de dades productes,

**- idproducto** 🡺 com que el id dels productes han de ser únics, per evitar errors en les relacions, aquest camp es marca a la base de dades com únic. El tipus de dada , es bigint ja que en una botiga no sabrem amb total seguretat quants usaris tindrem, ni número de productes... a més a més, per homogeinetzar aquests camps que estàn relacionats, he optat per utilitzar bigint amb 20 caràcters, més que de sobra per al meu projecte. Tot i que seria més convenient per la memòria que utiltiza INT que Bigint ja que utilitza menys informació (informació obtinguda a https://learn.microsoft.com/es-es/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql?view=sql-server-ver16) el meu projecte ha de respectar, en la mesura del que siga possible, el que seria una tenda virtual real.

- **categoriaid** 🡺 la categoriaid prove de la base de dades de categoria i per a poder relacionar-lo, he d’utilitzar el mateix format que en dita base de dades. La justificació de la utilització d’aquest tipus de dades en la base de dades categoria, es la mateixa que en l’apartat anterior.

- **precio** 🡺 el preu al ser númeric i de dos decimals, utilitza el format decimal, indicant que te un màxim de 10 caràcters a l’esquerra de la coma i dos decimals.

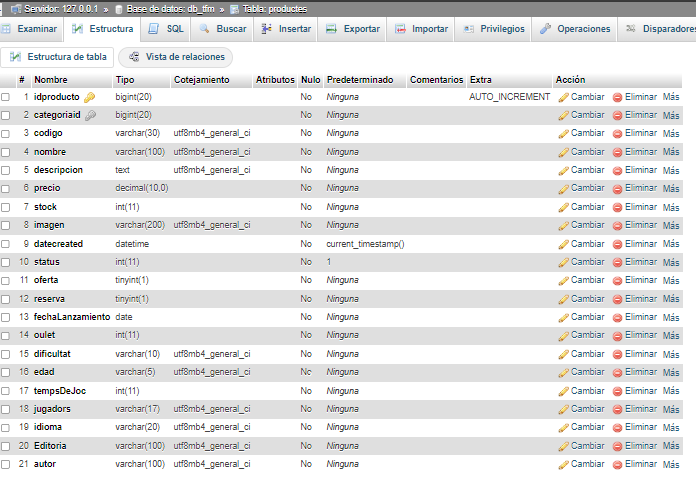
- **codigo** 🡺 nombre, dificultat editorial i autor en aquests casos, són camps que contrindran textos però d’una llargària reduïda, per això, el màxim d’aquests camps no exedeixen la longitud màxima de cadena en bytes de 100, no son els caràctes(font: https://learn.microsoft.com/es-es/sql/t-sql/data-types/char-and-varchar-transact-sql?view=sql-server-ver16) .

- **descripcio** 🡺 s’utilitza un camp de tipo text, ja que la seua ampliaria pot ser molt llarga i no es coneix l’amplària de text que descriurà el joc de taula o producte.

- **stock** 🡺 el número d’estoc ha de ser un nombre sencer, ja que no es pot vendre “mig joc”, per això el tipo de dada es INT, que fa referencia als números sencers.

- **fecha\_lanzamiento** 🡺 al ser un camp de data, ha d’utilitzar-se el camp date, però hi ha que tenir en compte de formatejar el format data, ja que es de tipus aaaa/mm/dd. Per a retornar de forma correcta la data hauré d’utiltizar date(‘d/m/y’, strtotime()).

- **oferta i reserva** 🡺 son de tipo bolea ja que em servirà per marcar amb un true o false si es troben en aquestes situacions.



Com es pot veure en la següent captura de pantalla, idproducto es de tipo



# **4. Utilització template per a la gestió de rols i usuaris.**

Com a estudiant que està elaborant un treball final de desenvolupament web, és fonamental justificar l'ús de plantilles o templates per al desenvolupament de la pàgina web. En aquest cas, l'elecció d'utilitzar el template https://pratikborsadiya.in/vali-admin/ amb Bootstrap 5 està plenament justificada per diverses raons, les quals es detallen a continuació:

* **Demostració d'habilitats**: En el TFM és deu demostrar els coneixements que he aprés durant el curs, per això he optat per utilitzar é una plantilla complexa com Vali Admin amb Bootstrap 5 permet demostrar la capacitat d'integrar tecnologies diverses i crear una aplicació web de gran qualitat.
* **Personalització de la plantilla**: ETot i que utilitzaré com a base la plantilla, aquesta la personalitzaré per a adaptar-la a les necessitats del meu projecte, per tant, no serà un “copy and paste” sinó un treball que comportarà un grau d’elaboració. Això també es podrà percebre amb la gran quantitat de comentaris que te el meu text.
* **Eficiència en el desenvolupament**: les meues habilitats en HTML i CSS es demostraran en la part de la tenda que veu el usuari, es a dir on es fan les compres, la part que es veu pels usuaris que no son administradors del lloc web. Utilitzar una plantilla de qualitat com Vali Admin permet accelerar el procés de desenvolupament, en la part de gestió d’usuaris i rols. En comptes de crear cada component i estil des de zero, es pot aprofitar el treball previ dels desenvolupadors de la plantilla, reduint considerablement el temps i els recursos necessaris per al desenvolupament, i dedicar-los a altres tasques que demostren els meus coneixements.
* **Gestió més fàcil de la part d'administració**: Utilitzar una plantilla per a la part d'administració de l'aplicació és particularment útil per a una gestió més fàcil i eficient. Les plantilles com Vali Admin estan dissenyades específicament per a aquest propòsit, oferint una interfície d'usuari intuitiva i funcions avançades per a la gestió de continguts i usuaris.

En resum, l'ús de la plantilla Vali Admin amb Bootstrap 5 per al desenvolupament de la pàgina web del TFM és una decisió estratègica que permet demostrar habilitats tècniques, accelerar el desenvolupament, garantir una experiència d'usuari de qualitat i facilitar la gestió de l'aplicació. A més, proporciona una base sòlida per a la personalització i adaptació segons les necessitats específiques del projecte.

# **5. Configuració del Template ValiADmin**

El primer que he fetr es descarregar-me el template en https:://patrikbordsadiya.in/vali-admin/ per obtener la plantilla necesaria per començar el meu projecte de la botiga. La seua integració es realitzarà en la meua plantilla MVC, la qual, ja ha estat explicada anteriorment.

El primer que faig es dins de la carpeta descarregada de vali-admin-master copiar els arxius css i js en la meua carpeta Assets, i en la subcarpeta corresponent de CSS i JS, per tenir-los dins del meu MVC.

A continuació copie el codi HTML de l’arxiu Black\_page.html i el pegue en dashboard.php que incorporaré en Views. El arxiu estarà en la carpeta Dashboard.

A continuació cree l’arxiu Dasboard.php i li pegue el contingut de Home.php, canviant-li la classe que ara es dirà Dashboard i la funció també es dirà dashboard. A més, canviaré la informació del pag\_tag, page\_title i page\_name per a saber que estic en Dashboard.

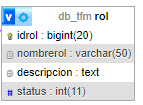
Ara ja podré visualitzar la meua pàgina en http://localhost/TFM/dashboard, però els estils encara no es carregaran. Per a carregar les dades de forma correcta, això és com a conseqüència de que no està be l’enllaç del CSS, per a solventar aquest problema, utilitzaré un helper denominat media(), que el que fa es indicar la base URL http://localhost/TFM i ho concatena amb la cadena /Assets, i per tant, si ho

# **6. Sistemes de Rol i permisos d’usuari**

A continuació, explicaré la part més important del codi que gestiona la part dels usuaris, ja que explicar cada detall faria massa extens aquest document, i hi ha parts molt repetitives, que no és necessari reproduir.

## **6.1. Creació dels rols en la base de dades**

El primer que farem es crear el mètode per obtenir aquests registres. La taula que vaig a utilitzar es la taula denominada rol



Aquesta taula te quatre registres. El primer que farem es configurar la connexió amb la base de dades. La configuració a la connexió la trobarem en Config/Config.php, que es la següent:

//creem les variables per a conectarnos a la base de dades

const DB\_HOST = "localhost"; // Nom del nostre servidor

const DB\_USER = "root"; //Nom del nostre usuari

const DB\_PASSWORD = ""; //Contrasenya per conectar

const DB\_NAME= "db\_TFM"; //Nom de la base de dades

const DB\_CHARSET = "charset=utf8";

Ara anem a Controllers/Roles.php, i creem un mètode per a la conexio denominat getRolest (get per ser un mètode de captura de dades) on cridem a mdeol de

        public function getRoles(){

            $arrayDades = $this -> model -> selectRoles();

        }

En aquest funció cridem al model (recordem que model ve de Libraires\Core\Controllers.php de la funció loadModel() que captura les classes.

A continuació en la carpeta Models, cree l’arxiu rolesModel.php i copie el contigunt de homeModel, ja que és la base per a crear models i cridar a les bases de dades.

En aquest arxiu agrege la funció selectRoles() que encara no existeix,

        public function getRoles(){

            $arrayDades = $this -> model -> selectRoles();

//ara el que fem es convertir el array en format JSON amb la funció json\_encode

            echo json\_encode($arrayDades, JSON\_UNESCAPED\_UNICODE);

            //die el que fa es finalitzar el proces

            die();

        }

En relació a la ordre de SQL el que ens esta deient que seleccione de la taula rol tots aquells stats diferents a valor 0, i com que els estats seran 1 actiu i 2 inactiu, deuria seleccionar-los tots.

            $sql = "SELECT \* FROM rol WHERE status !=0";

Per a la configuració de la tabla he utilitzat la documentació de la següent pàgina web:

<https://datatables.net/examples/ajax/objects.html>



Com es pot veure es necessari jquery per a utilitzar-lo, i el codi utilitza es

//Cree una funcio per a que quan es carregue tota la pàgina s'execute

document.addEventListener('DOMContentLoaded', function(){

    //utilitzem el format que trobem en https://datatables.net/examples/ajax/objects.html

    tableRoles = new DataTable('#tableRoles', {

        "aProcessing":true,

        "aServerSide":true,

        "language": {

            //Idioma en català

            "url": "//cdn.datatables.net/plug-ins/1.10.20/i18n/Catalan.json"

        },

        "ajax":{

            "url": " "+base\_url+"/Roles/getRoles",

            "dataSrc":""

        },

        "columns":[

            {"data":"idrol"},

            {"data":"nombrerol"},

            {"data":"descripcion"},

            {"data":"status"},

            {"data":"options"}

        ],

        "resonsieve":"true",

        "bDestroy": true,

        "iDisplayLength": 10,

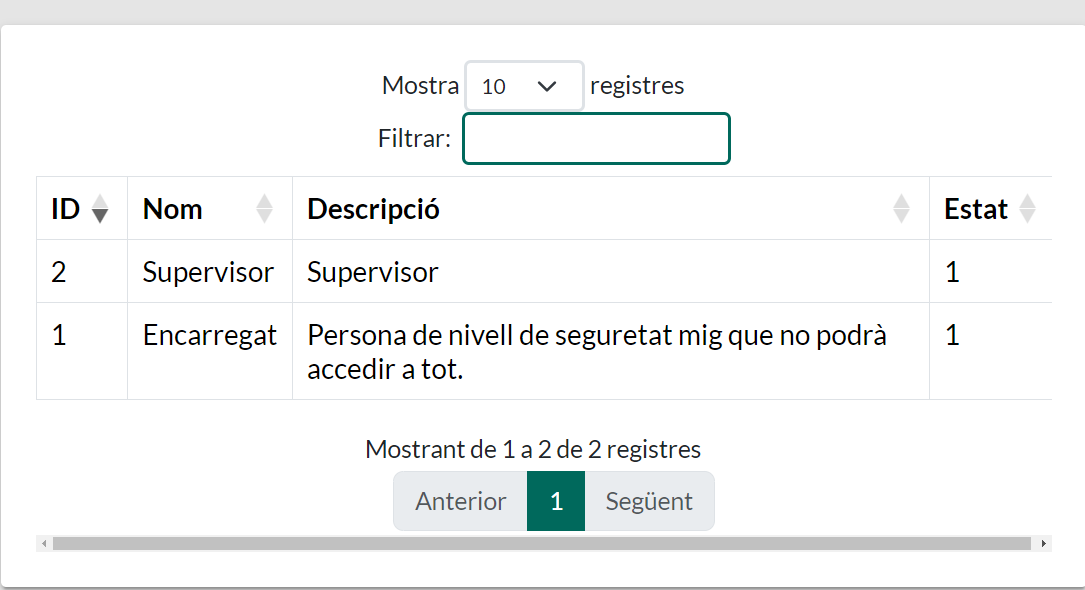
        "order":[[0,"desc"]]

    });

})

Les útlimes configuracions fan referencia a que siga reponsive, que mostre 10 files i que ordene numèricament.

I ara podem visualitzar la taula de forma correcta



A continuació, per a que siga més visual si un estat està o no actiu, es cambia el 1 o el 2 per un Actiu en color verd i un color rogi si esta inactiu. Per això, cree un bucle for que recorrega tots els rols i en cadascú d’ells canvie el valor.

//Ara per a manipular els estatus cree un bucle for

for($i=0;$i<count($arrayDades); $i++){

//ara cree una validació de si es 1 done un valor i si es 2 un altre

if($arrayDades[$i]['status'] == 1)

   {

//Si es 1 esta actiu i per tant, li donem el format verd

   $arrayDades[$i]['status'] = '<button class="btn btn-success">Actiu</button>';

                }

else{

//Si no es 1 esta inactiu i per tant, li donem el format roig

$arrayDades[$i]['status'] = '<button class="btn btn-danger">Inactiu</button>';

                }

    }

I a continuació, cree els botons per eliminar o modificar els rols (aquests son models trobats a https://getbootstrap.com/docs/4.0/components/buttons/)

                //ara creem els botos per modificar o eliminar els elements

$arrayDades[$i]['options'] = '<div class="text-center" style="white-space: nowrap;">

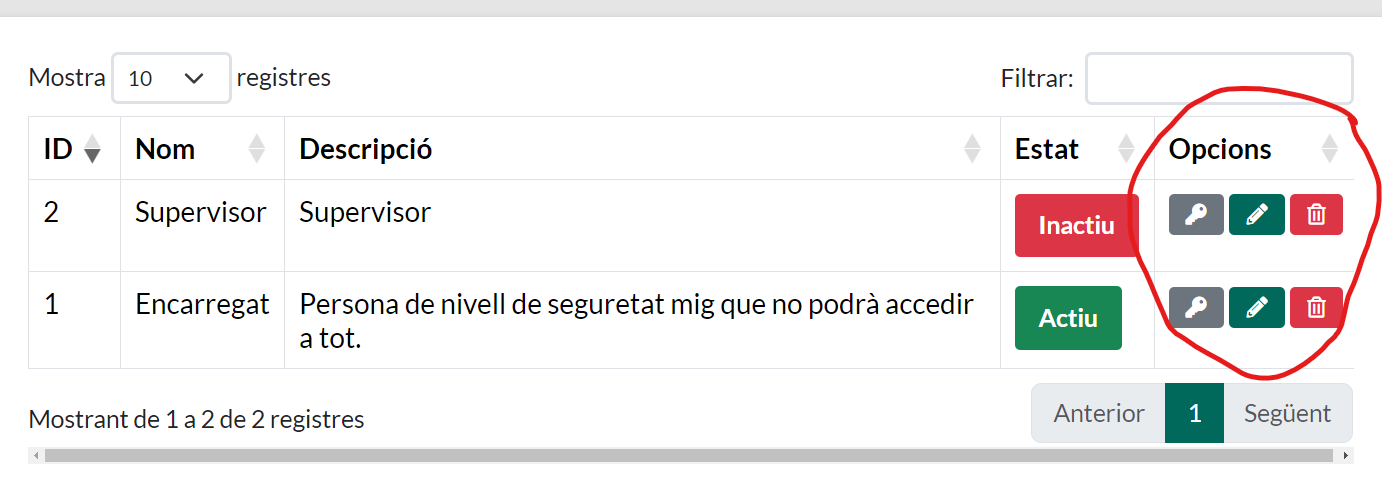
                <button class="btn btn-secondary btn-sm btnPermisosRol" rl="'.$arrayDades[$i]['idrol'].'" title="Permisos"><i class="fas fa-key"></i></button>

                <button class="btn btn-primary btn-sm btnEditRol" rl="'.$arrayDades[$i]['idrol'].'" title="Editar"><i class="fas fa-pencil-alt"></i></button>

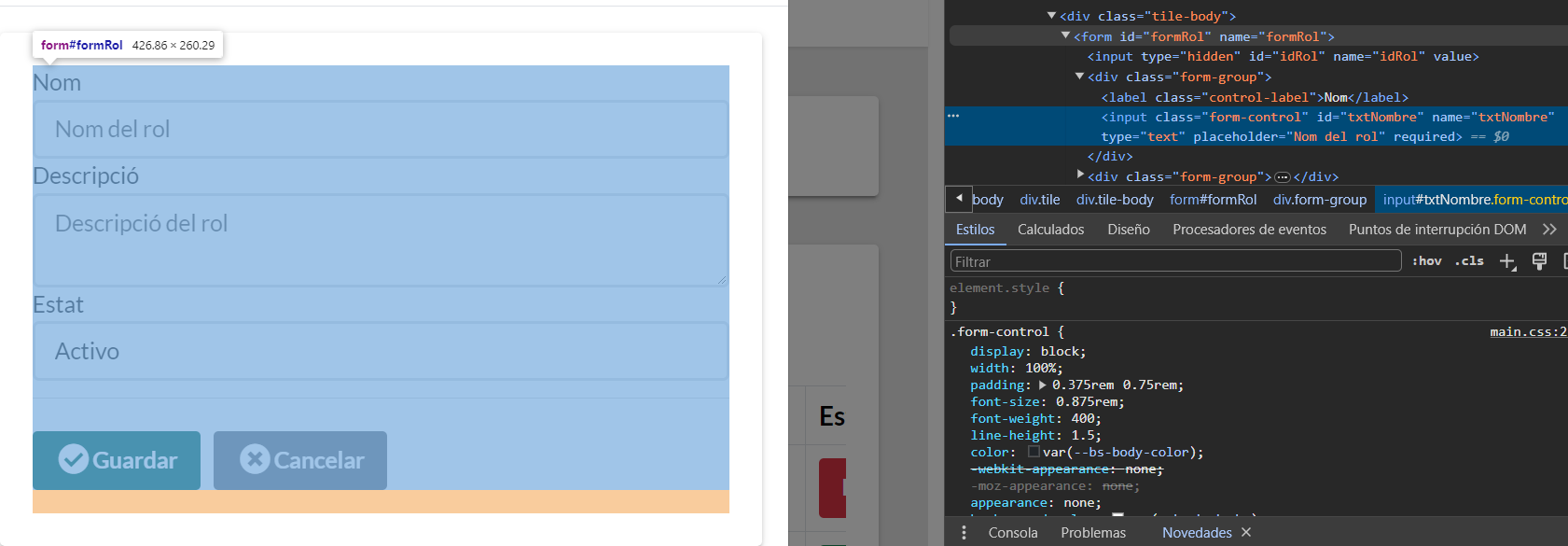
                <button class="btn btn-danger btn-sm btnDelRol" rl="'.$arrayDades[$i]['idrol'].'" title="Eliminar"><i class="far fa-trash-alt"></i></button>

                </div>';

Amb això he implementat els botons que es poden veure a continuació:



Ara el que faig es que el formulari de rols envie de forma correcta la informació a la base de dades. Per a aconseguir-ho, el que faig es crear una variable formRol, que marcarà un selector querySelector, ja que en aquest cas no es necessari utilitzar la llibreria JQuery. El que seleccionarà es el id formRol que es el id del formulari, com es pot veure a continuació:



Per evitar les possibles manipulacions del formulari en el HTML, he fet diferents mesures de seguretat, ja que si sols fiquem en el HTML requerided alguns usuaris podrien eliminar-ho fàcilment i enviar informació incompleta.

Per evitar això he fet el següent:

En primer lloc, un codi amb JS amb el que el color del boto guardar siga gris i estiga inhabilitat, mentre tots els camps no continguen informació tots els camps amb el següent codi:

//Seleccione el id del formulario formRol

var formRol = document.querySelector("#formRol");

    //Ara el que faig es capturar les variables nom, descripció i status

function canvisFormulari(e){

  var strNom = document.querySelector("#txtNombre").value;

  var strDescripcion = document.querySelector("#txtDescripcion").value;

  var intStatus = document.querySelector("#listStatus").value;

// Comprova si alguna de les variables és buida

if(strNom !== "" && strDescripcion !== "" && intStatus !== "") {

  // Si alguna condició no es compleix, activa el botó i canvia l'estil

  btnActionForm.disabled = false;

  btnActionForm.classList.remove('btn-secondary');

  btnActionForm.classList.add('btn-primary');

  } else {

  // Si es compleixen totes les condicions, desactiva el botó i restaura l'estil

  btnActionForm.disabled = true;

  btnActionForm.classList.remove('btn-primary');

  btnActionForm.classList.add('btn-secondary');

  }

}

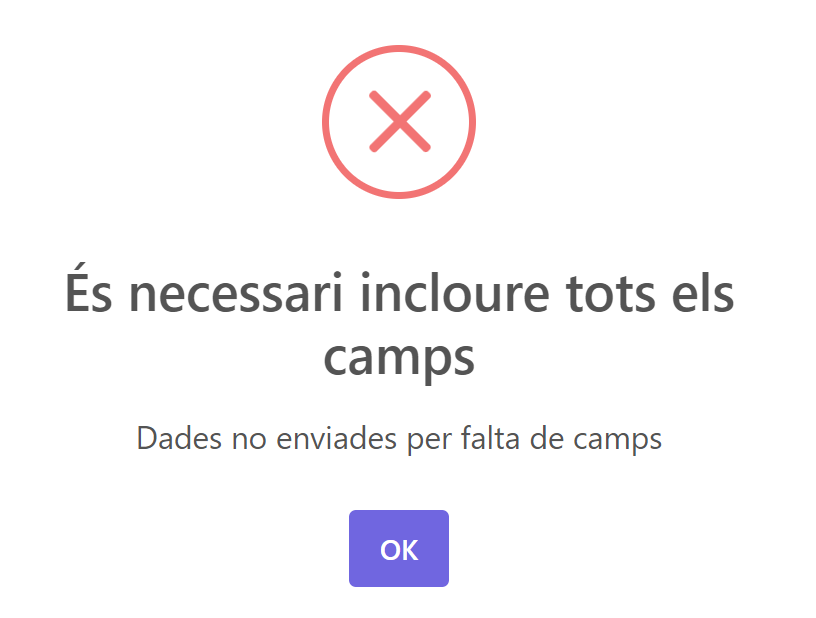
  // Crida la funció checkConditions quan es canvia algun dels camps

  formRol.addEventListener("input", canvisFormulari)

  formRol.addEventListener("click", canvisFormulari)

Però aquesta validació també te un problema, i es que es pot seguir manipulant el DOM. No obstant, millora la usabilitat, ja que el usuari sap que mentre no estiguen tots els camps omplits, el boto guardar no funcionarà.

I la mesura de seguretat real radica en que utilitzan els plugins de <https://sweetalert2.github.io/> indique al navegador amb un onsubmit, que si està buit algun dels camps envie el següent missatge:



## **6.2.** **Funcionalitat botons**

Ara el que faig es donar-li funcionalitat als botons per a que quan punxe el boto cancel·lar o la creuete de la cantonada superior dreta, es tanque la finestra emergent del formulari.

Per a aconseguir-ho incorporem data-bs-dismiss="modal" al nostre botonatge, que es un element de bootstrap que fa que directament es tanque sense necessitat de programar res.

### **6.2.1. Funcionalitat boto modificar**

Després en relació als botons de modificar, com son varios botons, els seleccione tots amb un querySelectAll y els recorreré amb un bucle foreach, per a escollir tots els elements de la classe btnEditRol, i els deixe a l’escolta dels click, i li dic, que en cas de fer click, que aplique el modal show de Bootstrap, com es pot veure en el següent codi:

function fntEditRol(){

  //seleccionem tots els elements de la classe btnEditRol

  var btnEditRol = document.querySelectorAll("#btnEditRol");

  //Recorrem tots els elements de la classe btnEditRol

  btnEditRol.forEach(function(btnEditRol){

    //Deixem a la escolta els elements dels click, i si fa click

    btnEditRol.addEventListener('click', function(){

      document.querySelector('#titleModal').innerHTML = "Actualitzar Rol";

      document.querySelector(".modal-header").classList.replace("headerRegister", "headerUpdate");

      document.querySelector("#btnActionForm").classList.replace("btn-primary", "btn-info");

      document.querySelector("#btnText").innerHTML="Actualizar";

      $('#modalFormRol').modal('show')

    })

  })

}

I li indique al navegador, que quan es carregue tota la web aleshores, carregue tota la funció

    window.addEventListener('load', function(){

      fntEditRol();

    },

        false)

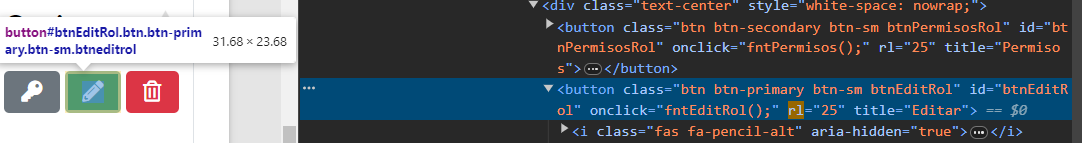
I com necessite que quan torne a punxar el boto de nou, apareixca el mateix que en el anterior, li dic a modal, que formatge l’estil.

Com es pot vore, s’intenta donar soport a internet a explorer, amb el següent condicional

      var request = (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject("Microsoft.XMLHTTP");

Com es pot vore en el codi, el primer que faig es modificar el valor del titol, per a que canvie el titol de la página emergent de Nou rol a Actualizar Rol, canviem el color de la capçalera de blau a verd i per a que reemplace el color del boto de guardar a actualizar.

A més, en aquesta funció capturem el id de cada element de la base de dades, ja que en el momento de generar-se els botons, em creat un element rl que el número del id conicidisca amb el del rl.



function fntEditRol(){

  //seleccionem tots els elements de la classe btnEditRol

  var btnEditRol = document.querySelectorAll("#btnEditRol");

  //Recorrem tots els elements de la classe btnEditRol

  btnEditRol.forEach(function(btnEditRol){

    //Deixem a la escolta els elements dels click, i si fa click

    btnEditRol.addEventListener('click', function(){

      document.querySelector('#titleModal').innerHTML = "Actualitzar Rol";

      document.querySelector(".modal-header").classList.replace("headerRegister", "headerUpdate");

      document.querySelector("#btnActionForm").classList.replace("btn-primary", "btn-info");

      document.querySelector("#btnText").innerHTML="Actualizar";

      //rl ho trobem en els botos, ja que l'atribut coinsideix amb el id

      var idrol = this.getAttribute("rl");

      //Això ho faig per donar soport a microsoft explorer;

      var request = (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject("Microsoft.XMLHTTP");

      var ajaxURL = base\_url+'Roles/getRol/'+idrol;

      //Amb get capturem la informació

      request.open("GET", ajaxURL, true);

      request.send();

      request.onreadystatechange = function()

      {

        //Comprobem que dona resposa Ok! i ha arribat la petició

        if(request.readyState == 4 && request.status == 200){

          var objData = JSON.parse(request.responseText);

          if(objData.status == true){

            document.querySelector("#idRol").value = objData.data.idrol;

            document.querySelector("#txtNombre").value = objData.data.nombrerol;

            document.querySelector("#txtDescripcion").value = objData.data.descripcion;

            if(objData.data.status == 1)

            {

              var optionSelect = '<option value="1" selected class="notBlock">Actiu</option>';

            }

            else

            {

              var optionSelect = '<option value="2" selected class="notBlock">Inactiu</option>';

            }

            var htmlSelect = `

            ${optionSelect}

            <option value="1">Actiu</option>

            <option value="2">Inactiu</option>

            `;

            document.querySelector("#listStatus").innerHTML = htmlSelect;

            $('#modalFormRol').modal('show');

          }

          else{

            Swal.fire({

              icon: "error",

              title: "objData.msg",

              text: "Per favor, recarrega la pàgina i torna a intentar-ho",

              footer: '<button type="button" class="btn btn-danger text-white" data-dismiss="modal">Cancel·lar</button>'

            });

          }

        }

      }

    })

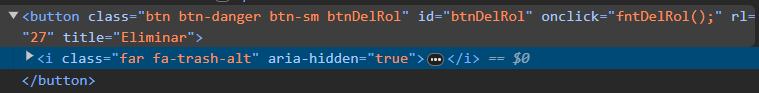
  })

}

### **6.2.2. Funcionalitat boto eliminar**

En aquest apartat explicaré com li he donat funcionalitat al botó d’eliminar que està en l’apartat administrador.

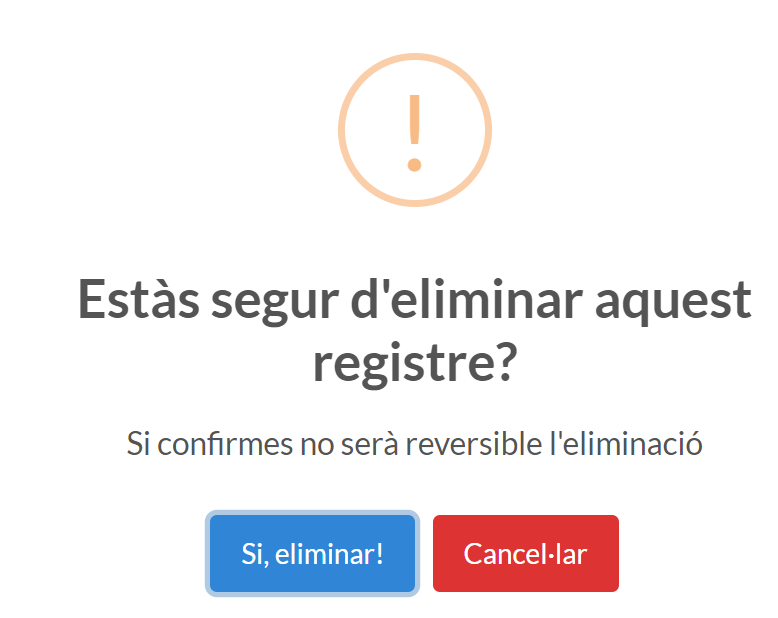
En primer lloc he creat una funció denominada fntDelRol en JavaScript. El primer que fem es capturar en una variable let tots els botons que tenen la classe btnDelRol, que contenen els botons amb aquesta imatge 



Per a capturar-los, he utilitzat el quereSelectorAll amb el que seleccione tots els elements amb la classe btnDelRol

let btnDelRol = document.querySelectorAll(".btnDelRol");

A continuació cree un bucle forEach en el que amb una funció anònima faig que tots els botons amb la imatge  estiguen a la escolta del click. Capture el rl, que no explicaré com, ja que s’explica en l’apartat anterior, i faig que amb el swal em done un avis de si estic segur d’eliminar el registre.



Si apretem el boto cancel·lar es tanca la pantalla emergent gracies a les classes de bootstrap, mentre que si apretem Si, eliminar!, salta el .then, que si es confirma, que s’ha eliminat el registre, i amb el formData li done una relació de que idrol es idrol en php. En acabant envie les dades amb el mètode POST que és més segur, i recarregue les funcions d’editar i eliminar, ja que si no es recarreguen, deixarien de funcionar en la pantalla.

  //Cree un buclq for per recorrer tots els botons

  btnDelRol.forEach(function(btnDelRol){

    //Cree un esdeveniment per a que estiga a l'escolta dels clicks

    btnDelRol.addEventListener("click", function(){

      var idrol = this.getAttribute("rl");

      //Ara utilitze un disseny de SweetAlert2 per a generar un avis d'alerta

      Swal.fire({

        title: "Estàs segur d'eliminar aquest registre?",

        text: "Si confirmes no serà reversible l'eliminació",

        icon: "warning",

        showCancelButton: true,

        confirmButtonColor: "#3085d6",

        cancelButtonColor: "#d33",

        confirmButtonText: "Si, eliminar!",

        cancelButtonText: "Cancel·lar"

      }).then((result) => {

        if (result.isConfirmed) {

          var formData = new FormData();

          formData.append('idrol', idrol);

          var request =  (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject ("Microsoft.XMLHTTP");

          var ajaxURL = base\_url+"Roles/delRol/";

          var strData = "idrol=" + idrol;

          request.open("POST", ajaxURL, true);

          //Indiquem com volem enviar les dades informació en https://developer.mozilla.org/es/docs/Web/HTML/Element/form#enctype

         ////request.setRequestHeader("content-type", "application/x-www-form-urlencoded");

          request.send(formData);

          request.onreadystatechange = function(){

            // El valor 4 de la propietat readyState indica que la sol·licitud ha finalitzat i la resposta està disponible.

            if(request.readyState == 4 && request.status == 200){

              var objData = request.responseText;

              Swal.fire({

                title: "Elimnat",

                text: "Aquest rol ha estat eliminat!",

                icon: "success"

              });

              //I recarreguem les funcions en el Datatable

              tableRoles.ajax.reload(function(){

                fntEditRol();

                fntDelRol();

              })

            }

          }

        }

      });

    })

  }

  )

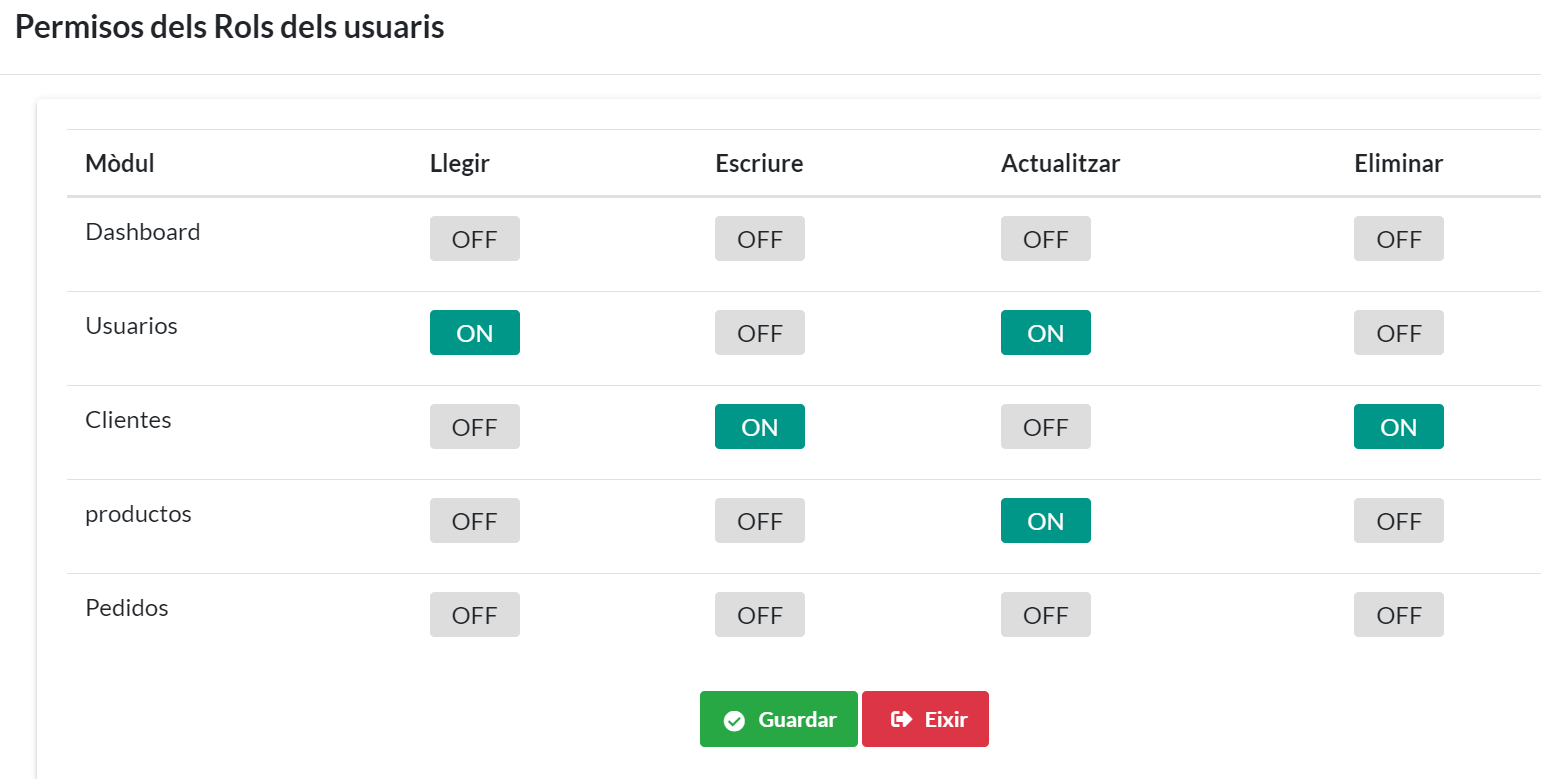
}

### **6.2.3. Funcionalitat boto permisos**

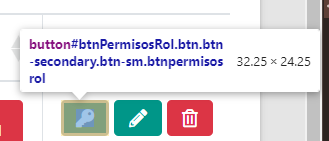
Amb aquesta funcionalitat, busque crear un panell intuïtiu on els usuaris puguen assignar diverses autoritzacions als diferents mòduls del meu projecte, com ara llegir, escriure, actualitzar o eliminar. L'objectiu és evitar que els usuaris puguin accedir a opcions no desitjades, com la capacitat d'eliminar codis de comandes o registres.

Aquesta funcionalitat es similar al estudiat a l’assignatura Desarrollo front-end con frameworks JavaScript, on en AngularJS, es protegeixen les rutes per a garantir la seguretat. No obstant això, en aquest cas, utilitzo PHP i JavaScript per a aconseguir aquesta funcionalitat.

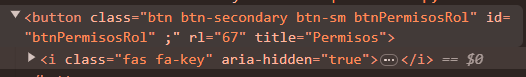
Per millorar l'experiència d'usuari, he dissenyat el panell utilitzant Bootstrap. El codi inclou les etiquetes data-toggle-on="ON" i data-toggle-off="OFF", que permeten als usuaris canviar entre les visualitzacions "ON" i "OFF" que es poden vore a continuació.



Quan es mostra aquest panell, es fa mitjançant un modal de Bootstrap controlat mitjançant JavaScript. Al botó se li assigna un event listener, que escolta els clics de l'usuari. Com es pot observar en el següent codi, el primer pas és recórrer els elements dins del modal per identificar els botons amb la classe btnPermisosRol, això o faig amb un bucle foreach, els quals son els diferents botons amb forma .



Això permet assignar un event listener a cadascun d'aquests botons. Quan un usuari fa clic en algun d'aquests botons, el codi captura l'ID del rol corresponent, que es el “rl” que trobem dins del codi button, com es pot apreciar a continuació:



Per a assegurar la compatibilitat amb navegadors antics, es realitza una conversió si és necessari, especialment per a casos d'Internet Explorer, ja que l'ús d'AJAX pot requerir un tractament diferent.

A continuació, el codi captura la URL on es troba la funcionalitat getPermisosRol i envia la informació mitjançant el mètode POST. Si la resposta de la petició és satisfactòria (status 200 i readyState 4), es mostra el modal per a assignar els diferents permisos als usuaris.

Amb aquest boto el que vull aconseguir es un panell intuïtiu en el que es puguen otorgar funcionalitats de llegir, escriure actualitzar o eliminar, els diferents mòduls del meu projecte. Amb aço el que vull aconseguir es que un usuari no tinga opció de eliminar codi de pedidos, ni registres ni tan sols puga llegir-los. Es paregut al que es fa en el màster amb angular, per protegir rutes, però amb PHP i JavaScript.

 function fntPermisos() {

    let btnPermisosRol = document.querySelectorAll(".btnPermisosRol");

    // Itinerem respecte a cada boto

    btnPermisosRol.forEach(function(btnPermisosRol) {

      // Agregem el event al listener per a aquest boto

      btnPermisosRol.addEventListener('click', function() {

      let  idrol = this.getAttribute("rl");

      let  request = (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject('Microsoft.XMLHTTP');

      let  ajaxUrl = base\_url + 'Permisos/getPermisosRol/' + idrol;

        //Obrim la peticio indiquem que el mètode va a ser el tipo GET i a la direcció anterior,

      request.open("POST", ajaxUrl, true);

      //Enviem la peticio

      request.send();

      request.onreadystatechange = function() {

        if (request.status == 200 && request.readyState == 4) {

          console.log(request.responseText)

          document.querySelector('#contentAjax').innerHTML = request.responseText;

          $('.modalPermisos').modal('show');

          document.querySelector('#formPermisos').addEventListener('submit',fntSavePermisos, false )

        }

      }

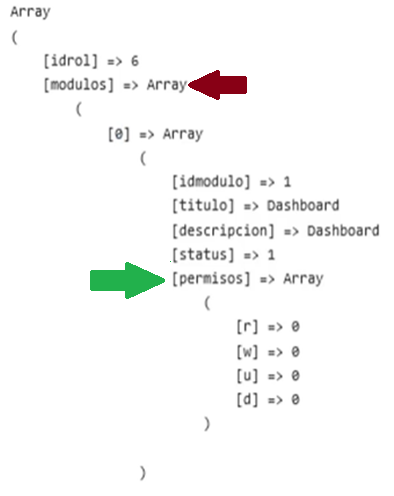
    })

    });

  }

En el codi anterior, es pot observar que al enviar el formulari (submit), es declara un event que crida la funció fntSavePermisos.

Aquesta funció té com a objectiu capturar i emmagatzemar a la base de dades les diferents autoritzacions associades a cada rol. A continuació, es presenta un exemple:



Com es pot notar, dins de l'array permisos, hi ha quatre camps. Cada un d'ells indica a la base de dades si l'usuari té autorització per llegir, escriure, actualitzar o eliminar els elements de cada secció.

Al analitzar el codi JavaScript, l'aspecte més rellevant és el mètode setPermisos, que governa com s'emmagatzema la informació a la base de dades. La resta del codi es considera explicada en els apartats anteriors, ja que segueixen un procés similar.

function fntSavePermisos(e){

  e.preventDefault();

  var request = (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject('Microsoft.XMLHTTP');

  //En el controlador s'ha de crear setPermisos, per a que s'execute aquesta info!

  var ajaxUrl = base\_url+'/Permisos/setPermisos';

  var formElement = document.querySelector("#formPermisos");

  var formData = new FormData(formElement);

  request.open("POST",ajaxUrl,true);

  request.send(formData);

  console.log(formData)

  request.onreadystatechange = function(){

      if(request.readyState == 4 && request.status == 200){

          var objData = JSON.parse(request.responseText);

          if(objData.status)

          {

            Swal.fire({

              icon: "success",

              title: objData.msg,

              text: "Permisos usuaris actualitzats",

              footer: 'Ja podrà veure els nous permisos de cada usuari',

              showConfirmButton: false,

              timer: 1500

            });

          }else{

            Swal.fire({

              icon: "error",

              title: objData.msg,

              text: "No ha estat possible actualitzar els permisos",

              footer: 'Per favor, torne a intentar-ho passats uns minuts'

            })

          }

      }

  }

}

El codi setPermisos es troba a Controllers/Permisos i està escrit en PHP. Aquest codi té la finalitat de capturar la informació enviada a través del mètode POST. Primer, obté el número d'identificació del rol, assegurant-se que sigui un número enter amb la funció intval.

Seguidament, captura la sub-array de mòduls i, finalment, itera sobre cada categoria per a assignar els diferents permisos de lectura, escriptura, actualització i eliminació.

Un cop s'han recollit tots els permisos, utilitza el model insertPermisos per inserir l'identificador del rol, l'identificador del mòdul i els permisos associats. Segons el resultat obtingut, retorna un missatge que indica si l'actualització s'ha realitzat correctament o si s'ha produït un error.

public function setPermisos()

         {

            //Cree un if per verificar que realment s'ha enviat per post la informació

             if($\_POST)

             {

                    //M'assegure que siga un ParseInt el valor de idrol

                 $intIdrol = intval($\_POST['idrol']);

                 //capture del array del post el subarray que conte modulos

                 $modulos = $\_POST['modulos'];

                 // de \Models\PermisosModel.php prove deletePermisos

                 $this->model->deletePermisos($intIdrol);

                 //Cree un for per recorrer tot el array com si fos un for in per donar-li valor 0 o 1 a cada camp

                 foreach ($modulos as $modulo) {

                     $idModulo = $modulo['idmodulo'];

                     $r = empty($modulo['r']) ? 0 : 1;

                     $w = empty($modulo['w']) ? 0 : 1;

                     $u = empty($modulo['u']) ? 0 : 1;

                     $d = empty($modulo['d']) ? 0 : 1;

                     //Retorne els valors a una variable amb la funcio que trobem a \Models\PermisosModel.php

                     $requestPermiso = $this->model->insertPermisos($intIdrol, $idModulo, $r, $w, $u, $d);

                 }

                 //Ara, si el valor de request es major de 0 vol dir que te alguna cosa, i per tant, s'ha capturat de forma correcta el valor

                 if($requestPermiso > 0)

                 {

                     $arrResponse = array('status' => true, 'msg' => 'Permisos assignats correctament.');

                 }

                 //Pel contrari vol dir que hi ha un error

                 else{

                     $arrResponse = array("status" => false, "msg" => 'No ha estat possible assignar els permisos, torne ha intentar-ho');

                 }

                 //Capture les dades en format JSON;

                 echo json\_encode($arrResponse,JSON\_UNESCAPED\_UNICODE);

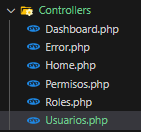
             }

             die();

         }

# **7. Sistema d’ususaris**

En aquest apartat del TFM com he crear el sistema d’usuaris per gestionar-ho. Per a aconseguir-ho el primer que he de fer es crear el controlador usuarios, en la carpeta Controllers.



En Usuarios php, per a poder gestionar la secció d’usuaris, implemente el següent codi que es molt paregut al del home, i on definiré la classe Usuari per a seguir utilitzant el meu patró MVC

class Usuarios extends Controllers {

        public function \_\_construct(){

            //utilitze parent per accedir i executar el mètode constructor de la nostra class

           parent::\_\_construct();

        }

        public function Usuarios()

        {

           //data va a ser un array que continga tota la informació de la pàgina web.

           $data['page\_tag']='Usuarios'; //es el nom que apareixerà al costat del favicon

           $data['page\_title'] = 'Usuarios <small> Tenda Virtual</small>'; //es el titol del meta que tindrà aquesta url

           $data['page\_name'] = 'usuarios';//nom de la seccio

            // cridem per a la pàgina principal la vista home;

           $this->views ->getViews($this,'usuarios',$data);

        }

    }

Continuant amb la gestió usuaris, vaig a la carpeta wiews i cree una subcarpeta, per a gestionar la vista de Usuarios.I alhora, en eixa subcarpeta cree l’arxiu usuarios.php



Una vegada creada la vista, copie el format HTML de rols, i el pegue, i en acabant adapte la taula als camps d’usuaris i canvie el id a tableUsuarios



A continuació vaig a Modals i cree un modal per als usuaris, i com he fet en el HTML, ara el modal em copie el que tenia fet per a rols, ja que la base es la mateixa.

Per a adaptar-lo, el primer que faig es modificar el id per a que siga modalFormUsuarios.



Per a aquest formulari li done la classe de boostrap, form-horizontal que podem trobar aquí <https://getbootstrap.com/docs/5.0/forms/layout/#horizontal-form>.

També modifique el id del primer div del formulari:



Per als camps del formulari he utiltizat boostrap per a la seua implementació, utilitzant un formulari de fila, amb un form-group de 6 columnes i li he donat un id anomenat txtIdentificacion.

<div class="form-row">

 <div class="form-group col-md-6">

   <label for="txtIdentificacion">Identificació</label>

   <input type="text" class="form-control" id="txtIdentificacion" name="txtIdentificacion" requerided="">

 </div>

</div>

Una vegada creat tot el formulari, li he donat una validació en JS ja que no es suficient requerided en HTML, al poder ser manipulat pels usuaris de forma fàcil amb uns coneixements bàsics.

La validació en JavaScript,

El primer que he fet es crear una variable on diga que en principi el formulari es vàlid,

let valid = true;

Per tal d’eliminar espais en els valors he utiltizat trim()

const value = field.value.trim();

En relació a la validació, la primera que he fet es que els camps Identificació han de tenir una longitud mínima de 5 caràcters

      if (identificacion.value.length < 5) {

          identificacionError.textContent = 'La longitud mínima ha de ser de 5 caràcters.';

          valid = false;

      } else {

          identificacionError.textContent = '';

      }

I en relació als camps , Nom i cognoms he fet el mateix, però el número mínim de caràcters serà de 3.

Que el telèfon sols continga números i la seua longitud siga de 9 caràcters

      if (telefono.value.length !== 9 || !/^[0-9]{9}$/.test(telefono.value)) {

          telefonoError.textContent = 'El format ha de ser 6XXXXXXXX';

          valid = false;

      } else {

          telefonoError.textContent = '';

      }

A més el email ha de tenir un format de email correcte

      if (!/^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/.test(email.value)) {

          emailError.textContent = 'El format ha de ser d\'email';

          valid = false;

      } else {

          emailError.textContent = '';

      }

I per últim, per assegurar-me de que el llistat dels estats te un seleccionat he afegit aquesta validació:

else if (field.id === 'listStatus') {

        valid = value === '1' || value === '2';

      }

Si assoleix totes les proves s’aplicarà la classe is-valid i sinó, la classe is-invalid

      if (!status.value) {

          statusError.textContent = 'Es necessari seleccionar un dels dos estats';

          valid = false;

      } else {

          statusError.textContent = '';

      }

I si tots els camps son vàlids, el que faig es canviar es el color del botó per a que quan tots els camps siguen vàlids el boto tinga un color blau, i quan no, un color gris amb btn-secundary de Boostrap.

      if (valid) {

          btnActionForm.disabled = false;

          btnActionForm.classList.replace("btn-secondary", "btn-primary");

      } else {

          btnActionForm.disabled = true;

          btnActionForm.classList.replace( "btn-primary", "btn-secondary");

      }

En relació a al funció modal per a que aparega el formulari, el que he fet es crear un arxiu JavaScript, anomenat funtions\_usarios.js i crear la funció anemnada openModal(), i el codi és el següent:

  function openModal()

  {

    document.querySelector('#idUsuario').value="";

    document.querySelector(".modal-header").classList.replace("headerUpdate","headerRegister")

    document.querySelector("#btnActionForm").classList.replace("btn-info", "btn-primary");

    document.querySelector("btnText").innerHTML ="Guardar";

    document.querySelector("#titleModal").innerHTML ="Nou Usuari";

    document.querySelector("#formUsuario").requestFullscreen();

    $("#modalFormUsuario").modal("show");

    }

En primer lloc seleccione el id idUsuario i li done un valor buit. En acabant ens referim al modal-header i la substituisc, en cas de que siga headerUpdate per headerRegister, que es la classe del formulari per registrar-se



En acabant seleccionem el boto amb la classe btnActionForm



I el que faig es substituir, si existeix, la classe bt-infor per bt-primary. En acabant, per a substituir el títol li dic que el contingut de la classe titleModal serà Nou Usuari. En acabat resetege els camps per a que estiguen buits amb reset()

document.querySelector("#formUsuario").reset();

i mostre el formulari amb modal(‘show’)

$("#modalFormUsuario").modal("show");

Com tant en el modal de rols com el de usuaris, la funció es diu obrirmodal() el que faig indicar-li al navegador que carregue el codi JavaScript de la pagina corresponent

    <?php if($data['page\_name'] == "Rols"){

      ?>

        <script src="<?= media(); ?>/js/funtions\_roles.js "></script>

      <?php } ?>

    <?php if($data['page\_name'] == "usuarios"){

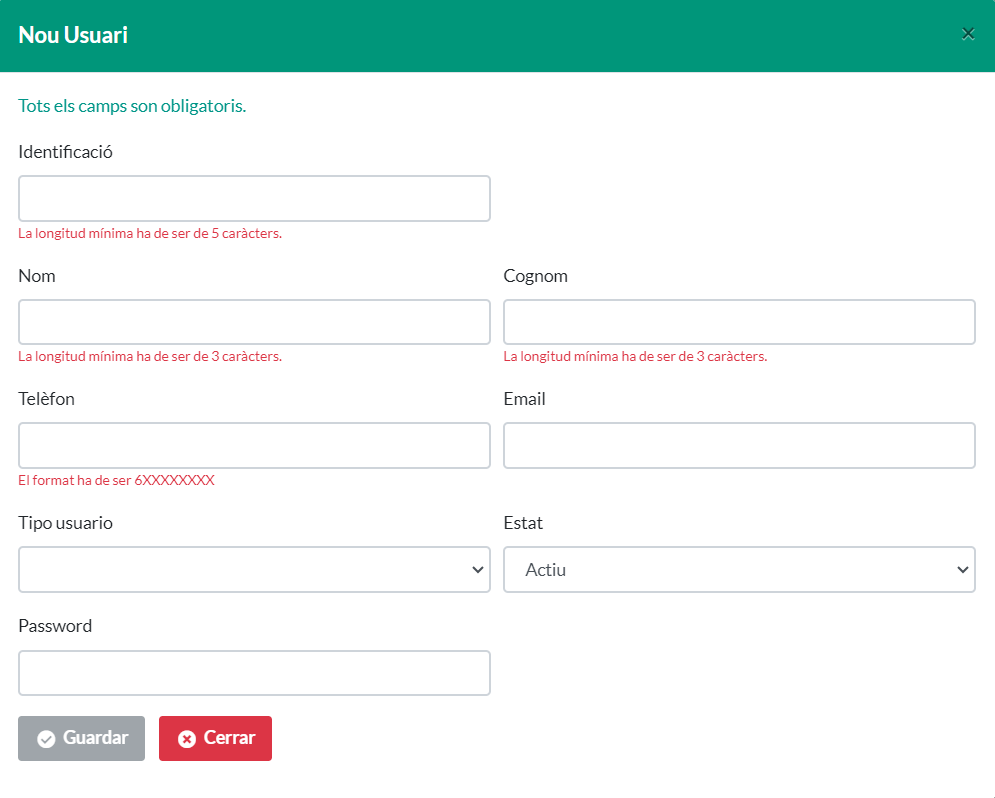
    ?>

        <script src="<?= media(); ?>/js/funtions\_usarios.js "></script>

    <?php } ?>

Això també farà que el rendiment de la pàgina web siga més òptim, ja que sols carregarà aquells arxius que siguen necessaris

La visualització actual del formulari es la següent:



Ara el que implemente es que el tipo d’usuari carregue directament de la nostra base de dades. El primer que faig es en Contorllers/Roles, implementar la funció getSelectRoles(), de tal forma que primer cree una variable buida anomenada, htmlOptions

        $htmlOptions = "";

Despres cride al mètode selectRoles i li dic que, si el numero de tipo de rols, es major a 0, es a dir, que existeixen rols, aleshores vaja un a un, buscant-los, i que cree un html amb option i el valor de dit rol

        $arrData = $this->model->selectRoles();

        if(count($arrData) >0){

            for($i =0; $i< count($arrData); $i++){

                $htmlOptions .= '<option value= "'.$arrData[$i]['idrol'].'">'.$arrData[$i]['nombrerol'].'</option>';

            }

        }

En acabant imprimisc el rol amb echo. La funció es queda com es pot vore a continuació:

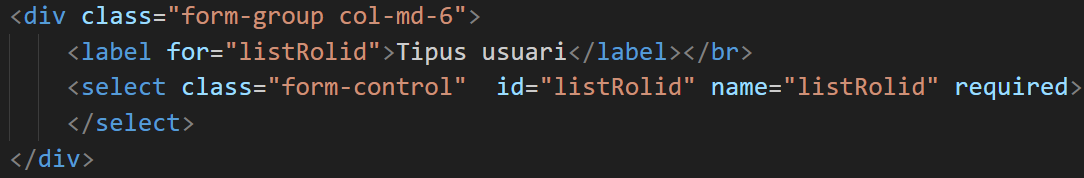


Una vegada creat el mètode getSelectRoles, el següent pas és crear una funció anomenada tipoRolsUsuaris dins del fitxer functions\_roles.js. En ella, mitjançat una promesa, cride en primer lloc el metodo anterior amb

var ajaxURL = base\_url + 'Roles/getSelectRoles';

Després comprovaré si el navegador es un internet explorer o no. Si no ho és, aplique XMLHttpRequest, mentre que si es un internet explorer, apicaré Microsoft.XMLHTTP.

En acabant capture les dades amb el mètode get en la direcció que es troba ajaxURL. En acabant envie la informació al servidor amb request.send(). i després li dic que si el readyState es 4 i per tatla sol·licitud s’ha completat i el status es igual a 200, es a dir la resposta es correcta, aleshores en la linea de codi amb id listRolid, visualitze el llistat dels rols amb select3.



Finalment el mètode tipoRolsUsuaris, ha quedat aixina:

function tipoRolsUsuaris() {

    return new Promise((resolve, reject) => {

        var ajaxURL = base\_url + 'Roles/getSelectRoles';

        var request = (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject('Microsoft.XMLHTTP');

        request.open("GET", ajaxURL, true);

        request.send();

        request.onreadystatechange = function() {

            if (request.readyState === 4) {

                if (request.status === 200) {

                    document.querySelector('#listRolid').innerHTML = request.responseText;

                    $('#listRolid').select3();

                    resolve(request.responseText);

                } else {

                    reject(new Error("Error en la solicitud: " + request.statusText));

                }

            }

        }

    });

}

## **7.1. Enviament de dades des del formulari de registre d’usuaris**

El primer que faig es un escoltador d’events, per a que observe qualsevol canvi en el DOM amb

document.addEventListener("DOMContentLoaded", function()

A continuació cree una variable per a que em seleccione el formulari.

let formUsuario = document.querySelector("#formUsuario");

Després, utilitzant l'event onsubmit del formulari, faig un preventDefault per a que no es recarregue la pàgina, i així poder capturar les dades:

    let formUsuario = document.querySelector("#formUsuario");

    formUsuario.onsubmit = function(e){

e.preventDefault();

A continuació, seleccione els diferents camps on estàn els formularis amb

        //Capture el valor dels diferents camps

        let strIdentifcacion = document.querySelector("#txtIdentificacion").value;

        let strNombre = document.querySelector("#txtNombre").value;

        let strApellido = document.querySelector("#txtApellido").value;

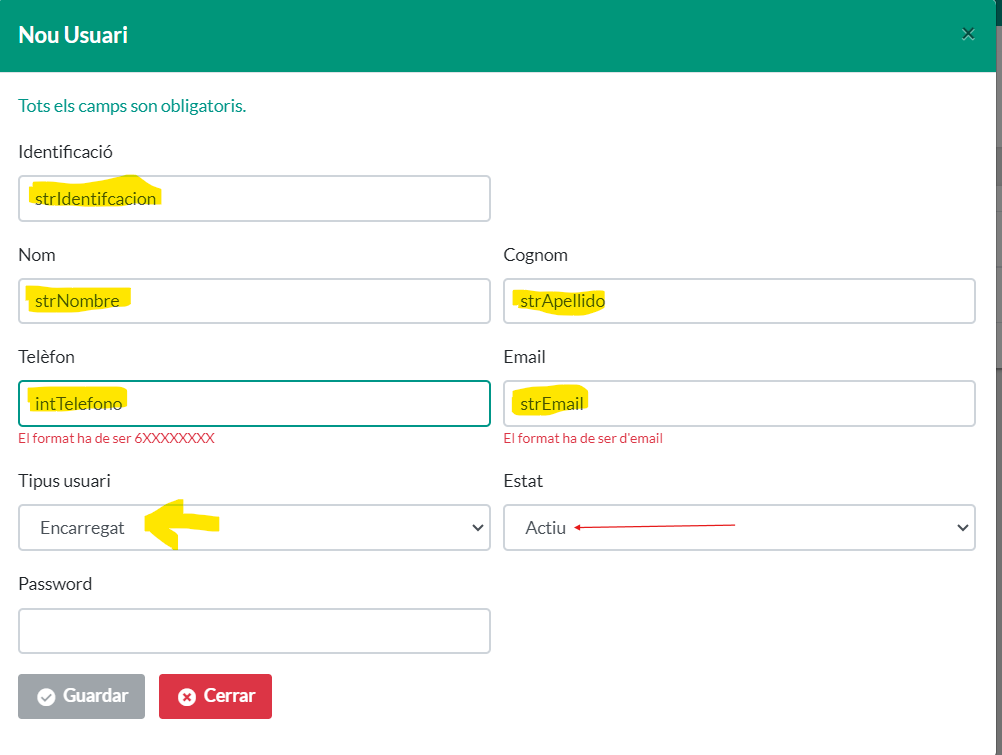
        let strEmail = document.querySelector("#txtEmail").value;

        let intTelefono = document.querySelector("#txtTelefono").value;

        let intTipousuario = document.querySelector("#listRolid").value;

        let strPassword = document.querySelector("#txtPassword").value;

visualment els camps capturats serien:



A banda de la validació explicada anteriorment, li faig un alert per a advertir que tots els camps son obligatoris, i que en cas de que

//Validació

        if(strIdentifcacion ==''|| strNombre == '' || strApellido == '' || strEmail

== '' || intTelefono == '' || intTipousuario == ''){

            Swal.fire({

                icon: "error",

                title: "Atenció!!!",

                text: "Tots els camps son obligatoris",

              });

            return false;

        }

Amb la variable ajaxURL capture la url on es troba el mètode setUsuario:

        let ajaxURL = base\_url + 'Usuarios/setUsuario';

El següent pas es verificar quin serà comprovar el navegador

let request = (window.XMLHttpRequest) ? new XMLHttpRequest() : new ActiveXObject('Microsoft.XMLHTTP');

I el següent pas es capturar les dades amb un formData per a poder recopilar les dades del formulari i poder enviar-les a la base de dades, mitjançant el mètode request.open i enviant-les amb request.send.

        let formData = new FormData(formUsuario)

        request.open("POST", ajaxURL, true);

        request.send(formData);

A continuació es comprova que la informació s’ha enviat de forma correcta amb

if (request.readyState == 4 && request.status == 200)

I transforme la informació a JSON

let objData = JSON.parse(request.responseText);

Amb això el que faig es capturar el estat que em retorna la funció setUusario de l’apartat Controllers Usuario el qual controla el retorn dels estats.

if ($request\_user === 'exist') {

     $arrResponse = array('status' => false, 'msg' => 'Atenció! El correu electrònic o la identificació ja existeixen!');

                    }

else if ($request\_user > 0) {

     $arrResponse = array("status" => true, 'msg' => 'Dades guardades correctament');

                    }

else {

     $arrResponse = array('status' => false, 'msg' => 'No ha estat possible emmagatzemar les dades!');

                    }

                } catch (Exception $e) {

   $arrResponse = array('status' => false, 'msg' => $e->getMessage());

                }

Per a determinar si ja està registrada la identificació o el email, en el Models en l’arxiu UsuariosModel.php he implementat en la classe UsuariosModel. En està classe l’he extingut a les meues funcions de sql.

El primer que he fet es crear una execució de si ja existeix en la base de dades usuario el email o la identificació amb SELECT \* from

$sql = "SELECT \* FROM usuario WHERE email = '{$this->strEmail}' OR identificacio = '{$this->strIdentificacion}'"

I ara comprovaré si en tota la base de dades es compleix aquest paràmetre

$request = $this->select\_all($sql);

I amb un if analitze si el resultat anterior retorna un valor buit, que implicaria que no s’ha trobat ni el emial ni el identificador en tota la base de dades, insertaré les dades del formulari en la nostra base de dades amb el comandament:

if(empty($request)){

            $query\_insert = "INSERT INTO usuario(identificacio, nom, cognoms, telefono, email, rolid, estat, contrasenya) VALUES(?,?,?,?,?,?,?,?)";

            $arrData = array(

                $this->strIdentificacion,

                $this->strNombre,

                $this->strApellido,

                $this->intTelefono,

                $this->strEmail,

                $this->intTipoId,

                $this->intStatus,

                $this->strPassword

            );

            $request\_insert = $this->insert($query\_insert, $arrData);

            $return = $request\_insert;

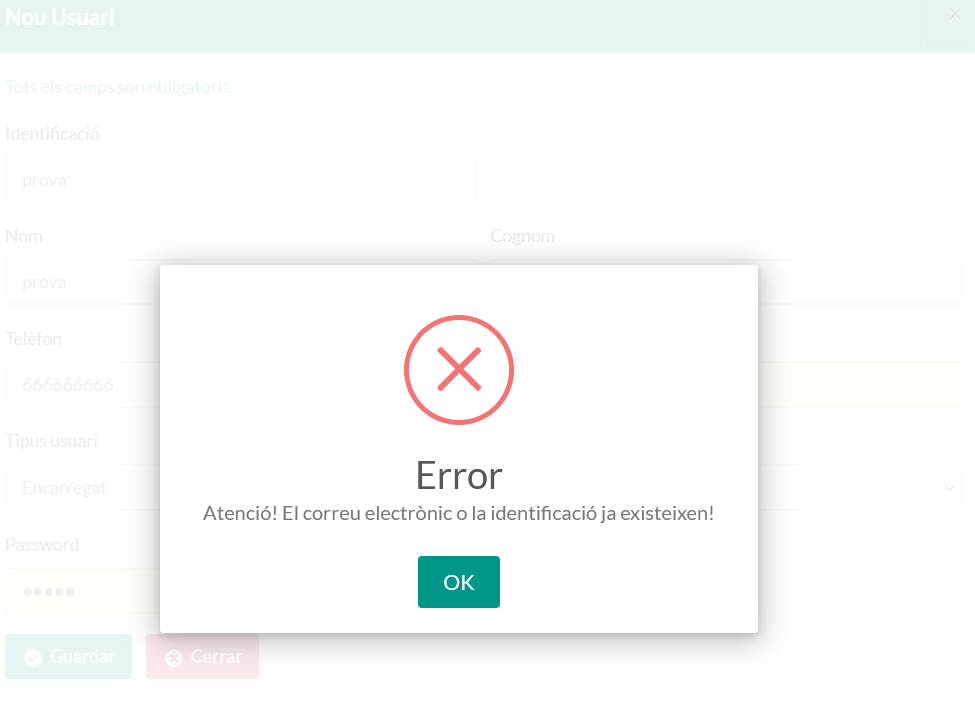
        }

I si no està buit voldrà dir que ja existeix o el email o la identificació i per tant retorna exist. Este exist l’aplique en el controller de Usuarios.php,

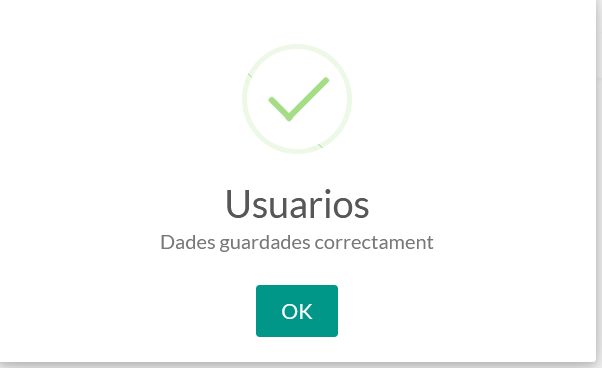
if ($request\_user === 'exist') {

   $arrResponse = array('status' => false, 'msg' => 'Atenció! El correu electrònic o la identificació ja existeixen!');

                    }



En canvi si no està registrat ni l’usuari ni la identificació



Una vegada obtinguda aquesta informació, el següent pas és que aquesta informació es reflexe en la taula.