

# POLICLOUD



JAVIER PRIMA MARGIOTTA

2º DAW      2021/2022



**FAMÍLIA INFORMÀTICA I COMUNICACIONS**



IES LA VEREDA - SANTI GIMENO

## ÍNDICE

1.....	INTRODUCCIÓN	3
1.1 Módulos a los que aplica.....		3
1.2 Breve descripción del proyecto.....		4
2.....	ESTUDIO PREVIO	4
2.1 Estudio de soluciones existentes .....		4
3.....	PLAN DE TRABAJO	5
4.....	DISEÑO	5
4.1 Diseño General .....		6
4.2 Diseño Detallado .....		6
4.2.1 Página de Inicio .....		7
4.3.2 Página de Registro de Usuario .....		12
4.3.3 Página de Perfil de Usuario .....		13
4.3.4 Página de Publicación de Usuario .....		15
5.....	IMPLEMENTACIÓN	18
5.1 Herramientas de servidor .....		18
5.2 Diagrama Entidad Relación .....		19
5.3 Lenguajes utilizados .....		19
5.3.1 IDE .....		20
5.3.2 Patrón de arquitectura.....		20
5.3.3 Framework .....		20
Complementos para Symfony.....		21
6.....	DESARROLLO DE LA APLICACIÓN	21
6.1 ÁRBOL DE FICHEROS.....		22
6.2 FUNCIONALIDAD .....		22
6.2.1 Frontend.....		22
6.2.2 Backend .....		23
Inicio de Sesión.....		23
Perfil .....		25
7.....	LEYENDA DE IMÁGENES	28

8.....	BIBLIOGRAFÍA Y ANEXOS
.....	29

## 1. INTRODUCCIÓN

### 1.1 Módulos a los que aplica

#### Base de Datos

- Gestionar información interpretando su diseño lógico
- La programación de procedimientos almacenados

#### Desarrollo Web en Entorno Cliente

- Uso de AJAX

#### Desarrollo Web en Entorno Servidor

- Insertar código en páginas web
- Programación basada en lenguajes de marcas con código embebido
- Generar de manera dinámica páginas web
- La utilización de técnicas de acceso a datos
- Programación de servicios web
- Generación dinámica de páginas web interactivas

#### Desarrollo Web en Entorno Cliente

- Preparar archivos multimedia para la Web, analizando sus características y manejando herramientas específicas
- Planificar la creación de una interfaz web valorando y aplicando especificaciones de diseño

## 1.2 Breve descripción del proyecto

El proyecto PoliCloud consiste en una aplicación web que permite registrar contenido multimedia de los usuarios, para posteriormente ser consultado por usuarios registrados y visitantes y valorar el contenido por el mismo usuario y/u otros usuarios registrados. Para cada publicación se registran cinco elementos los cuales tres son necesarios para registrar la publicación.

### Objetivos y Requisitos

- Acceso para usuarios registrados y visitantes
- Registrar contenido variado de una manera sencilla
- Permitir el registro y consulta multimedia desde diferentes rutas

## 2. ESTUDIO PREVIO

PoliCloud se usaría como idea de red social que permita difundir contenido publicado por los creadores y una valoración por el resto de los usuarios a modo de “me gusta”. En esta aplicación se iría agregando nuevas funcionalidades para una mayor comodidad hacia los usuarios y para sus necesidades a la hora navegar.

### 2.1 Estudio de soluciones existentes

Algunas otras aplicaciones las cuales tienen una idea similar a mi proyecto:



Estas aplicaciones web registran principalmente imágenes, pero también permiten audio, vídeo, vectoriales y más. Son completas en cuanto funcionalidades y apartados y disponen de mucho contenido lo cual es bastante favorable para los usuarios que estén buscando un tipo de información concreta. Una funcionalidad interesante es que permiten escoger la resolución de las imágenes a la hora de descargarlas y son de muy buena calidad.

Ambas aplicaciones web disponen de contenido registrado en el dominio público según Licencias Creative Commons / Creative Commons CC0. CC0 permite a los

creadores y propietarios de contenidos con derechos de autor a renunciar a ellos para que otros puedan usarlos libremente para cualquier propósito sin restricción.

### 3. PLAN DE TRABAJO



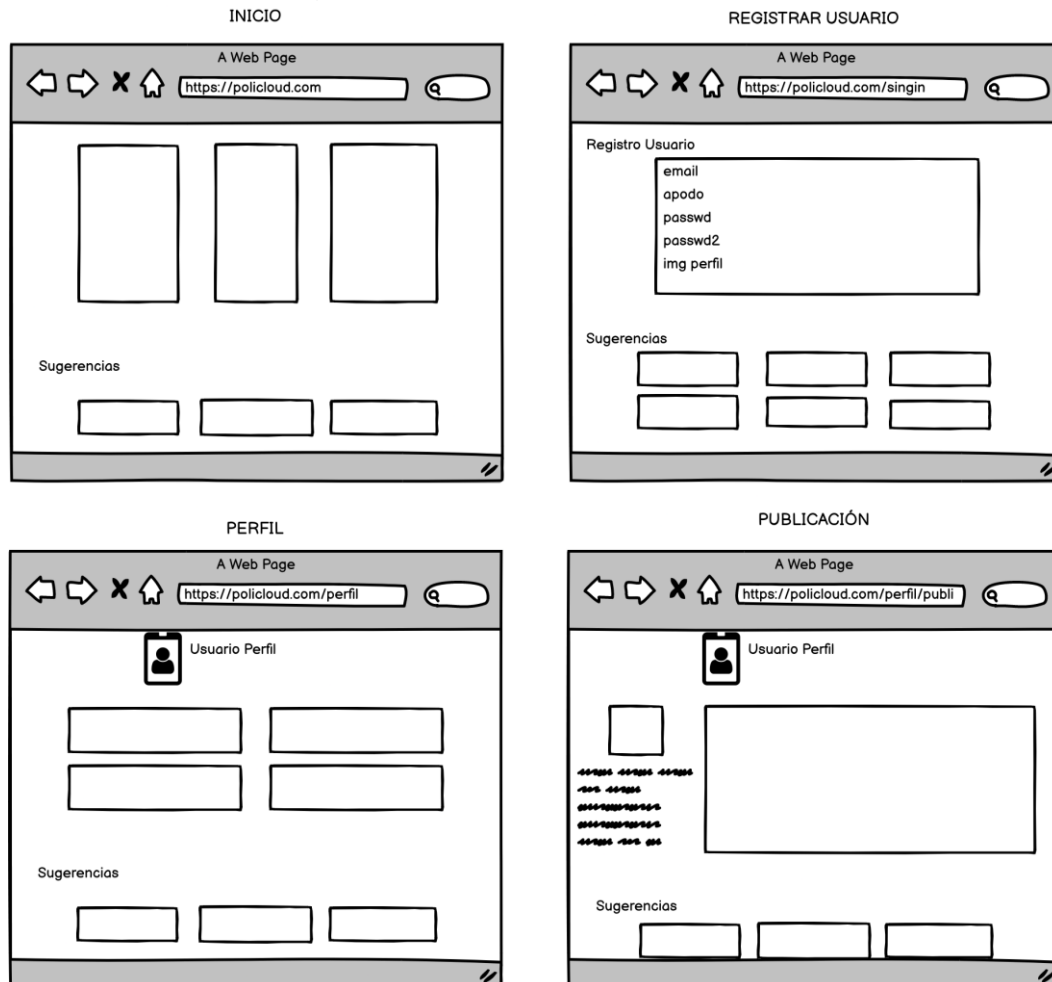
### 4. DISEÑO

La aplicación web PoliCloud está compuesta por diferentes pantallas de navegación las cuales permiten poder ejecutar todas las funcionalidades.

## 4.1 Diseño General

*Inicio, registro de usuario, perfil de usuario, publicación del usuario.*

### *Páginas que estructuran la aplicación*



2- Diseño de las páginas de PoliCloud

Todas las páginas están conectadas mediante enlaces y botones distribuidos en elementos, excepto publicación que solamente se puede acceder desde la página perfil.

## 4.2 Diseño Detallado

En todas las páginas de la aplicación web se muestra una cabecera y un pie de página más un apartado de publicaciones sugeridas.

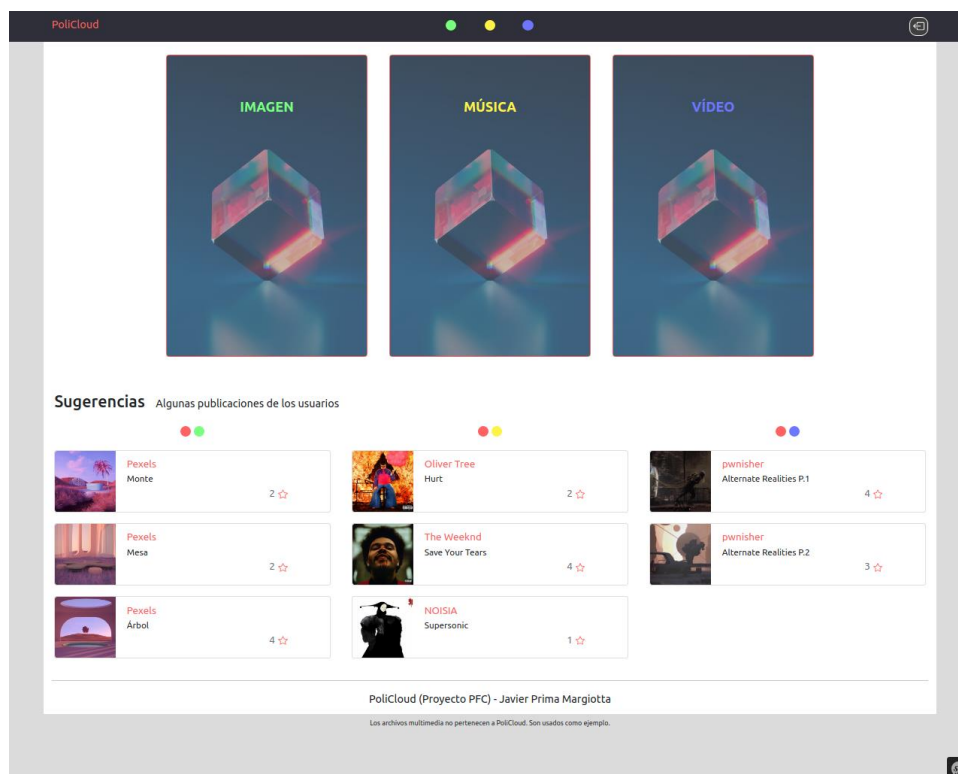
#### 4.2.1 Página de Inicio

La página de inicio es la principal como entrada a PoliCloud. En ella se muestran tres tarjetas de introducción y posteriormente un bloque con publicaciones sugeridas como cuerpo, además de la cabecera y el pie de página.

Se permite la navegación con las páginas: *Registro de usuario* y *Perfil de usuario*.

Hay elementos que varían según si se ha iniciado sesión.

Esta primera imagen es la página de inicio sin cuenta de usuario iniciada a primera vista:

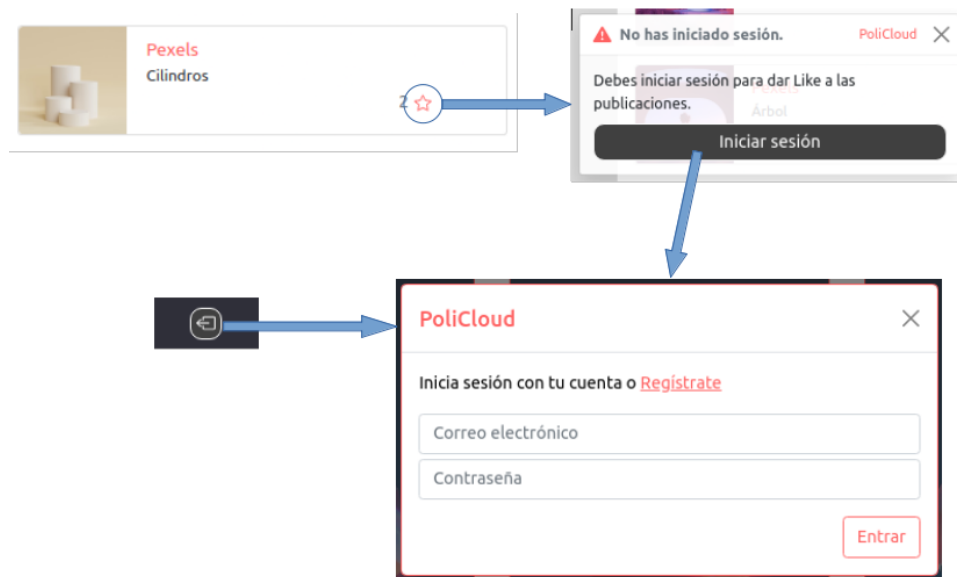


3- Página de inicio

También dispone de elementos emergentes: *Aviso de Sesión* y *Formulario de Inicio de Sesión*.

En caso de dar “me gusta” a una publicación se mostrará una alerta flotante temporal en la parte inferior izquierda de la pantalla la cual avisa de que es necesario haber iniciado sesión con anterioridad.

Principalmente se accede al formulario de inicio de sesión desde el botón ubicado a la derecha de la cabecera.



4- Elementos emergentes - página inicio

Con la cuenta de usuario iniciada los elementos cambian como se muestra en las siguientes imágenes.

Parte derecha de la cabecera:



5- Menú desplegable – cuenta iniciada

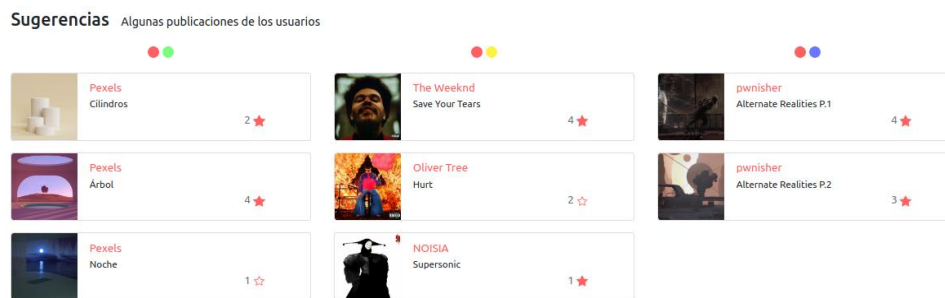
En la tarjeta de publicación, en caso de haber dado “me gusta” con anterioridad se marca el icono de la estrella:



6- Página inicio Ajax



El bloque de sugerencias muestra hasta un máximo de nueve publicaciones, de las cuales tres son imágenes, tres son audio y tres son vídeo.



7- Sugerencias

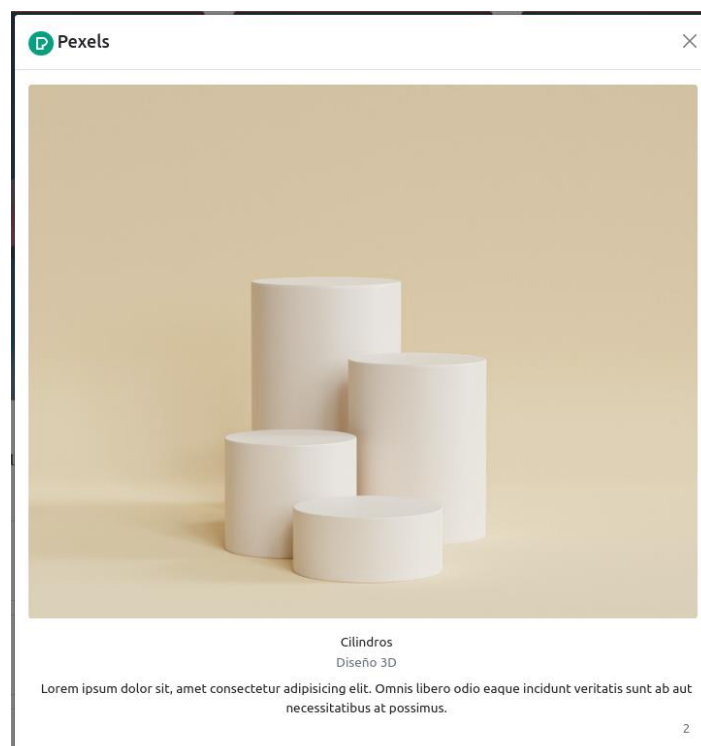
Independientemente de haber iniciado sesión, se puede visualizar el contenido de las publicaciones mostradas en el bloque de sugerencias pinchando sobre la portada.

En cada presentación se muestran los siguientes datos sobre la publicación elegida: *Imagen de perfil del usuario, nombre del usuario, contenido, nombre de la publicación, categoría, descripción y número de “me gusta”.*

En caso de visualizar las publicaciones de audio se muestra su portada.

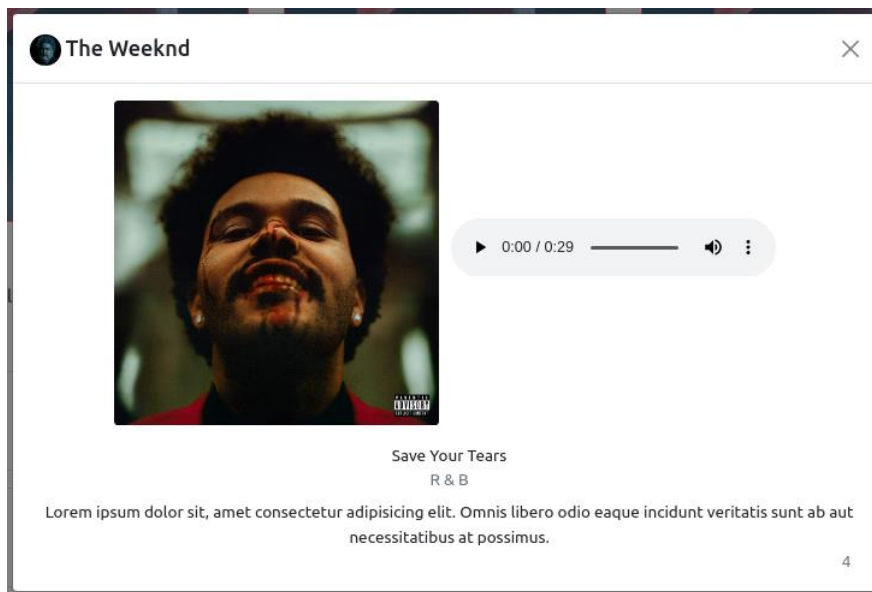
Estas son las presentaciones de cada tipo de publicación:

### Imagen



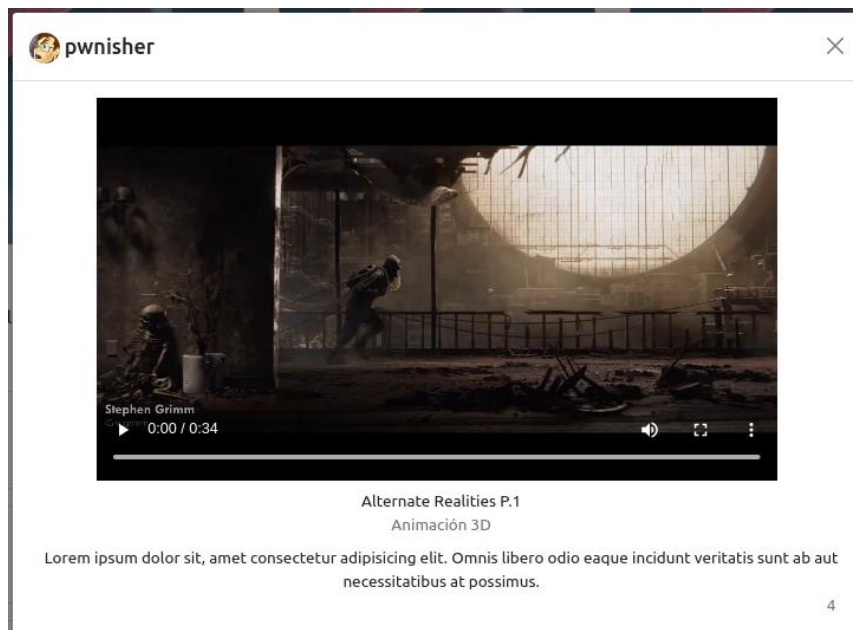
8- Publicación imagen – sugerencias

## Música



9- Publicación música – sugerencias

## Vídeo



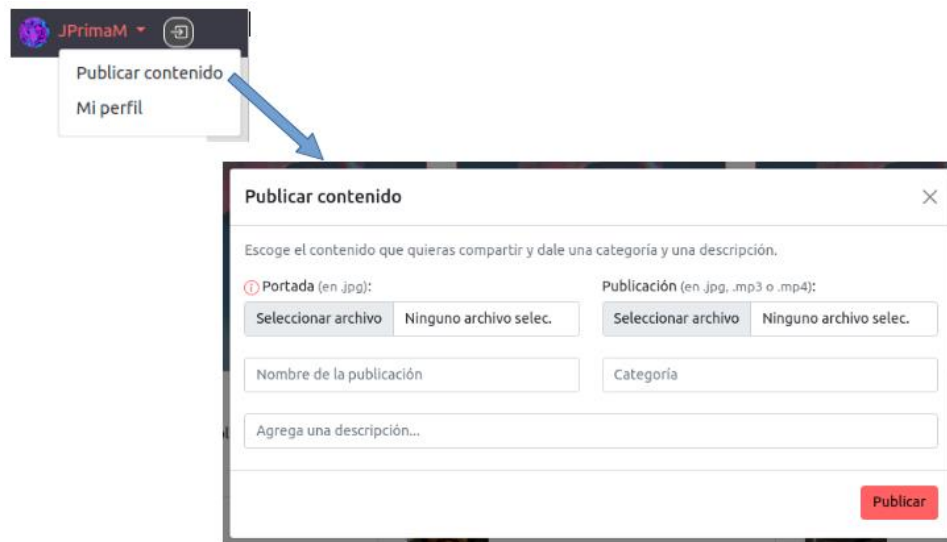
10- Publicación vídeo – sugerencias

Volviendo a la cabecera, al pinchar sobre el nombre de usuario este despliega un menú con las opciones *Publicar contenido* y *Mi perfil*. Al darle a la primera opción mostrará una ventana flotante en el centro de la pantalla con un formulario.

Para publicar contenido son requeridos tres campos de un total de cinco:

*Portada\**, *Contenido\**, *Nombre\**, *Categoría*, *Descripción*.

*\* elementos necesarios a completar para registrar la publicación.*



11 - Formulario publicar

En los campos portada y publicación se indican el formato que deben de tener estos archivos. En caso de no seleccionar los archivos correctos se cancela la publicación y aparece un mensaje de aviso.

La opción mi perfil redirige a la página del usuario actual. Esta página se explica posteriormente a la página de registro de usuario.

#### 4.3.2 Página de Registro de Usuario

Para acceder a esta página debemos de abrir el formulario de inicio de sesión y pinchar sobre “Regístrate”.

**Pol iCloud**

Inicia sesión con tu cuenta o [Regístrate](#)

Correo electrónico

Contraseña

Entrar

**Registrarse como usuario**

Correo electrónico

Apodo

Contraseña

Repite la contraseña

Seleccionar archivo Ninguno archivo selec.

Crear cuenta

**Sugerencias** Algunas publicaciones de los usuarios


PoliCloud (Proyecto PFC) - Javier Prima Margiotta

Los archivos multimedia no pertenecen a Pol iCloud. Son usados como ejemplo.

12 - Formulario registrarse – página de registro de usuario

Una vez estemos dentro de la página de registro de usuario se muestra un formulario donde se deben de poner los datos requeridos para la creación de una cuenta:

*Correo electrónico, Apodo, Contraseña, Repetir contraseña, Imagen de perfil de usuario.*

Todos los campos deben completarse para iniciar sesión. En caso de indicar un correo electrónico en uso no se creará la cuenta y se visualiza un mensaje advirtiéndolo. En

caso de que las contraseñas no coincidan se visualiza otro mensaje haciéndolo saber y tampoco se creará la cuenta de usuario.

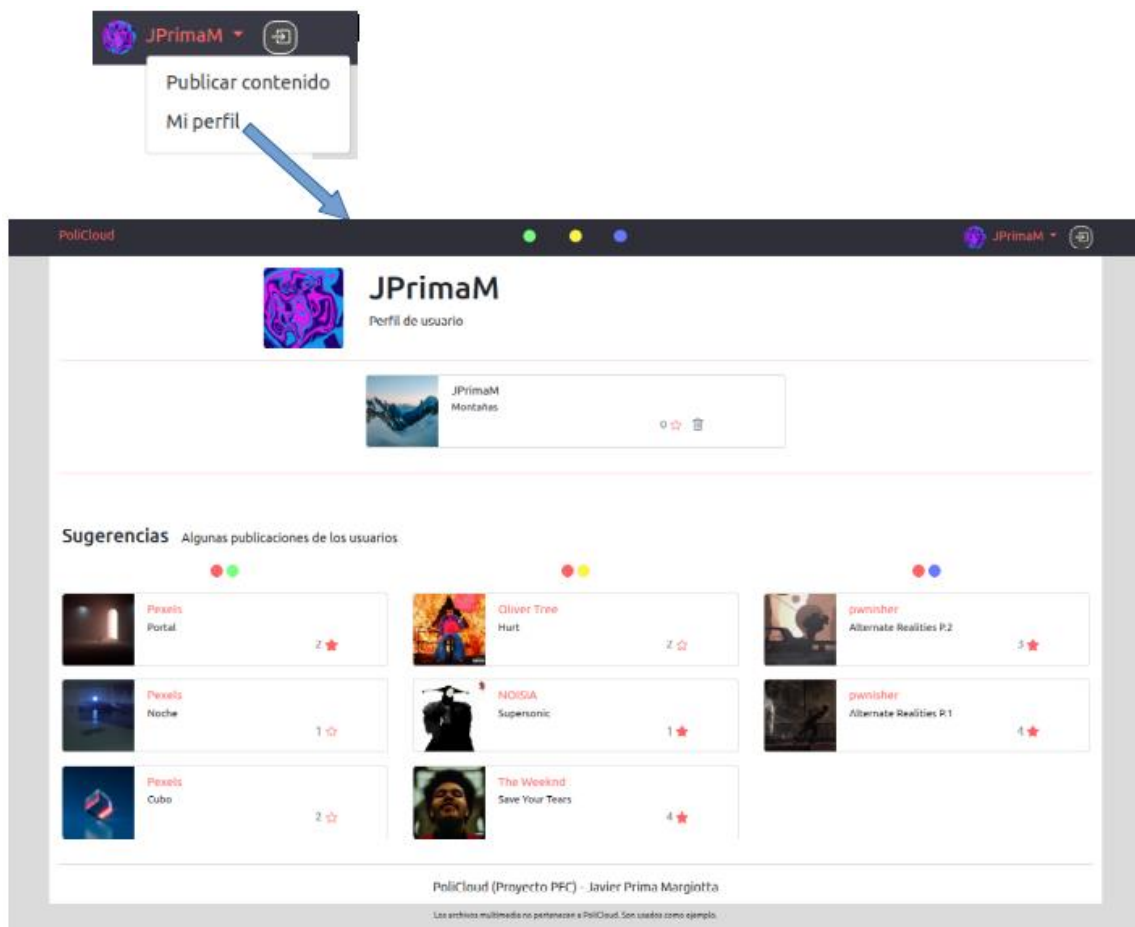
La página registro de usuario al igual que la página inicio de sesión, también dispone del apartado sugerencias con las mismas funcionalidades.

En caso de iniciar sesión, esta página no es accesible para el usuario.

#### 4.3.3 Página de Perfil de Usuario

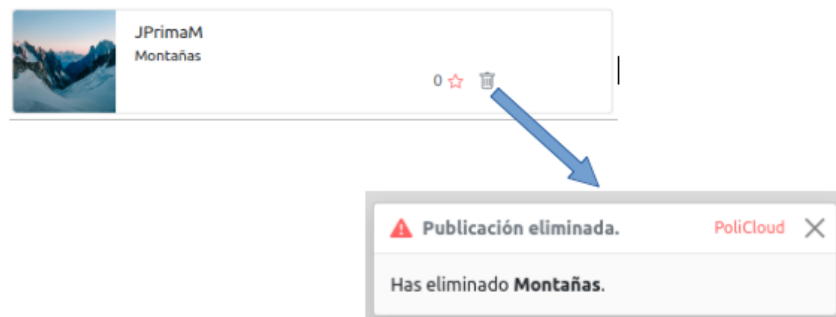
PoliCloud permite visitar los perfiles de todos los usuarios teniendo una cuenta iniciada o no. En cada perfil se muestra todo su contenido en forma de tarjetas con el mismo diseño que en el bloque de sugerencias, pero con la diferencia de que las tarjetas de perfil no disponen de una ventana flotante con una presentación completa de la publicación, sino que redirige a una página dedicada nombrada como publicación de usuario, explicada en el siguiente punto.

En esta página también se visualiza la foto de perfil del usuario visitado, su apodo y un mensaje indicando en qué página te encuentras.



13 - Perfil de usuario propio

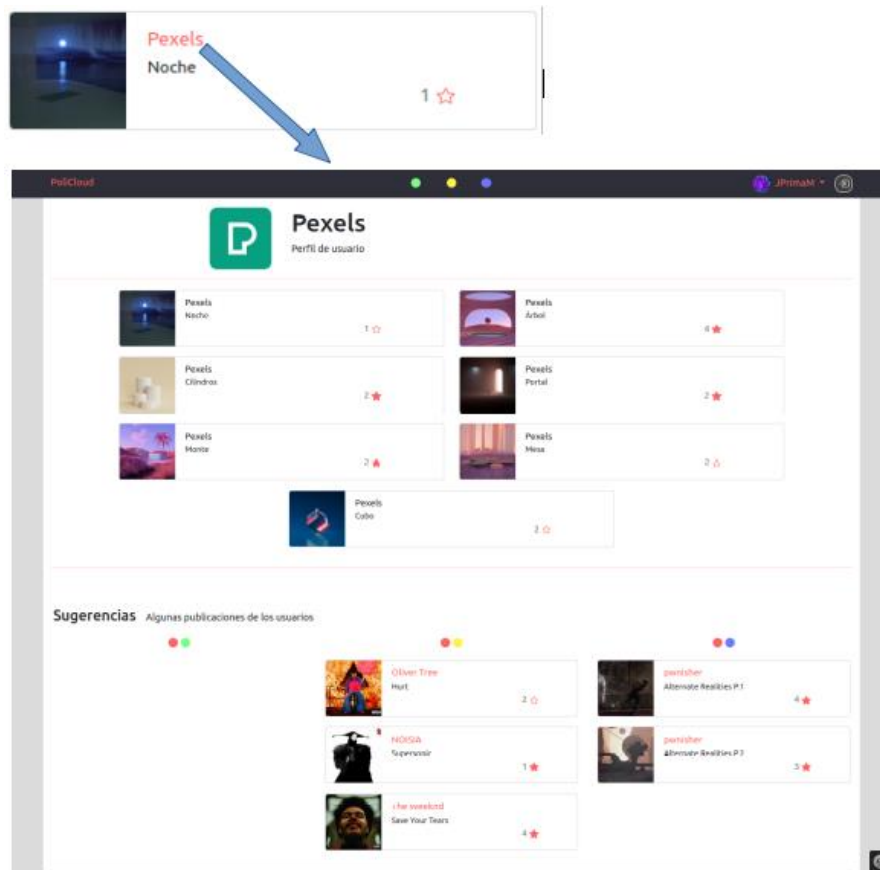
Como se puede ver en la imagen, en el propio perfil del usuario además de visualizar todas las publicaciones en forma de tarjeta permite eliminar la publicación haciendo clic sobre el icono con forma de contenedor de basura. Instantáneamente se elimina y se muestra un mensaje flotante y temporal en la parte inferior izquierda avisando de que la publicación se ha eliminado correctamente junto el nombre.



14 - Eliminar publicación

En caso de visitar un perfil de otro usuario, esta última funcionalidad no se podrá realizar y si se visita sin iniciar sesión además de no tener esta funcionalidad no se podrá dar “me gusta”, en caso de hacer clic sobre el icono de la estrella mostrará el mensaje de advertencia de inicio de sesión explicado en la página de inicio.

Para acceder al perfil de otros usuarios hay que hacer clic sobre el nombre de usuario mostrado en las tarjetas del bloque de sugerencias:



15 – Perfil de otros usuarios

Para evitar la información redundante, las publicaciones mostradas en el perfil no aparecerán en el bloque de sugerencias y en su lugar otras publicaciones existentes en ese tipo.

#### 4.3.4 Página de Publicación de Usuario

La página de publicación tiene la funcionalidad de mostrar todos los datos sobre la publicación elegida.

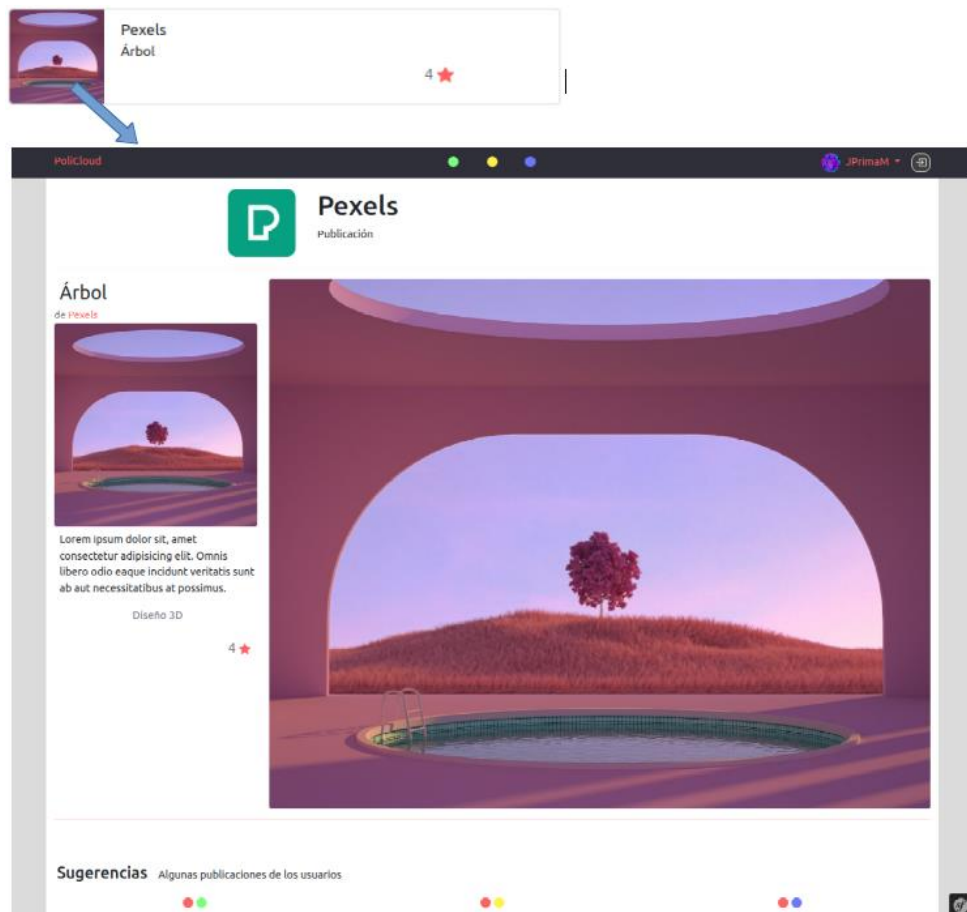
Debajo de la cabecera se muestra datos del perfil del usuario creador como su imagen de perfil y nombre de usuario además de un texto que indica que página se está mostrando.

Hay tres maneras de mostrar el contenido según el tipo (imagen, música, vídeo).

Imagen

En el lateral izquierdo indica el nombre de la publicación junto el nombre del propietario, al hacer clic en este redirige al perfil. La portada de la publicación, la descripción, la categoría el número total de usuarios que le han dado a “me gusta” y su icono.

En el resto del bloque se muestra la imagen al completo para observarla con comodidad.

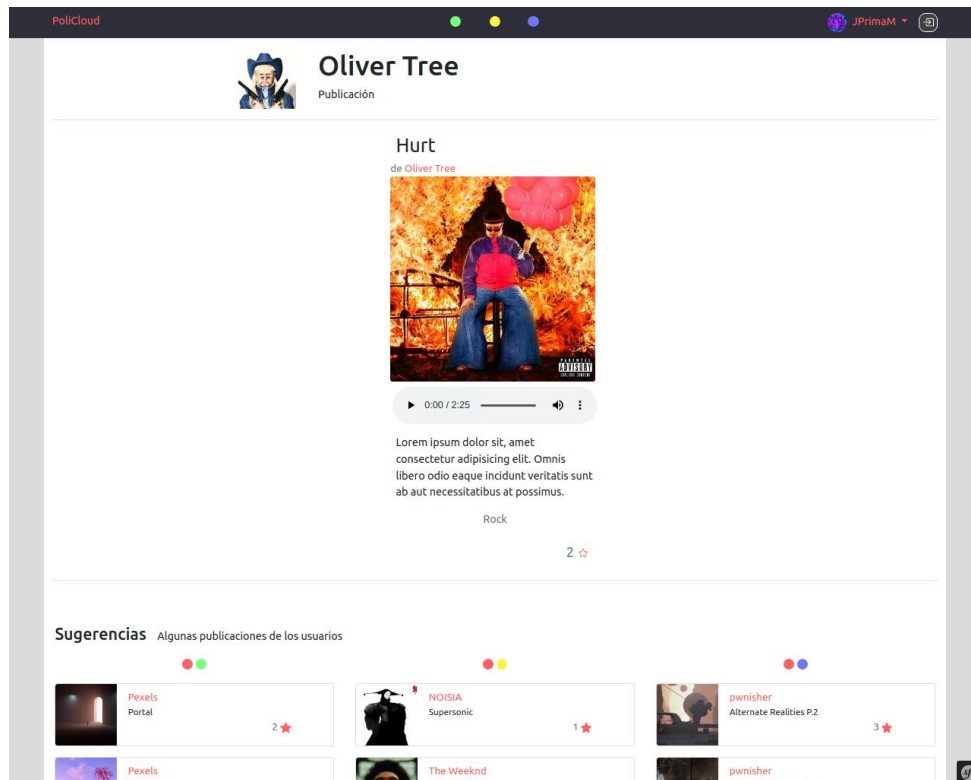


16- Publicación imagen – publicaciones

## Música

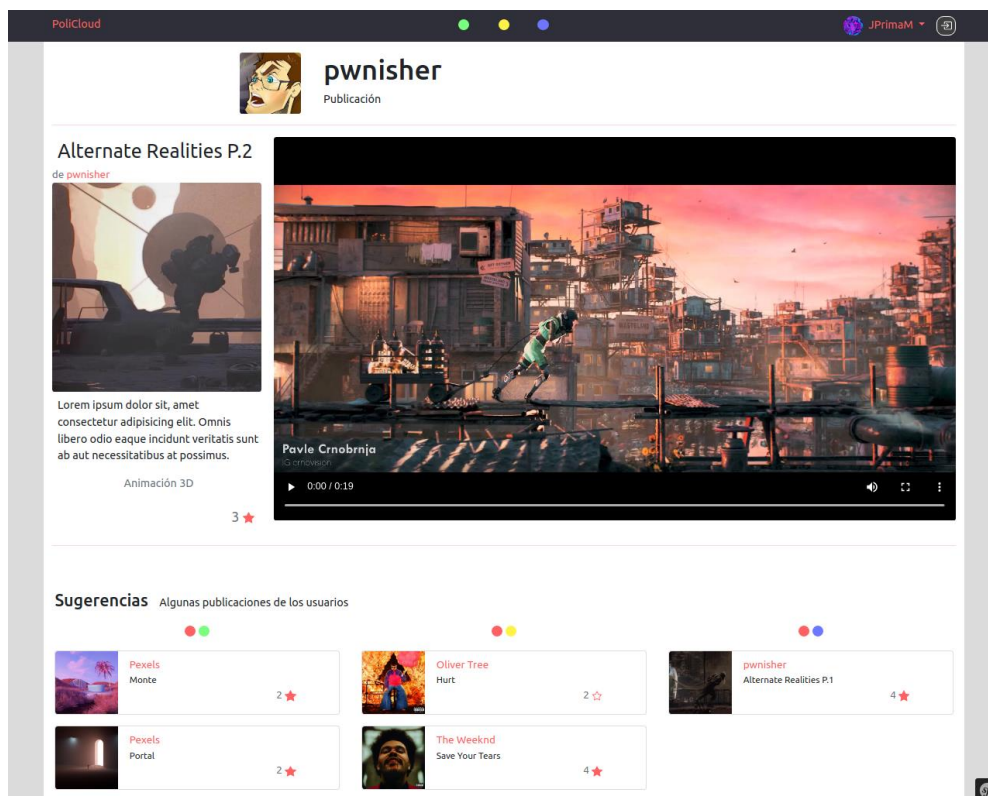
En este caso se muestra toda la información en el centro de la página.





17- Publicación música– publicaciones

## Vídeo



18 - Publicación vídeos – publicaciones

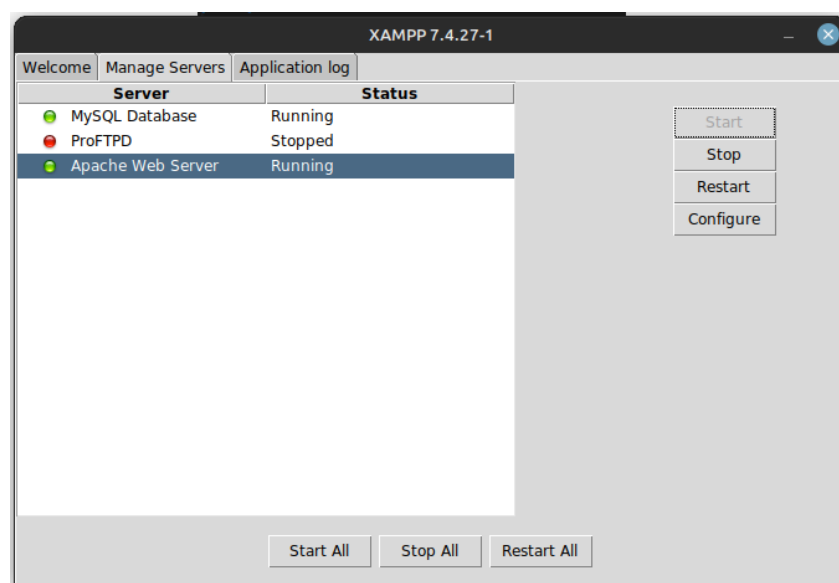
Estas tres páginas también están complementadas por el bloque de sugerencias. De la misma manera que en la página de perfil, la publicación mostrada no se mostrará en el bloque de sugerencias para evitar redundancia.

## 5. IMPLEMENTACIÓN

Para el desarrollo de PoliCloud he utilizado las siguientes herramientas.

### 5.1 Herramientas de servidor

Se ha utilizado una base de datos MySQL y servidor web Apache mediante el paquete de software libre XAMPP.



19 - XAMP

Para poder almacenar archivos de tamaño considerable he configurado MySQL y Apache de la siguiente manera:

Para MySQL he insertado los siguientes comandos en la consola de phpMyAdmin:

```
set global net_buffer_length=1000000;  
set global max_allowed_packet=1000000;
```

net\_buffer\_length=10000 establece la máxima memoria para el buffer que puede ocupar un fichero y max\_allowed\_packet=10000 indica el tamaño máximo del paquete.

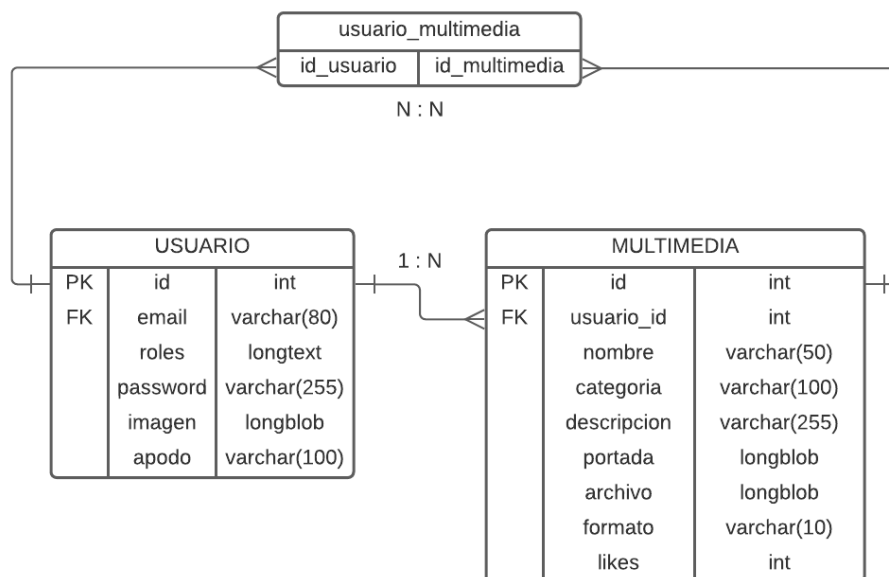
Por otra parte, hay que modificar el fichero de configuración de Apache php.ini dejando así los siguientes valores:

```
/etc/php/7.4/cli/php.ini

memory_limit=512M
post_max_size=0    # 0 indica tamaño ilimitado
upload_max_filesize=128M
```

## 5.2 Diagrama Entidad Relación

La base de datos está formada por tres tablas, dos de ellas son las entidades Usuario y Multimedia y la tercera es creada por la relación N:N la cual es representada por los “me gusta”. La relación 1:N indica que una Multimedia pertenece a un único Usuario y que muchas Multimedia pueden pertenecer a un mismo Usuario.



20 - Diagrama ER – base de datos

## 5.3 Lenguajes utilizados

Programación: PHP, JavaScript

Etiquetado: HTML5

Diseño gráfico: CSS mediante el lenguaje compilable SCSS

Para las plantillas he utilizado el motor creador de plantillas Twig, el cual se utiliza con PHP y le brinda a este una solución al tratamiento de cuestiones visuales. Permite trabajar y mostrar variables generadas en PHP e intercalar plantillas en otras.

### 5.3.1 IDE

PoliCloud está desarrollado en el entorno de desarrollo integrado Visual Studio Code con las siguientes extensiones instaladas:

- PHP Intelephense v1.8.2: Funciona como pack de características esenciales para el desarrollo con el lenguaje de programación PHP. Algunas de las características son el autocompletado, permite dar formato a la estructura de código, muestra definiciones de elementos de este lenguaje de programación, y por otro lado diferencia HTML/JS/CSS entre otras características.
- Twig v1.0.2: Esta extensión da soporte a la sintaxis a la hora de programar con el motor generador de plantillas Twig.

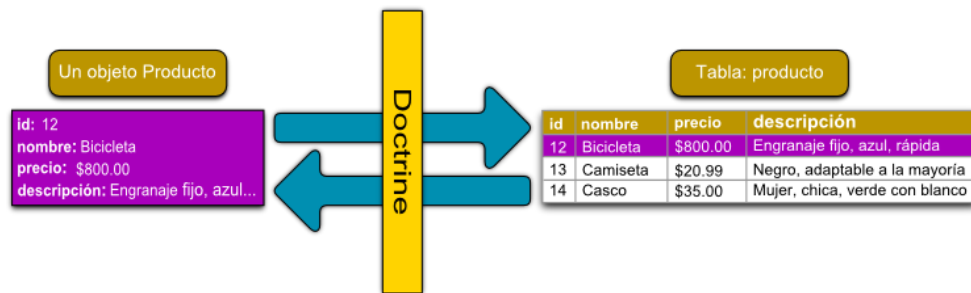
### 5.3.2 Patrón de arquitectura

El Modelo Vista Controlador (MVC), es un patrón de arquitectura de software, que divide una aplicación en 3 componentes principales, que son el Modelo, la Vista y el Controlador, que pueden ser trabajados de manera independiente, haciendo que la aplicación sea más fácil de desarrollar y mantener.

- Modelo: El modelo representa la estructura lógica de los datos en una aplicación de software. Este modelo de objetos no contiene información sobre la interfaz de usuario. Es un puente de comunicación entre la vista, el controlador y la base de datos.
- Vista: La vista es la presentación al usuario de la información contenida en el modelo. Esto generalmente consiste en pantallas que contienen información del modelo. Los datos pueden mostrarse en campos, en ventanas de editor, en tablas, etc. Además, los datos pueden ser de solo lectura o pueden ser editables.
- El Controlador: El Controlador acepta solicitudes que hace el cliente (usuario) a través del navegador, contacta al Modelo para cualquier dato que pueda necesitar, y luego toma la Vista adecuada para mostrarle esos datos a el usuario.

### 5.3.3 Framework

Como framework se utiliza Symfony el cual está diseñado para desarrollar aplicaciones web basado en el patrón Modelo Vista Controlador. Para implementarlo anteriormente hay que instalar Composer que es un controlador de paquetes PHP para administrar, descargar e instalar dependencias y librerías, similar a NPM en Node.js y necesario para crear un proyecto Symfony. Por otra parte, también se utiliza Doctrine que permite trabajar con bases de datos de una manera más interesante ya que se manejan los datos de las tablas de la base de datos como objetos.



21 - Fuente imagen <https://gitnacho.github.io/symfony-docs-es/book/doctrine.html>

### Complementos para Symfony

Para ampliar la libertad a la hora de desarrollar en la parte de frontend he utilizado la herramienta de compilación Webpack Encode.

Una vez descargado e importado en el proyecto de Symfony hay que configurar el fichero `webpack.config.js` de la siguiente manera:

El primer campo del método `.addStyleEntry()` recibe las rutas del fichero de salida donde queremos que deje el fichero compilado y el segundo el fichero de entrada que queremos que compile:

```
.addStyleEntry('css/bootstrap', './assets/css/bootstrap.scss')
.addStyleEntry('css/global', './assets/css/global.scss')
```

Para activar SASS/SCSS hay que descomentar el siguiente método:

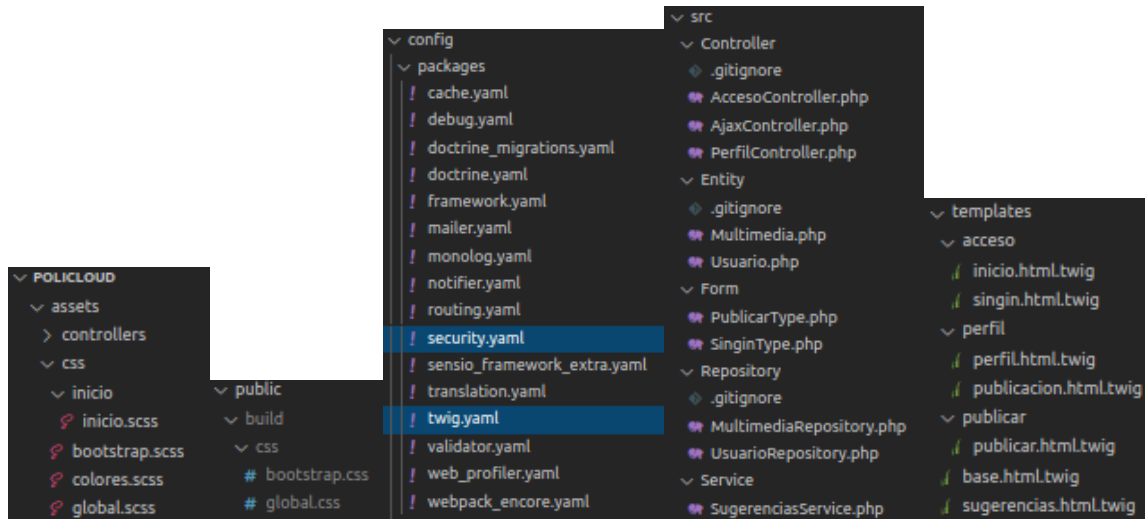
```
.enableSassLoader()
```

También se utiliza el administrador de paquetes para utilizarlo como compilador de los ficheros indicados anteriormente en `.addStyleEntry()`. Para activarlo se debe de ejecutar el comando `yarn watch` dentro del directorio del proyecto.

## 6. DESARROLLO DE LA APLICACIÓN

El Backend es la parte trasera de cualquier página web. Se trata de todo el conjunto del desarrollo que se encarga de que una página sea funcional.

## 6.1 ÁRBOL DE FICHEROS



Árbol de ficheros – PoliCloud

## 6.2 FUNCIONALIDAD

En este apartado se explican algunas partes de código utilizadas en el proyecto, tanto de Frontend como Backend.

### 6.2.1 Frontend

Para el diseño gráfico de las páginas primeramente he modificado el valor de cuatro variables existentes del framework CSS Bootstrap para asignarles los colores que representa PoliCloud.

colores.scss --> bootstrap.scss

```

1 $verde-bola:   rgb(122, 255, 129);
2 $amarillo-bola: rgb(255, 243, 72);
3 $lila-bola:    rgb(109, 119, 255);
4
5 $rojo-principal: rgb(255, 98, 98);
6 $gris-principal:  rgb(47, 47, 57);
7 $gris-secundario: rgb(64, 64, 64);
8
9 $blanco:        rgb(255, 255, 255);

1 @import './colores.scss';
2
3 $primary: $rojo-principal;
4 $secondary: $verde-bola;
5 $dark: $amarillo-bola;
6 $light: $lila-bola;
7
8 @import '~bootstrap/scss/bootstrap';

```

23 - SCSS configuración variables – código

Las variables `$primary`, `$secondary`, `$dark`, `$light` las he utilizado en el fichero `global.scss`. Este fichero contiene algunos estilos más concretos que Bootstrap no puede dar como tamaños específicos de texto, tamaños de elementos... El fichero

`bootstrap.scss` y `global.scss` son compilados a `.css` en la ruta `public/build/css`, explicado con detalle en el punto 5. Implementación.

Por otra parte, he importado Bootstrap la plantilla Twig [base.html.twig](#). En esta plantilla se importan el resto de bloques y macros Twig que he utilizado de manera que adquieren esos estilos.

La manera en la que he cargado mis estilos a la plantilla `base.html.twig`. Al indicar “asset” a la ruta buscará esta desde la carpeta `public` del árbol de archivos.

```
{{ asset('build/css/bootstrap.css') }}
```

```
{{ asset('build/css/bootstrap.css') }}
```

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>{% block title %}PoliCloud{% endblock %}</title>
    <link rel="icon"
        href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 64 64%22>%3Cpath d='M32 0c17.673 0 32 14.327 32 32S49.673 64 32 64 0 49.673 0 32S14.327 0 32 0zm0 21v21h-.001l-21 21'/%3E%3C%2Fsvg%3E'" />
    {# Run `composer require symfony/webpack-encore-bundle` to start using Symfony UX #}
    {% block stylesheets %}
        <link rel="stylesheet" href="{{ asset('build/css/bootstrap.css') }}">
        <link rel="stylesheet" href="{{ asset('build/css/global.css') }}">
    {% endblock %}
</head>
```

## 24 – Cargar hoja de estilos en Twig – código

Para la estructura de los elementos he utilizado las clases de Bootstrap consultando su documentación.

### 6.2.2 Backend

A continuación, se muestran algunas de las partes de código que permiten la funcionalidad de PoliCloud.

## Inicio de Sesión

He diseñado un formulario con la utilización de clases de Bootstrap. A la hora de hacer submit este formulario carga el método `login()` - `app_login` declarado en el controlador `AccesoController.php`. Esto es gracias a la etiqueta Twig que llama a la ruta con `{{path('app_login')}}`.

base.html.twig

```

<!-- Modal LOGIN -->
<div class="modal fade mt-5" id="modalLogin" tabindex="-1" aria-labelledby="modalLoginLabel"
  aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content border border-primary">
      <div class="modal-header">
        <h5 class="modal-title text-primary">PoliCloud</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
          aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <p class="text-black">Inicia sesión con tu cuenta o <a
          href="{{ path('app_singin') }}" class="text-primary">Regístrate</a></p>
        <form action="{{ path('app_login') }}" method="post" class="text-center">
          <input type="text" class="form-control mb-1" placeholder="Correo electrónico"
            aria-label="Escribe tu correo electrónico" aria-describedby="button-addon2"
            name="username">
          <input type="password" class="form-control mb-3" placeholder="Contraseña"
            aria-label="Escribe tu contraseña" aria-describedby="button-addon2"
            name="password">
          <div class="flex-end">
            <button class="btn btn-outline-primary" type="submit"
              id="button-addon2">Entrar</button>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>

```

25 – Formulario Login en Twig - código

### AccesoController.php

```

/**
 * @Route("/login", name="app_login", methods={"POST"})
 */
public function login(AuthenticationUtils $authenticationUtils): Response
{
    $error = $authenticationUtils->getLastAuthenticationError();
    $lastUsername = $authenticationUtils->getLastUsername();

    $usuario = $this->getUser();
    if (!$usuario) {
        return $this->render('acceso/inicio.html.twig', [
            'error' => $error,
            'lastUsername' => $lastUsername
        ]);
    } else {
        return $this->redirectToRoute('app_singin');
    }
}

```

26 – Formulario Login en PHP - código

El objeto AuthenticationUtils adquiere todos los elementos del formulario mediante la cabecera POST y realiza una comprobación de los datos con la base de datos permitiendo o no el inicio de sesión.

Para esto es necesario haber hecho una configuración previa en [security.yaml](#).

```

main:
  lazy: true
  form_login:
    provider: app_user_provider
    login_path: app_login
    check_path: app_login
  logout:
    path: app_logout

```

27 - Configuración Login y Logout en yaml - código



### Perfil

El controlador del perfil es uno de los más completos junto al control de publicación. Este recibe la ID del usuario a visitar mediante la etiqueta `{{path('app_perfil',{idPerfil:multimedia.usuario.id})}}` que se encuentra en la plantilla `sugerencias.html.twig`.

La función `perfil()` - `app_perfil` recoge el repositorio del usuario con la ID recibida para posteriormente trabajarla, extrayendo el repositorio Multimedia asociado a este Usuario, es decir todas sus publicaciones. Una vez se adquieren tenemos tanto el Usuario como Multimedia, se codifica la imagen de perfil del Usuario y portadas de cada Multimedia (también son imágenes).

Las portadas codificadas se almacenan en el array `$portadasCodificadas` como solución alternativa al error que se generaba al insertar la portada compilada de nuevo a la misma entidad Multimedia.

### PerfilController.php

```
/**
 * @Route("/perfil/{idPerfil}", name="app_perfil")
 */
public function perfil(Request $request, EntityManagerInterface $em, $idPerfil): Response
{
    /** Perfil visitado */
    $perfil_repositorio = $em->getRepository(Usuario::class)->findOneBy(array("id" => $idPerfil));
    $multimedia_perfil_repositorio = $perfil_repositorio->getMultimedia();
    $imagen_perfil = base64_encode(stream_get_contents($perfil_repositorio->getImagen(), -1, -1));

    $portadasCodificadas = array();
    foreach($multimedia_perfil_repositorio as $multimedia_perfil) {
        array_push($portadasCodificadas, base64_encode(stream_get_contents($multimedia_perfil->getPortada(), -1, -1)));
    }
    /** Fin apartado perfil visitado */
}
```

28 – Formulario publicar 2 en PHP- código

En caso de tener la sesión de usuario iniciada es necesario recoger datos de este usuario para mostrarlos en la cabecera (imagen de perfil y apodo) y dar funcionalidades como publicar contenido desde esta misma página.

```
/** En caso de tener sesión iniciada */
$usuario = $this->security->getUser();

if ($usuario) {
    /* Se recogen los valores del usuario necesarios para mostrarlos hacer render en la plantilla Twig */
    $rol = $usuario->getRoles();
    $id = $usuario->getId();
    $apodo = $usuario->getApodo();
    $likes = $usuario->getLikes();

    $imagen = base64_encode(stream_get_contents($usuario->getImagen(), -1, -1));

    /** Esto es porque en caso de que el usuario visite su propio perfil,
     * se chafa el objeto y vacía el campo imagen y no se ve img de perfil en header.
     * Con esto se le vuelve dar el valor. */
    if($usuario->getId() == $perfil_repositorio->getId()) {
        $imagen = $imagen_perfil;
    }
}
```

29 – Formulario publicar 2 en PHP - código

En esta captura se muestra cómo se prepara el formulario y se comprueba que haya sido enviado y validado, al ser así, se realizan comprobaciones en la imagen de portada y en el archivo de la publicación.

```

/* Se recoge nuevamente el usuario que ha iniciado sesión */
$usuario_repositorio = $em->getRepository(Usuario::class)->findOneBy(array("email" => $usuario->getUserIdentifier()));
$multimedia = new Multimedia();
/** Se prepara un formulario relacionado con la clase PublicarType para
 * relacionar los campos del formulario, recogerlos y trabajarlos */
$form = $this->createForm(PublicarType::class, $multimedia);
$form->handleRequest($request);

$formato_portada_valido = true;
$formato_archivo_valido = true;

/** Una vez el formulario es enviado y válido empieza la comprobación de los campos portada y archivo contengan valor y
 * que sus extensiones sean las aceptadas.
 */
if ($form->isSubmitted() && $form->isValid()) {
    $portada = $form->get("portada")->getData();
    $archivo = $form->get("archivo")->getData();

    if ($archivo && $portada) {
        $formato_portada = $portada->guessExtension();
        $formato_archivo = $archivo->guessExtension();

        $formatos = array("jpg", "mp3", "mp4");
    }
}

```

30 – Formulario publicar 3 en PHP- código

Si se cumplen los requisitos de formato se almacena el resto de datos adquiridos en el formulario en la nueva entidad Multimedia para posteriormente enviar la entidad a la base de datos. En caso de no cumplir los requisitos de formato se modifican las variables `$formato_archivo_valido` y `$formato_portada_valido` a falso.

```

/** En caso de que las extensiones hayan sido aceptadas se asignan los valores a la nueva
 * entidad Multimedia creada anteriormente */
if ($formato_portada == "jpg") {
    if (in_array($formato_archivo, $formatos)) {
        $multimedia->setPortada(file_get_contents($portada));

        $multimedia->setArchivo(file_get_contents($archivo));
        $multimedia->setFormato($archivo->guessExtension());

        $multimedia->setUsuario($usuario_repositorio);

        try {
            /** Se prepara la consulta y se envía a la base de datos mediante Doctrine */
            $em->persist($multimedia);
            $em->flush();
        } catch (\Exception $e) {
            return new Response("Esto no va nonono. Posiblemente haya que modificar el tamaño de paquete en MySQL/php.ini");
        }
        return $this->redirectToRoute('app_inicio');
    } else {
        $formato_archivo_valido = false; /* En caso de que el archivo no tenga formato .jpg, .mp3, .mp4 */
    }
} else {
    $formato_portada_valido = false; /* En caso de que la portada no tenga formato .jpg */
}

```

31 – Formulario publicar 4 en PHP- código

Una vez se han manejado todos los datos y enviados a la base de datos, las variables se envían a la plantilla Twig `perfil.html.twig` para ser mostrados por interfaz gráfica. El primer bloque `if()` se ejecuta en caso de que haya un usuario con la cuenta iniciada y que tenga el `ROLE 'ROLE_RESIDENTE'` (rol asignado a los usuarios registrados). El segundo bloque `if()` se ejecuta cuando no hay usuario activo, es decir, si se está visitando la página perfil sin haber iniciado sesión ya que no se disponen de los datos de usuario.

```

if (in_array('ROLE_RESIDENTE', $rol)) {
    return $this->render('perfil/perfil.html.twig', [
        'rol' => $rol[0],
        'id' => $id,
        'apodo' => $apodo,
        'imagen' => $imagen,
        'likes' => $likes,
        'form' => $form->createView(),
        'formato_portada_valido' => $formato_portada_valido,
        'formato_archivo_valido' => $formato_archivo_valido,
        'perfil_repositorio' => $perfil_repositorio,
        'imagen_perfil' => $imagen_perfil,
        'multimedia_perfil' => $multimedia_perfil_repositorio,
        'portadas_multimedia_perfil' => $portadasCodificadas
    ]);
} else {
    return $this->render('perfil/perfil.html.twig', [
        'rol' => "",
        'id' => "",
        'apodo' => "",
        'imagen' => $imagen_perfil,
        'likes' => "",
        'form' => "",
        'formato_portada_valido' => "",
        'formato_archivo_valido' => "",
        'perfil_repositorio' => $perfil_repositorio,
        'imagen_perfil' => $imagen_perfil,
        'multimedia_perfil' => $multimedia_perfil_repositorio,
        'portadas_multimedia_perfil' => $portadasCodificadas
    ]);
}

```

32 – Formulario publicar 5 en PHP- código

### perfil.html.twig

En la plantilla Twig se muestran todos los elementos necesarios, pero voy a mostrar los elementos que considero más relevantes ya que ahora solo es mostrar las variables recibidas por `PerfilController.php`.

He importado los estilos de `base.html.twig` de la siguiente manera:

```
{%extends 'base.html.twig'%}
```

```
{%block cuerpo%
```

Esta segunda etiqueta le asigna el nombre cuerpo al bloque de esta plantilla para incorporar todo el contenido en la etiqueta `{%block cuerpo%}{%endblock%}` encontrada en `base.html.twig`.

Se utiliza un bucle `for`, el cual tiene un contador que almacena el número de vueltas que realiza para posteriormente desplazarse por las posiciones del array que contiene las portadas de cada publicación. Este bucle `for` se ejecuta tantas veces como Multimedia se disponga en la array `multimedia_perfil`. Este bucle funciona como un bucle `foreach` en php.

```

<div class="row px-0 justify-content-center">
    {% for contador, multimedia in multimedia_perfil %}
        <div class="col-5" id="tarjeta_{{ multimedia.id }}">
            <div class="card mb-3" style="max-width: 540px;">
                <div class="row g-0">
                    <div class="col-md-4 visor">
                        <a type="button" class="m-0 p-0" href="{{ path('app_publicacion', {'idPerfil':multimedia.usuario.id, 'idPublicacion':multimedia.id}) }}">
                            
                        </a>
                    </div>
                </div>
            </div>
        </div>
    {% endfor %}
</div>

```

33 – Diseño de plantilla 1 en Twig - código

Para mostrar el icono con forma de estrella activada que hace referencia a “me gusta”, se hace una serie de comprobaciones según si el usuario ha iniciado sesión, en caso de ser así comprueba que la ID de la Multimedia tenga relación con la ID de la entidad Usuario y de ser así se pinta el icono. Relación y tabla: N:N usuario\_multimedia.

```
{% if likes == "" %} {# En caso de que no haya usuario logged #}
<a onclick="darLike(this)" id="{{ multimedia.id }}">
  <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-star p-0 m-0 btn link-pr
  <path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.356c.3
  </svg>
</a>
{% else %} {# Si está dentro... #}
{% set break = false %}
{% for like in likes %} {# ...comprueba que las IDs de LIKES (Colecciones Multimedia y Usuario) coincidan con la Multimedia
                          cargada y el Usuario logged y asigna "like" si es así #}
  {% if like.id == multimedia.id and like.usuario.id == multimedia.usuario.id %}
    <a onclick="quitarLike(this)" id="{{ multimedia.id }}">
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-star-fill link-pr
      <path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l-.83-4.73l.173 6.765c-.329-.314-.158-.888.283-.951l.898-.696l
      </svg>
    </a>
    {% set break = true %}
  {% endif %}
{% endfor %}
{% if break == false %} {# ...en caso de no encontrar ninguna coincidencia asigna "no-like" #}
  <a onclick="darLike(this)" id="{{ multimedia.id }}">
    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-star p-0 m-0 btn link
    <path d="M2.866 14.85c-.078.444.36.791.746.593l4.39-2.256 4.389 2.256c.386.198.824-.149.746-.592l-.83-4.73 3.522-3.35
    </svg>
  </a>
  {% endif %}
{% endif %}
```

34 – Diseño de plantilla 2 en Twig - código

## 7. LEYENDA DE IMÁGENES

- 1 – Plan de trabajo
- 2 – Diseño de las páginas de PoliCloud
- 3 – Página de inicio
- 4 – Elementos emergentes - página inicio
- 5 – Menú desplegable - cuenta iniciada
- 6 – Página inicio Ajax
- 7 – Sugerencias
- 8 – Publicación imagen - sugerencias
- 9 – Publicación música - sugerencias
- 10 – Publicación vídeo - sugerencias
- 11 – Formulario publicar
- 12 – Formulario registrarse - página de registro de usuario
- 13 – Perfil de usuario propio
- 14 – Eliminar publicación
- 15 – Perfil de otros usuarios

- 16 – Publicación imagen - publicaciones
- 17 – Publicación música - publicaciones
- 18 – Publicación vídeos - publicaciones
- 19 – XAMP
- 20 – Diagrama ER - base de datos
- 21 – Fuente imagen <https://gitnacho.github.io/symfony-docs-es/book/doctrine.html>
- 22 – Árbol de ficheros - PoliCloud
- 23 – SCSS configuración variables - código
- 24 – Cargar hoja de estilos en Twig - código
- 25 – Formulario LogIn en Twig - código
- 26 – Formulario LogIn en PHP - código
- 27 – Configuración LogIn y LogOut en yaml - código
- 28 – Formulario publicar 2 en PHP - código
- 29 – Formulario publicar 2 en PHP - código
- 30 – Formulario publicar 3 en PHP - código
- 31 – Formulario publicar 4 en PHP - código
- 32 – Formulario publicar 5 en PHP - código
- 33 – Diseño de plantilla 1 en Twig - código
- 34 – Diseño de plantilla 2 en Twig - código

## 8. BIBLIOGRAFÍA Y ANEXOS

### *Christian García*

Este vídeo explica como instalar Webpack Encoder junto Bootstrap para la utilización de SASS/SCSS en Symfony y su configuración.

Como Usar Preprocesadores Sass En Tu Web Hecha En Symfony.

<https://youtu.be/jvwJaNgakc>

### *Galatar*

Explicación de cómo crear servicios en Symfony

<https://galatar.com/como-crear-y-utilizar-servicios-en-symfony-4/>

## *Injsite*

Explicación de cómo crear servicios en Symfony.

<https://injsite.com/symfony-tutorial-10-los-servicios-la-inyeccion-de-dependencias/>

## *SoundCloud*

Música e imágenes utilizadas como ejemplo

<https://soundcloud.com/olivertree/hurt>

<https://soundcloud.com/noisia/supersonic-vip-skrillex>

<https://soundcloud.com/theweeknd/save-your-tears>

## *Pexels*

Imágenes utilizadas como ejemplo

<https://www.pexels.com/>

## *Prestasoo*

Explicación de cómo modificar el tamaño de archivos en Apache y MySQL

<https://www.prestasoo.com/blog/how-to-increase-maximum-upload-file-size-in-prestashop>

## *Prestasoo*

Explicación de cómo modificar el tamaño de archivos en Apache y MySQL

<https://www.prestasoo.com/blog/how-to-increase-maximum-upload-file-size-in-prestashop>

## *Symfony*

Documentación de Symfony

<https://symfony.com/doc/current/index.html>

## *Bootstrap*

Documentación de Bootstrap

<https://getbootstrap.com/docs/5.1/getting-started/introduction/>

Repositorio GIT alojado en GitHub, en él se encuentra el proyecto PoliCloud junto a esta documentación.



<https://github.com/JPrimaM/PoliCloud>