

IES POLITÉCNICO
HERMENEGILDO LANZ

GRANADA

DEPARTAMENTO INFORMÁTICA Y COMUNICACIONES



MANDOPEDIA

PROYECTO FINAL CICLO SUPERIOR DESARROLLO DE
APLICACIONES WEB

AUTOR: JUAN DE DIOS C.

Granada,¹ Junio de 2022

ÍNDICE

1. BREVE DESCRIPCIÓN DEL PROYECTO-----	3 -
2. ANÁLISIS DEL CONTEXTO-----	3 -
3. ESTRUCTURA DEL PROYECTO-----	4 -
3.1. ARQUITECTURA DE LA APLICACIÓN-----	4 -
3.2. TIPOS DE USUARIOS-----	4 -
3.3. MAPA DE NAVEGACIÓN-----	5 -
4. GUÍA DE ESTILOS-----	6 -
4.1. COLORES PRINCIPALES-----	6 -
4.2. TIPOGRAFÍAS-----	6 -
4.3. ICONOGRAFÍA-----	7 -
5. MOCKUPS-----	8 -
6. DISEÑO DE LA BASE DE DATOS-----	13 -
6.1. DIAGRAMA ENTIDAD-RELACIÓN-----	14 -
6.2. PASO A TABLAS-----	14 -
7. DESCRIPCIÓN DEL FRONT-END-----	15 -
7.1. TECNOLOGÍAS USADAS-----	15 -
7.2. FUNCIONALIDAD-----	16 -
8. DESCRIPCIÓN DEL BACK-END-----	23 -
8.1. TECNOLOGÍAS USADAS-----	23 -
8.2. FUNCIONALIDAD-----	24 -
9. DESPLIEGUE DE LA APLICACIÓN-----	29 -
10. BIBLIOGRAFÍA-----	32 -

1. BREVE DESCRIPCIÓN DEL PROYECTO

Este proyecto es una wiki acerca de los personajes de la serie de Star Wars: The Mandalorian, en donde aparte de obtener información, se pueden editar dichos artículos informativos.

2. ANÁLISIS DEL CONTEXTO

Se parte de un contexto en el que ya existen wikis de Star Wars, pero genéricas sin entrar al detalle, ya sea de una serie de televisión, de videojuegos, cómics, o de una era en concreto dentro del universo de esta saga. El objetivo de esta aplicación es **cubrir ese vacío**, centrándose en la serie The Mandalorian, y explicando al detalle cada personaje, con las funcionalidades de una wiki.

Se eligió este **formato** porque permite una muy alta personalización del diseño, pudiendo investigar y desarrollar nuevas áreas del mismo.

Los **requisitos** a cubrir en esta aplicación son:

- Artículos informativos de los principales personajes de la serie para todos los usuarios, sin necesidad de encontrarse registrado.
- Registro e inicio de sesión para que los usuarios puedan editar dichos artículos.
- Panel de administración para la gestión de la base de datos por parte del administrador de la aplicación.
- Creación de artículos por parte de ciertos usuarios.
- Gestión de las modificaciones de los usuarios sobre los artículos.

3. ESTRUCTURA DEL PROYECTO

3.1. ARQUITECTURA DE LA APLICACIÓN

La aplicación seguirá una arquitectura de modelo-vista-controlador, creada con Symfony, en la cual:

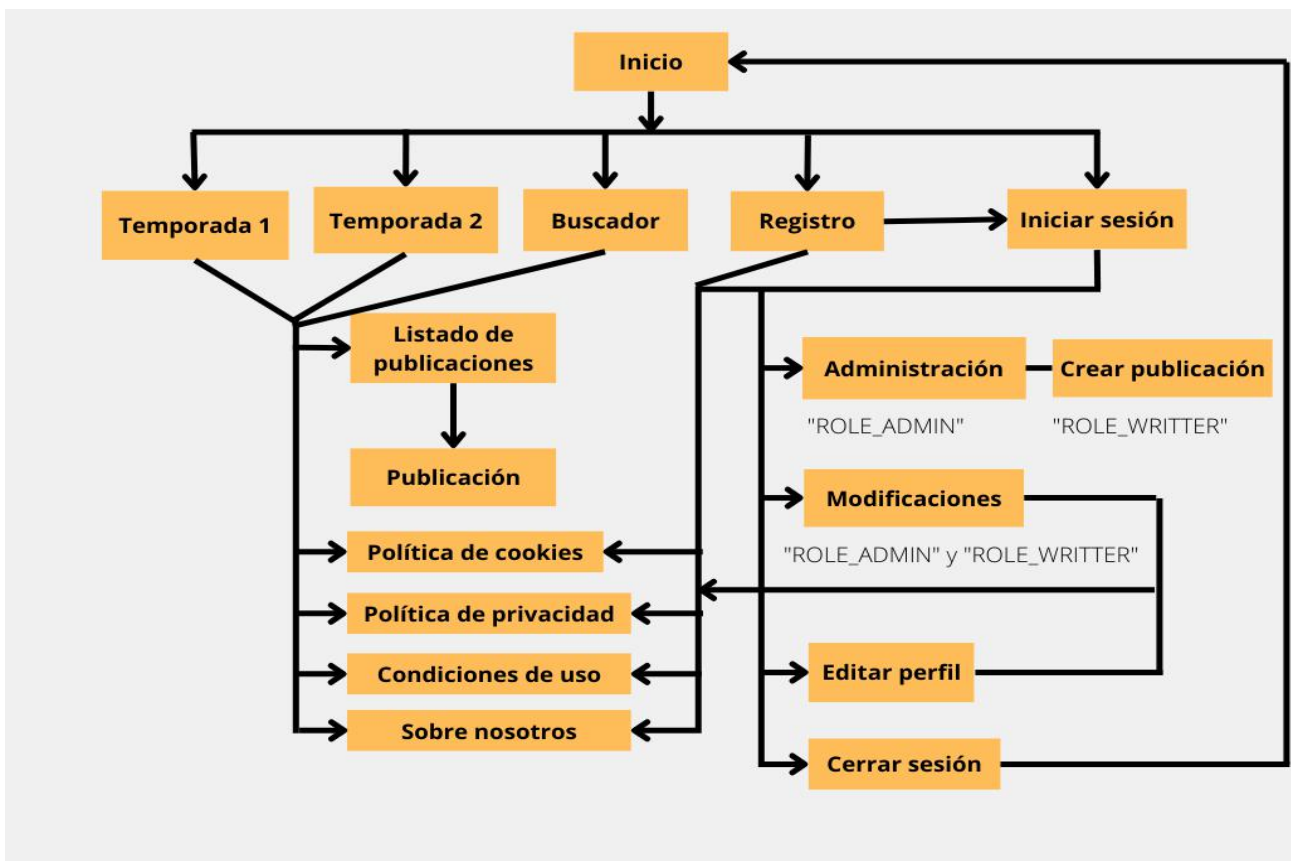
- Los modelos son los objetos con entidad propia, de los que se quiere guardar información en la base de datos. En la aplicación están las categorías (temporada 1 y temporada 2 de la serie), los artículos informativos, llamados a partir de ahora publicaciones, las modificaciones y los usuarios; acompañados de sus respectivos repositorios, o métodos para obtener y cambiar esta información de la base de datos.
- Las vistas son las partes de la aplicación que permiten al usuario final interactuar con el sistema. Se encuentra la vista de iniciar sesión, la de registro, listado de publicaciones por categoría, resultados de búsqueda, listado de modificaciones, y el panel de control para poder ver, editar y borrar los modelos explicados anteriormente.
- Los controladores se encargan de la lógica y funcionamiento de la aplicación. Gestionan las rutas y dirigen al usuario por la aplicación proporcionándole la información solicitada. Hay un controlador por cada entidad, gestionando la información que se pide en las vistas y llamando a los repositorios para hacer las consultas oportunas a la base de datos y obtener, insertar o eliminar dicha información.

3.2. TIPOS DE USUARIOS

Hay 3 tipos de roles de usuarios en la aplicación, es decir, tres perfiles distintos que realizan determinadas acciones:

- “ROLE_ADMIN” . Pertenece a un único usuario, el administrador del sistema, y puede: editar y borrar usuarios; crear, editar y borrar categorías; crear, editar y borrar publicaciones y crear, aceptar y borrar modificaciones hechas por los usuarios en las publicaciones.
- “ROLE_WRITTER” . El rol “escritor” es otorgado por el administrador a aquellos usuarios que crean un contenido de calidad y quieren compartir la responsabilidad de gestionar las publicaciones, creándolas y aceptando o eliminando las modificaciones de los usuarios. No puede borrar publicaciones.
- “ROLE_USER” . Rol genérico que tienen todos los usuarios una vez se hayan registrado en el sitio web. Les permite modificar el texto de las publicaciones.


3.3. MAPA DE NAVEGACIÓN




1. Mapa de navegación del sitio web, realizado con Canva

4. GUÍA DE ESTILOS

4.1. COLORES PRINCIPALES

Color de fondo: #1c1717 


Blanco para la fuente: #f6f6f6 


Fondo header: #2f2e2e 

Fondo menú usuario: #97a2b3 

Fondo header vista tablet y móvil: #475262 


Color principal dorado: #C8A12B 

Dorado claro: #E0C677 


Dorado oscuro: #A6872B 


Dorado oscuro para gradiente del título: #2d2513 


Dorado botón deshabilitado: #887c59 

Rojo: #aa160e 

Rojo-hover: #84130d 

Rojo-olas-boton: rgba(170, 22, 14, 0.4) 


Verde: #009d5e 

Verde-hover: #007747 

Verde-olas-boton: rgba(0, 157, 94, 0.4) 

Negro para sombra header y fondo video: #0E0E0E 

Negro para sombra header solo en la página principal: #000 

Rojo para la fuente de error en formulario: #e10026 

4.2. TIPOGRAFÍAS

Texto: HindVadodara-Regular

abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890.,:;' " (!?) +-*/=

¹² El veloz murciélago hindú comía feliz cardillo y kiwi. La cigüeña tocaba el saxofón detrás del palenque de paja. 1234567890

¹⁸ El veloz murciélago hindú comía feliz cardillo y kiwi. La cigüeña tocaba el saxofón detrás del palenque de paja. 1234567

²⁴ El veloz murciélago hindú comía feliz cardillo y kiwi. La cigüeña tocaba el saxofón detrás

Títulos: mandalorelaser

ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890.,: ' " (!?) -+/*

12 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA EL SAXOFÓN DETRÁS DEL PALENQUE DE PAJA. 1234567890
18 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA EL SAXOFÓN DETRÁS DEL PALENQUE DE PAJA. 1234567890
24 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA EL SAXOFÓN DETRÁS DEL PALENQUE DE
36 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA

Firma en página sobre nosotros: mandalore

ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890.,: ' " (!?) -+/*

12 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA EL SAXOFÓN DETRÁS DEL PALENQUE DE PAJA. 1234567890
18 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA EL SAXOFÓN DETRÁS DEL PALENQUE DE PAJA. 1234567890
24 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA EL SAXOFÓN DETRÁS DEL PALENQUE DE
36 EL VELOZ MURCIÉLAGO HINDÚ COMÍA FELIZ CARDILLO Y KIWI. LA CIGÜEÑA TOCABA

TAMAÑO DE FUENTES:

Texto:

- Muy pequeño: 11px
- Por debajo de 600px = 13px
- Por encima de 601px = 15px
- Por encima de 993px = 17px

Títulos de tarjetas:

- Por debajo de 600px = 16px
- Por encima de 601px (tarjetas grandes de inicio solo) = 24px
- Por encima de 601px = 20px

4.3. ICONOGRAFÍA

LOGO



Iconos de fontawesome versión 6.1.1. sin bordes en blanco



Icono de usuario



Icono facebook

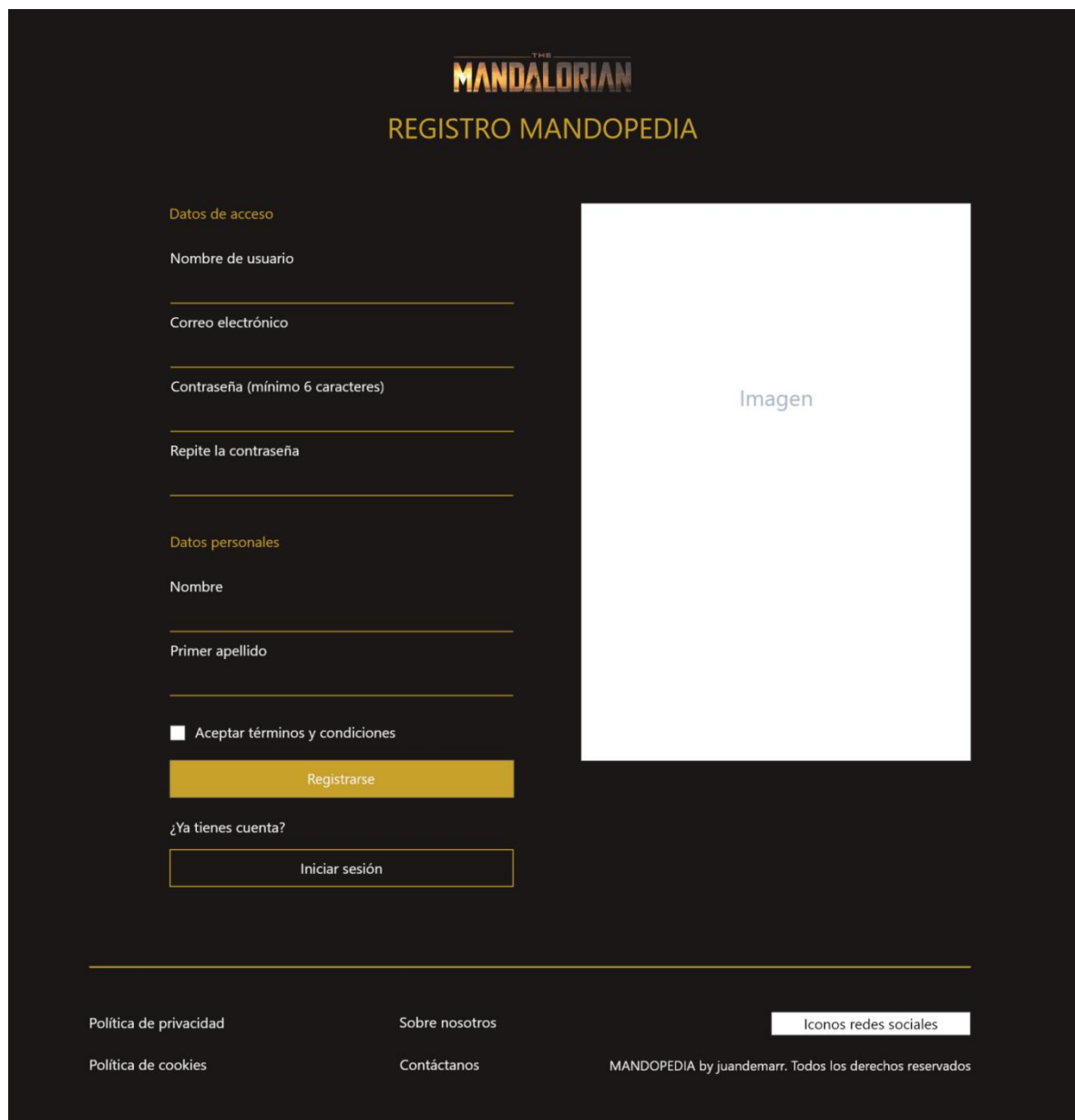


Icono twitter

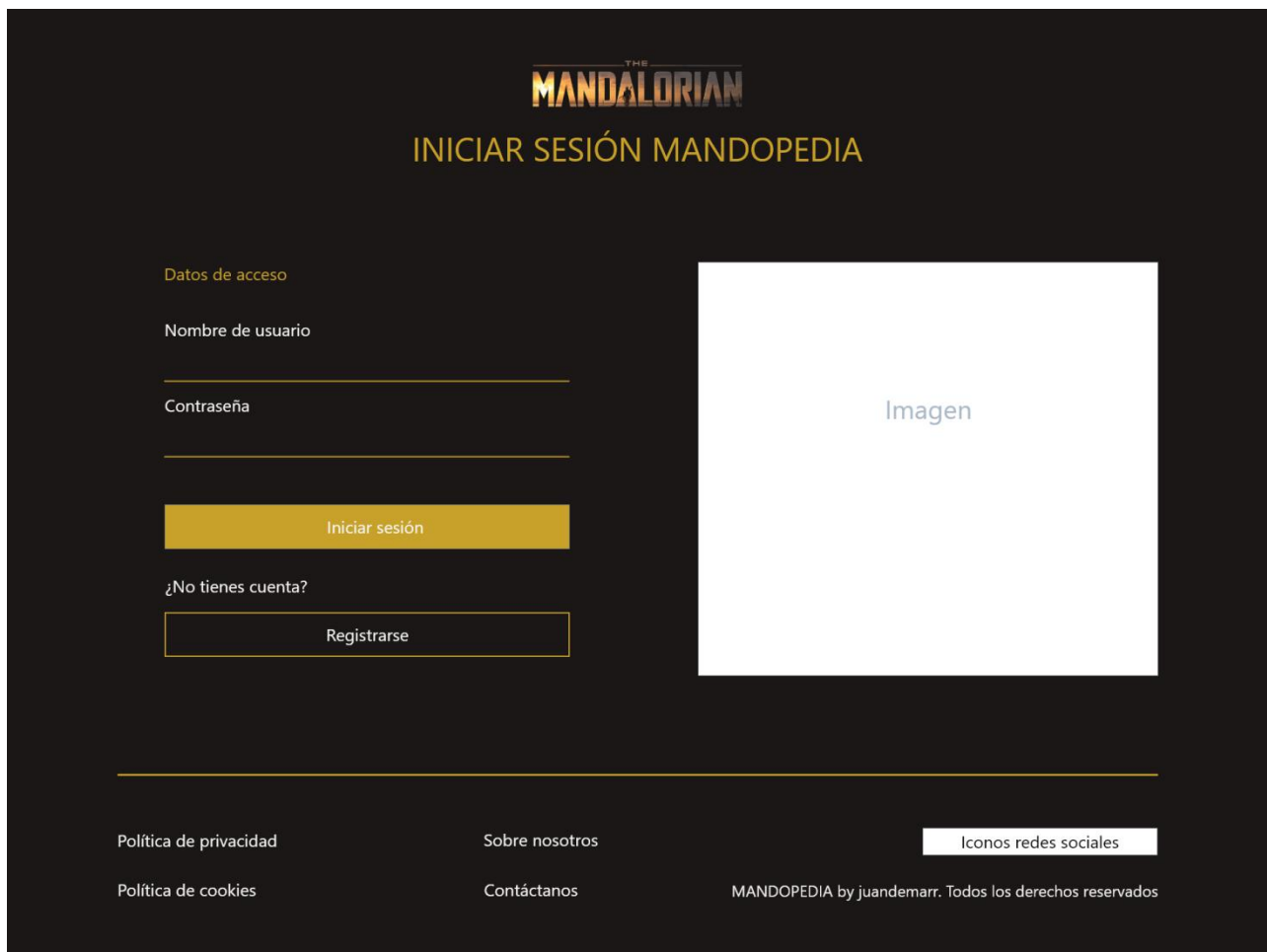
5. MOCKUPS

Los mockups son las plantillas que tienen los diseños finales de las distintas vistas de la aplicación, antes de empezar a programarla.

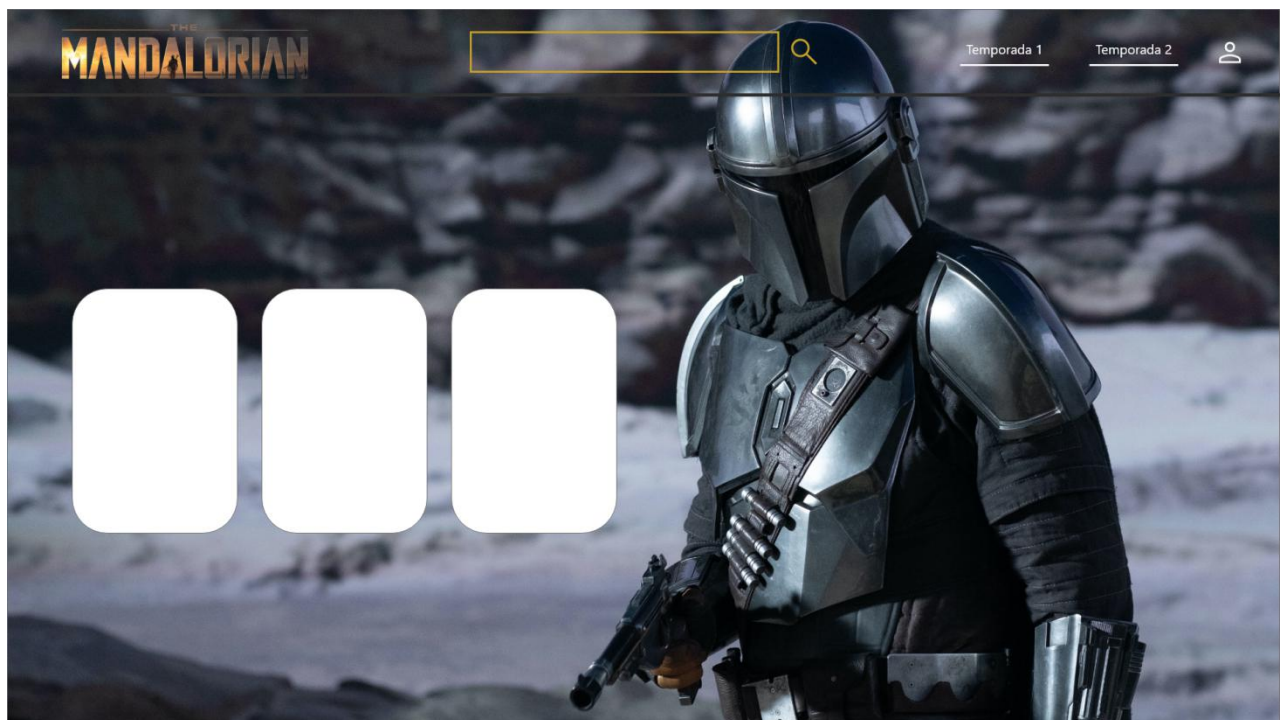
Realizado con Adobe XD.



2. Mockup registro



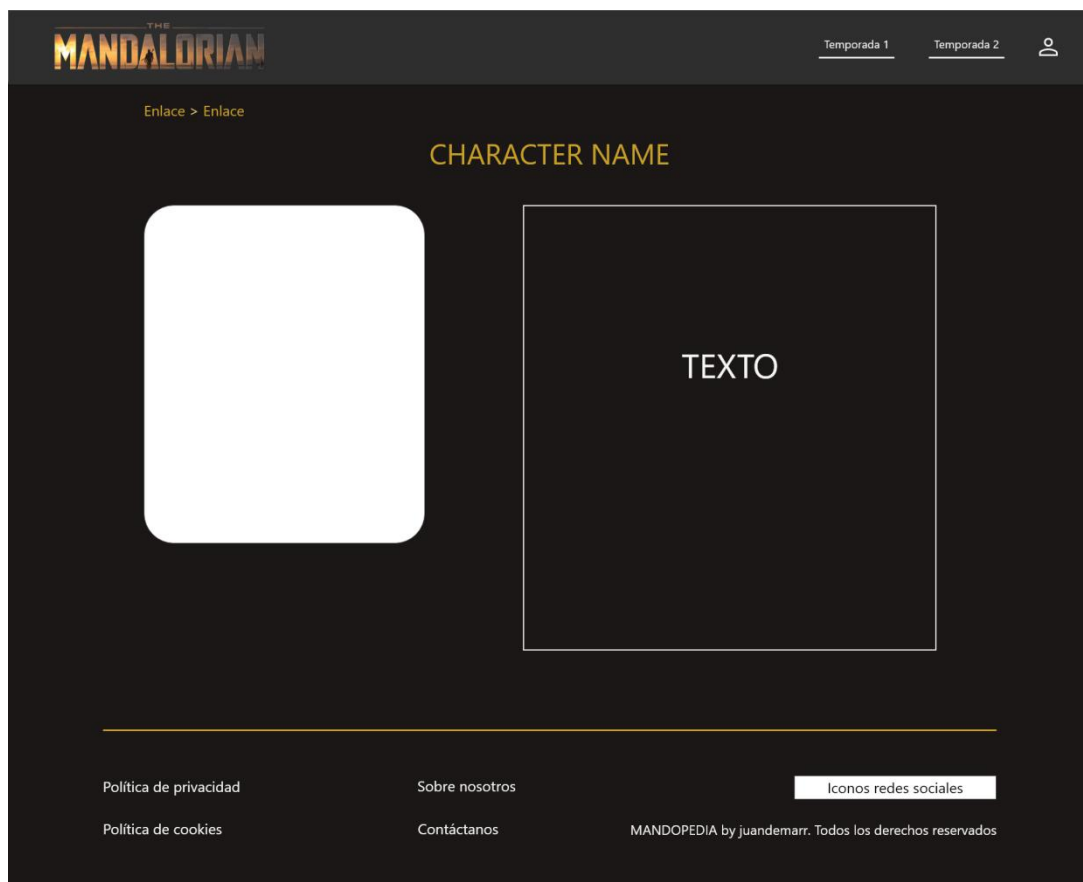
3. Mockup login



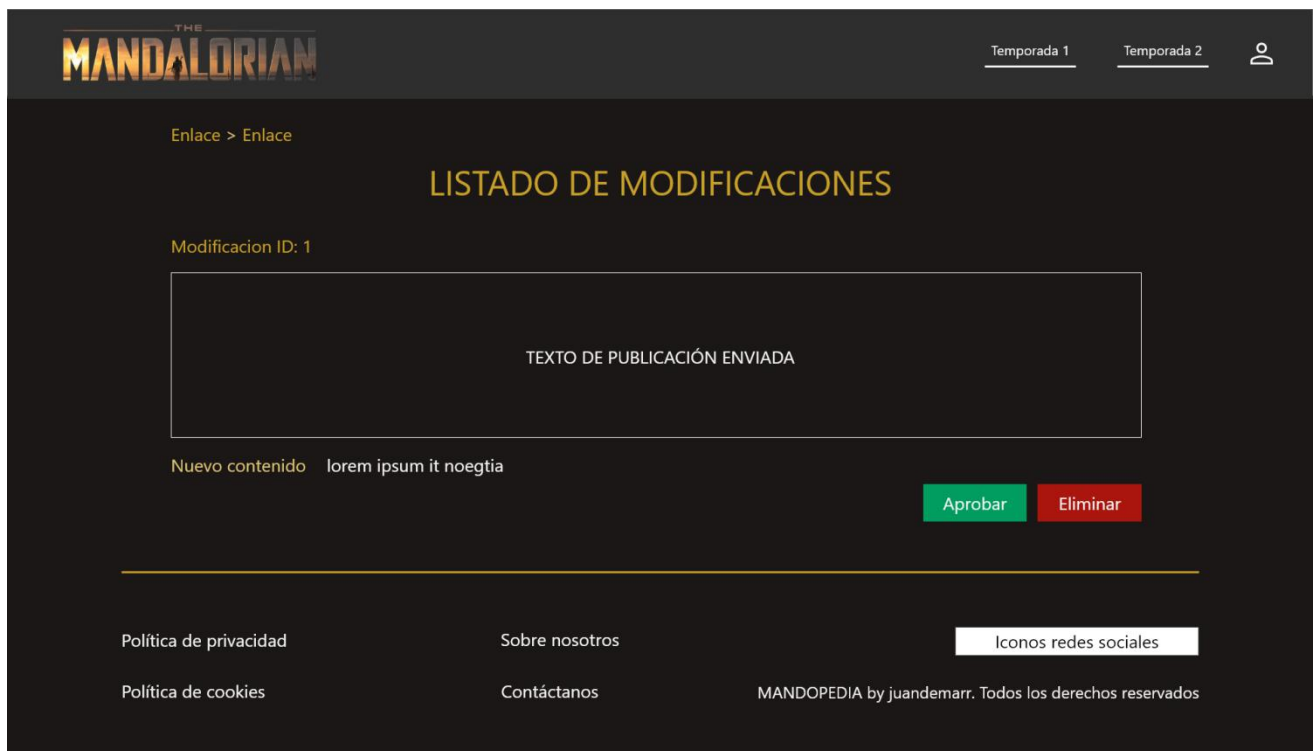
4. Mockup vista principal



5. Mockup lista de publicaciones de una categoría



6. Mockup publicación



7. Mockup listado de modificaciones



8. Menú de usuario en móvil y pc

THE MANDALORIAN

Temporada 1Temporada 2

Enlace > Enlace

EDITAR PERFIL

Nombre de usuario

Nombre

Primer apellido

Imágen de perfil

Correo electrónico

Contraseña (mínimo 6 caracteres)

Repite la contraseña

Guardar

Eliminar

Imagen

Política de privacidad

Política de cookies

Sobre nosotros

Contáctanos

Iconos redes sociales

MANDOPEDIA by juandemarr. Todos los derechos reservados

9. Mockup editar perfil



10. Mockup cookies y terminos de uso

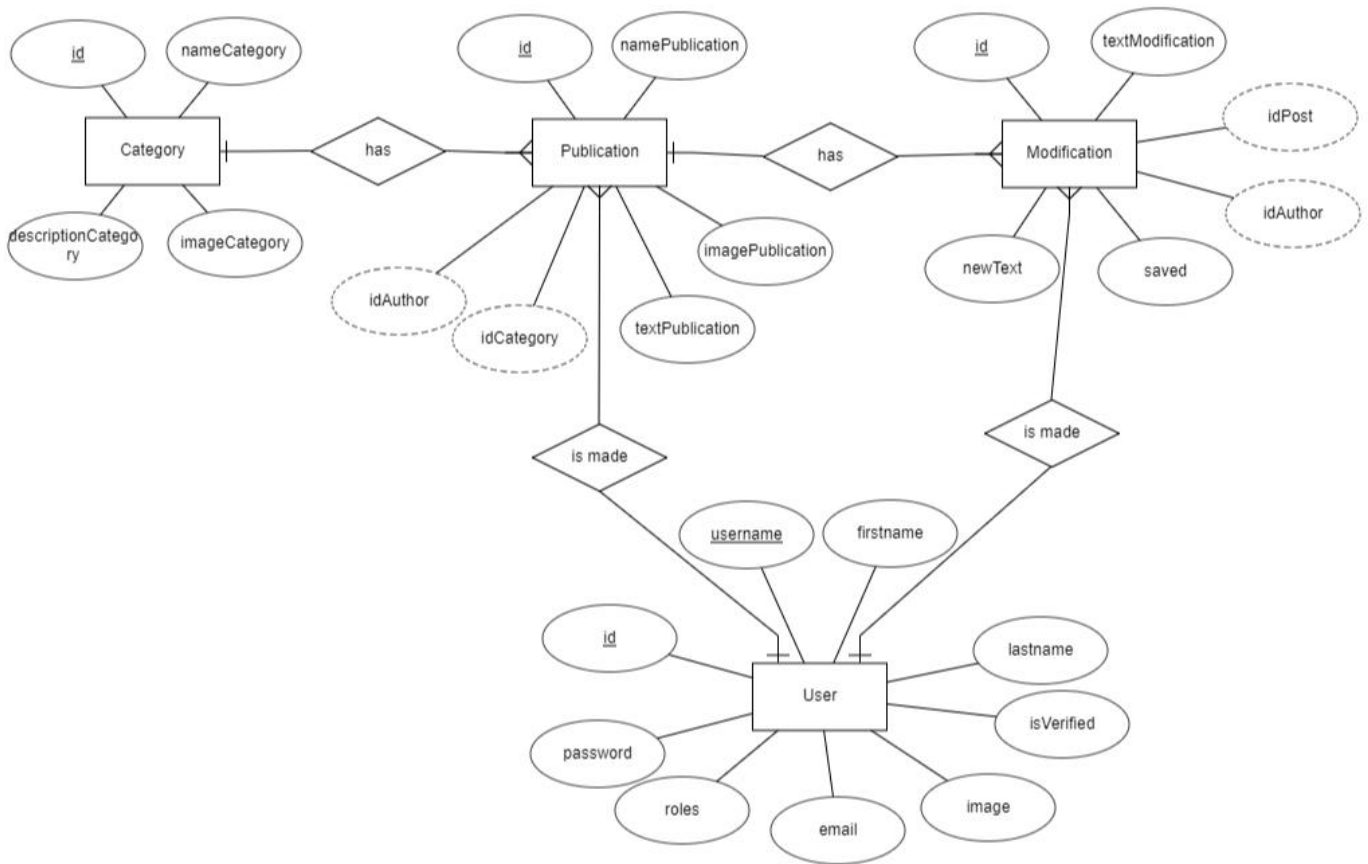
La vista de “sobre nosotros” y “condiciones de uso” es igual a la de políticas de cookies pero sin la imagen parallax.

6. DISEÑO DE LA BASE DE DATOS

Se utiliza una base de datos relacional implementada en mariadb desde la consola de comandos.

Las entidades son: **publicaciones** las cuales estarán clasificadas en **categorías** (temporada 1 y temporada 2), **modificaciones** hechas por los usuarios sobre cada publicación, y **usuarios**.

6.1. DIAGRAMA ENTIDAD-RELACIÓN



11. Diagrama del modelo entidad-relación, realizado con ERDPlus

6.2. PASO A TABLAS

Category (id, nameCategory, descriptionCategory, imageCategory)

Publication (id, namePublication, imagePublication, textPublication, idCategory, idAuthor)

Modification (id, textModification, newText, saved, idPost, idAuthor)

User (id, username, firstname, lastname, email, password, image, roles, isVerified)

7. DESCRIPCIÓN DEL FRONT-END

7.1. TECNOLOGÍAS USADAS

La interfaz ha sido implementada con Materialize, SASS y animate.css

- **Materialize:** Framework CSS desarrollado a partir de Material Design, el cual es un lenguaje de diseño creado por Google, para poder emplear los principios del diseño de la forma más innovadora y proporcionando una experiencia de usuario unificada en todos sus productos.

Materialize fue desarrollado por un equipo de estudiantes de la universidad Carnegie Mellon: Alvin Wang, Alan Chang, Alex Mark y Kevin Louie y se basa en los siguientes principios:

1. **Material es la metáfora.** La tecnología del diseño está inspirada por papel y tinta, y la idea es permitir esa misma creatividad e innovación mediante este framework.
2. Los componentes están diseñados no sólo para ser visualmente agradables, sino para dar una **sensación de jerarquía y significado** también, dando especial énfasis en las acciones que pueden realizar los usuarios con dichos componentes.
3. **El movimiento produce significado.** Este movimiento permite al usuario asociar lo que ve en la pantalla con lo que ve en la vida real, dándole una inmersión en una tecnología desconida. Gracias a la consistencia y continuidad que proporciona el movimiento, le da al usuario un significado inconsciente sobre los objetos y transformaciones que ve en el sistema.

- **SASS:** Preprocesador que permite compilar CSS, es decir, es un lenguaje que mediante sus funciones, mixins, variables y jerarquía permite usar el lenguaje CSS de una forma mucho más sencilla y reutilizable entre distintos

proyectos. Entre sus grandes ventajas se encuentra el anidar las etiquetas para representar la jerarquía del árbol DOM, o utilizar las medias queries dentro de cada etiqueta a la que se le quiera aplicar. También dispone de funciones propias para modificar los colores y facilitar su personalización.

- **Animate.css** es una librería CSS de animaciones “lista para usar” que permite incluir cualquier animación de una forma fácil e intuitiva en los proyectos web. Es ideal para dar énfasis y centrar la atención sobre determinados elementos.

Como lenguajes de programación se ha usado Javascript y jQuery:

- **jQuery**: Librería de Javascript, desarrollada inicialmente por John Resig, que permite la manipulación del DOM, el manejo de eventos, las animaciones y el uso de AJAX de una forma más sencilla que usando Javascript vanilla.

7.2. FUNCIONALIDAD

La primera vez que accedemos al sitio web, aparece un **vídeo a pantalla completa** el cual está implementado de la siguiente forma (ver imagen 12), para que cargue antes que el resto de elementos del árbol DOM, y no sean éstos los que se vean a modo parpadeo precediendo al vídeo.

Su aparición se controla con cookies, implementando sessionStorage para que aparezca siempre que el usuario abra por primera vez el sitio web.

```
window.addEventListener("DOMContentLoaded",function(){
  if(sessionStorage.mandopedia){
    document.getElementsByClassName("video-wrapper")[0].classList.add("d-none");
  }else{
    document.getElementById("video-logo").addEventListener("ended", function(){
      $(this).parent().hide("slow");
      sessionStorage.setItem("mandopedia",true);//only exists in the open tab
    })
  }
})
```

12. Carga de vídeo inicial

Cuando termina, el vídeo desaparece yéndose a la parte superior izquierda de la pantalla.

Las páginas aparecen de forma suave gracias a una animación con la opacidad, de 0 a 1.

En la página principal hay dos tarjetas, una por cada temporada de la serie. Las tarjetas están hechas en 3D aplicándoles una perspectiva y una transformación de rotación cada vez que el ratón pasa por encima de ellas, de la siguiente manera:

```
<a class="a-card" href="category/{{category.id}}">
  <div class="card-element card-size-main">
    
    <div class="frontal">
      <p class="card-element-title">{{category}}</p>
    </div>
    <div class="back">
      <p>{{category.descriptionCategory}}</p>
    </div>
  </div>
</a>
```

13. HTML tarjeta 3D

```
.a-card{
  display:inline-block;

  &:hover .card-element{
    transform:perspective(600px) rotateY(180deg);
  }

  & .card-element{
    border-radius:10px;
    position:relative;
    border:4px solid map.get($color,"white");
    transform-style: preserve-3d; //for the childrens to have their own 3d
    transform: perspective(600px) rotateY(0deg);
    transition: all ease-out .5s;

    & .frontal, & .back{
      width:100%;
      height:100%;
      display:flex;
      justify-content:center;
      align-items:center;
      backface-visibility: hidden; //appear the back of the image
      -moz-backface-visibility: hidden;
      -webkit-backface-visibility: hidden;
    }
  }
}
```

```

& img{
  position: absolute;
  object-fit: cover;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  border-radius: 6px;
}

& .card-element-title{
  z-index: 9;
  position: relative;
  color: map.get($color, "white");
}

& .back{
  position: absolute;
  top: 0;
  color: map.get($color, "white");
  transform: perspective(600px) rotateY(180deg);
  background: linear-gradient(to bottom, □ rgba(255,255,255,0.4), □ rgba(0,0,0,0.5));
}
}
}

```

14. CSS tarjeta 3D

Al hacer clic en una de las tarjetas, aparece una vista con el listado de personajes pertenecientes a esa temporada. Estos personajes también están representados en tarjetas 3D más pequeñas, mediante un grid. Las imágenes de las tarjetas disponen de un aspect-ratio para evitar deformaciones al modificar el tamaño de la ventana.

Cuando se selecciona un personaje, carga una vista con la foto del personaje en 3D, con una perspectiva para darle sensación de profundidad y un balanceo continuo de izquierda a derecha (ver imagen 15 y 16), y al lado aparece el artículo informativo, el cual se puede editar por un usuario registrado en el sistema.

```

<div class="img-publication">
  
</div>

```

15. HTML imagen 3D

```

& .img-publication{
  width:85%;
  margin: 0 auto 2.25rem;
  perspective:700px;

  @include response(m){
    width:38%;
    height: max-content;
    margin:0;
  }

  & img{
    width:100%;
    object-fit:cover;
    border:15px solid map.get($color,"white");
    border-radius:30px;
    aspect-ratio:3/4;
    animation-name: imagePublication;
    animation-duration: 8s;
    animation-iteration-count: infinite;
  }
}

```

16. CSS imagen 3D

En la página de la política de privacidad y de cookies, hay imágenes en parallax. Esto es una funcionalidad que permite a la imagen “desplazarse” a una velocidad distinta que el resto de la página cuando se hace scroll en la misma. Esto está implementado con el componente **parallax** de materialize.

Otros componentes de Materialize que también se han usado son:

- **Formularios.** Colocan el label, o texto descriptivo, encima del input, o entrada de texto, permitiendo que cuando se clique para escribir, el label se mueva hacia la parte superior del input.
- **Botones.** Según el lugar del botón en el que se haga click, aparece un efecto de ola, la cual se desplaza hacia los bordes del botón.
- **Navbar o menú de navegación.** Modifica el menú para vista de móvil o tablet, según el breakpoint definido, y aparece una barra desde la izquierda de la pantalla, ocupando todo el alto de la ventana.

VALIDACIONES DE FORMULARIOS

Mediante expresiones regulares, se comprueba cada entrada escrita por el usuario en los campos del formulario, con los requisitos a cumplir, como el tamaño del nombre, el formato del email, o la longitud de la contraseña, por ejemplo. Una vez se cumplan todos los requisitos, el botón del formulario se activa y ya se puede enviar.

```
let text = /\w+/i; //Omit the g flag beacuse it saves the state so the first test()
//will return true and the second on the same expression will return false
let password = /\w{6,}/i;
let email = /[a-zA-Z-._0-9]+@[a-z0-9-]+\.[a-z]{2,4}$/i;
```

17. Expresiones regulares empleadas

```
$("#loginUsername").on("keyup",function(){
    loginUsernameValidate = validateInput(text,$("#loginUsername")); //only change the styles of the input typping
    validateLoginForm(loginUsernameValidate,loginPasswordValidate);
});

$("#loginPassword").on("keyup",function(){
    loginPasswordValidate = validateInput(password,$("#loginPassword"));
    validateLoginForm(loginUsernameValidate,loginPasswordValidate);
});
```

18. Gestión campos del formulario Login

```
/**
 * Validate the value of an input
 * @param {Object} exp - The RegExp object to validate with
 * @param {Object} tagName - The DOM element obejct for validate
 * @returns {Boolean}
 */

function validateInput(exp,tagName){
    if(exp.test(tagName.val())){ //test the input
        tagName.removeClass("error correct"); //to avoid repeated classes
        tagName.addClass("correct");
        return true;
    }else{
        tagName.removeClass("error correct");
        tagName.addClass("error");
        return false;
    }
}
```

19. Función que aplica las clases de estilo al input

```

/**
 * If the inputs of the login form are valid, the button is activated
 * @param {Boolean} loginUsernameValidate
 * @param {Boolean} loginUsernamePassword
 */
function validateLoginForm(loginUsernameValidate,loginUsernamePassword){
    if(loginUsernameValidate && loginUsernamePassword)
        $("#login button[type='submit']").removeClass("disabled");
    else
        $("#login button[type='submit']").addClass("disabled");
}

```

20. Función que habilita o deshabilita el botón

AJAX

Cuando el usuario con el rol admin o writter inician sesión, en caso de haber modificaciones realizadas por los usuarios, aparecerá una notificación que al clicar, mostrará la página de las modificaciones (también se puede acceder a través del menú de navegación).

Aquí, las modificaciones se pueden aceptar o borrar. Si se aceptan, se hace un envío por medio de Ajax a una ruta definida en el controlador de symfony, el cual se encarga de gestionar la sustitución de la publicación original por esta nueva, guardando los datos necesarios de la misma, sin recargar la página.

```

$("#listModification .button-container a").on("click",function(e){
    let idPost = $(e.target).attr("data-id-post");
    let idMod = $(e.target).attr("data-id-mod");

    $.ajax({
        url:"/publication/update/" + idPost + "/" + idMod
    })
    .done(function(){
        $("#"+idMod).hide();
        infoNotification("Publicación actualizada correctamente");
    })
    .fail(function(){
        errorNotification("No se ha podido actualizar la publicación");
    })
})

```

21. Envío de la modificación por AJAX

SISTEMA DE NOTIFICACIONES

Se ha implementado un sistema de notificaciones en el sitio web por medio de Noty.

Esta librería permite definir notificaciones con distintos temas, colores, barra de progresión hasta que desaparezca la notificación (en caso de haber definido un timeout), etc.

Las notificaciones se utilizan para: indicar que el usuario se ha registrado con éxito o ha habido algún error; señalar el error, en caso de existir, al iniciar sesión; mostrar que se ha enviado con éxito la modificación de la publicación e indicar al usuario con rol admin o writer que hay nuevas modificaciones.

Se disponen de 4 tipos distintos de notificaciones: error (ver imagen 22), advertencia, éxito, y con un enlace al hacer clic (ver imagen 23).

```
function errorNotification(error){  
  new Noty({  
    type:"error",  
    theme:"metroui",  
    text:error,  
    timeout:"2500",  
    progressBar:true,  
  }).show();  
}
```

22. Notificación de error

```
function showModifications(){  
  new Noty({  
    type:"warning",  
    theme:"metroui",  
    text:"Hay solicitudes de nuevas modificaciones",  
    timeout:"2000",  
    progressBar:true,  
    callbacks:{  
      onClick:function(){  
        window.location = '/modification';  
      },  
    },  
  }).show();  
}
```

23. Notificación con función

Como último apartado del front, hay un reajuste automático del textarea en el cual se escribe la modificación de la publicación, por medio de jQuery, cada vez que el usuario escribe.

```
try{ //in other page this doesn't exist
    $("#textarea-modification")[0].style.height = ($("#textarea-modification")[0].scrollHeight + 5 + "px";

    $("#textarea-modification").on("input",function(){
        $(this)[0].style.height = "auto";
        $(this)[0].style.height = $(this)[0].scrollHeight + 5 + "px";
    });
}catch(e){}
```

24. Reajuste del cuadro de texto al escribir

8. DESCRIPCIÓN DEL BACK-END

8.1. TECNOLOGÍAS USADAS

La programación del lado del servidor ha sido implementada con Symfony y distintos bundles:

- Symfony: Framework de PHP y conjunto de componentes reusables de PHP, que permiten construir un sitio web desde cero y mantenerlo en el tiempo. Se basa en el patrón modelo-vista-controlador, en donde los modelos se encuentran en las carpetas entity y repositories, las vistas en templates y los controladores en controllers, todos dentro de la carpeta src del framework.

Bundles usados de symfony:

- Easyadmin: paquete que se encarga de gestionar las entidades de la base de datos, creando una vista estilizada con todas las funciones necesarias para el CRUD (create, read, update y delete) de cada entidad. Completamente personalizable, desde gestionar los permisos de lo que puede hacer cada tipo de usuario, hasta indicar los campos que deben

aparecer por entidad, o las columnas que deben ocupar las entradas de texto del formulario.

- VerifyUserEmail. Paquete que permite crear un enlace único cuando el usuario se registra, y que al ser clicado por el mismo usuario, se cambie en la base de datos el atributo isVerified, de la entidad user, a verdadero y ya pueda iniciar sesión en el sitio web.
- Mailer. Paquete que permite crear correos electrónicos para su posterior envío. Con una serie de funciones que se describen más adelante y un correo electrónico, se pueden enviar mensajes desde texto plano hasta plantillas hechas con HTML y CSS.
- GoogleMailer. Paquete que indica el “transporte” a usar por symfony para el envío de correos electrónicos. Se configura con una cuenta de gmail.

** Por desgracia, el 30 de Mayo de 2022, la opción de permitir el acceso de aplicaciones menos seguras a Gmail fue deshabilitado por Google, impidiendo su uso para enviar emails, por ejemplo, por aplicaciones de terceros.*

8.2. FUNCIONALIDAD

Cuando el usuario entra al sitio web, puede ver toda la información de los distintos personajes de la serie The Mandalorian, puede iniciar sesión o registrarse.

Si elige registrarse, y ha rellenado todos los campos del formulario correctamente, se le enviará un correo electrónico al email proporcionado, para verificar que dicho email es efectivamente el de ese usuario. Este enlace es generado por el bundle verifyUserEmail, y caduca en una hora.

Hay que configurar una variable global, MAILER_DSN, en el archivo .env, en donde se indica el servicio de transporte para el envío de emails que vamos a usar.

```
try{
    // generate a signed url and email it to the user
    $this->emailVerifier->sendEmailConfirmation('app_verify_email', $user,
        (new TemplatedEmail())
            ->from(new Address('mando.mandopedia@gmail.com', 'Mandopedia - Admin'))
            ->to($user->getEmail())
            ->subject('Por favor verifica tu email de Mandopedia')
            ->htmlTemplate('registration/confirmation_email.html.twig')
    );
    // do anything else you need here, like send an email
} catch(\Exception $e){ //in case there isn't an internet connection
    $entityManager->remove($user);
    $entityManager->flush();

    // for the validation on js the fields must be focused, so better is to redirectToRoute
    return $this->redirectToRoute('app_register',["error" => "No se ha podido enviar el correo electrónico de verificación"]);
}

return $this->redirectToRoute('app_login',["warning" => "Se ha enviado un enlace de verificación a tu correo electrónico"]);
```

25. Creación del email



26. Email de verificación enviado

Una vez que el usuario ha verificado su dirección de correo electrónica, ya puede iniciar sesión.

Al iniciar sesión, si el usuario ha olvidado la contraseña, puede recuperarla indicando su dirección de correo electrónico, a donde le llegará una nueva contraseña.

```
if (isset($_POST['recoverPasswordEmail'])) {  
  
    $user = $userRepo->findUserByEmail($_POST['recoverPasswordEmail']);  
    $newPassword = substr(md5(microtime()), 1, 8);  
  
    $email = (new TemplatedEmail())  
        ->from(new Address('mando.mandopedia@gmail.com', 'Mandopedia - Admin'))  
        ->to($_POST['recoverPasswordEmail'])  
        ->subject('Recuperar la contraseña en Mandopedia')  
        ->htmlTemplate('registration/recover_password_email.html.twig')  
        // pass variables (name => value) to the template  
        ->context([  
            'password' => $newPassword,  
        ]);  
}
```

27. Creación de contraseña y envío al email

En caso de que no se pudiera enviar el email, la contraseña no quedaría guardada.



28. Email de recuperación de contraseña enviado

En la página de un personaje, el usuario registrado puede editar dicho artículo. De esa modificación, no sólo se guarda todo el texto, sino que también se compara con el texto original para saber en qué carácter empieza y en qué carácter termina dicha modificación, y guardar además sólo el texto que ha sido añadido. Así, cuando un usuario con el rol necesario vea las modificaciones hechas, que pueda visualizar tanto el texto entero de la publicación cambiada, como lo añadido por separado.

```
$textPost = $publication->getTextPublication();
$textMod = $_POST["textModification"];

$firstIndex = strlen($textPost);
for($i = 0, $j = 0; $i < strlen($textPost) and $j < strlen($textMod) ; $i++, $j++){
    if($textPost[$i] != $textMod[$j]){
        $firstIndex = $i;
        break;
    }
}

for($i = strlen($textPost) - 1, $j = strlen($textMod) - 1; $i >= 0, $j >= 0 ; $i--, $j--){
    if($textPost[$i] != $textMod[$j]){
        $lastIndex = $j;
        break;
    }
}

$newText = substr($textMod,$firstIndex);
$newText = substr($newText, 0, $lastIndex - $firstIndex + 1);
```

29. Separación del nuevo texto añadido

Si el usuario con el rol “admin” o “writer” decide aprobar la modificación, ésta se envía por AJAX, como se menciona anteriormente, y se actualiza la publicación, cambiando el valor del atributo de la modificación “isSaved” a verdadero.

EASYADMIN

En el panel de administración, realizado con easyadmin, el usuario con rol “writer” sólo tiene visible en el menú el elemento “publicaciones” para crear, y el elemento “modificaciones”.

Ésto se consigue cambiando los permisos de la siguiente manera:

```
public function configureMenuItems(): iterable
{
    //yield MenuItem::linkToDashboard('Dashboard', 'fa fa-home');
    yield MenuItem::linkToCrud('Publicaciones', 'fa fa-tags', Publication::class)->setPermission("ROLE_WRITER");
    yield MenuItem::linkToCrud('Categorías', 'fa fa-tags', Category::class)->setPermission("ROLE_ADMIN");
    yield MenuItem::linkToCrud('Modificaciones', 'fa fa-tags', Modification::class)->setPermission("ROLE_WRITER");
    yield MenuItem::linkToCrud('Usuarios', 'fa fa-tags', User::class)->setPermission("ROLE_ADMIN");
    yield MenuItem::linkToUrl('Inicio', 'fa-solid fa-house-chimney', '/')->setLinkRel('app_main');
}
```

30. Permisos de los elementos del menú

También se pueden habilitar o deshabilitar las “acciones” que pueden realizar los usuarios, según un determinado rol, en los CRUD controllers, o controladores de la entidad de easyadmin.

Estas “acciones” son las diferentes tareas que se pueden realizar en dichos controladores. Son 4: “NEW”, que permite crear la entidad; “INDEX”, para mostrar un listado de objetos de esa entidad; “DETAIL” para ver los detalles de un objeto; y “DELETE” para eliminar el objeto.

```
public function configureActions(Actions $actions): Actions
{
    return $actions
        ->setPermission(Action::NEW, 'ROLE_WRITER')
        ->setPermission(Action::INDEX, 'ROLE_WRITER')
        ->setPermission(Action::DETAIL, 'ROLE_WRITER')
        ->setPermission(Action::DELETE, 'ROLE_ADMIN')
        ->add(Crud::PAGE_INDEX, Action::DETAIL)
        ;
}
```

31. Permisos de las acciones

Por último, se puede crear una entidad con un valor ya preestablecido, y que no se pueda cambiar en el formulario de creación, por ejemplo, al crear una publicación, que el autor sea el usuario que ha iniciado sesión. Ésto se consigue “seteando” dicha entidad con ese valor y deshabilitando ese campo en el formulario.

```
public function createEntity(string $entityFqn) {  
    $entity = new Publication();  
    $entity->setAuthor($this->getUser());  
    return $entity;  
}
```

32. Asignar un valor predeterminado

En éste caso en particular, ésto tiene una gran desventaja y es que al ser el campo autor obligatorio en la publicación, no se podría borrar el usuario. Por este motivo no se acabó implementando esta funcionalidad.

Indicar que todas las rutas están protegidas para que sólo puedan acceder los usuarios con los permisos adecuados.

9. DESPLIEGUE DE LA APLICACIÓN

El despliegue se ha realizado usando los servicios de AWS, en donde se ha creado una máquina EC2 con ubuntu server 20, instalado Nginx y asociado un dominio a la IP pública del servidor.

Durante la creación de dicha instancia, hay que crear un par de claves pública y privada con la que poder acceder a la máquina EC2. Estas claves serán usadas en PuTTY (programa que permite el acceso remoto a un equipo), y en FileZilla (programa que permite la transferencia de archivos a un equipo remoto), en caso de usar ambas aplicaciones.

La clave generada será .pem, pero se necesita que sea .ppk para usarla en los programas descritos anteriormente. Para ellos se usa PuTTYgen, que creará el archivo .ppk

También hay que crear un grupo de seguridad de red a la instancia, con unas reglas de acceso para indicar qué tipo de conexiones provenientes de Internet se van a permitir que reciba el servidor.

Se permitirá el acceso por HTTP en el puerto 80 a todas las IPs (0.0.0.0/0), HTTPS por el puerto 443 a todas las IPs y SSH por el puerto 22 a todas las IPs también.

Reglas de entrada (1/3)					
<input type="text" value="Filtrar reglas de grupo de seguridad"/>					
Versión de IP	Tipo	Protocolo	Intervalo de puertos	Origen	
IPv4	HTTPS	TCP	443	0.0.0.0/0	
IPv4	SSH	TCP	22	0.0.0.0/0	
IPv4	HTTP	TCP	80	0.0.0.0/0	

33. Reglas de acceso del servidor

Instancias (1) Información							
<input type="text" value="Buscar"/>							
<input type="checkbox"/>	Name	ID de la instancia	Estado de la i...	Tipo de inst...	Comprobación ...	Estado de la ...	
<input type="checkbox"/>	Mandopedia-ubuntu	i-04649ca8eed1442f9	En ejecución	t2.micro	2/2 comprobaci...	Sin alarmas	

34. Instancia creada en AWS

Se debe crear una dirección IP elástica, y asociarla a la máquina para que cuando esta se apague y encienda, no cambie de IP, sino que tenga esta IP fija.

La dirección IP elástica se ha asociado correctamente. La dirección IP elástica 44.205.87.240 se ha asociado a instancia i-04649ca8eed1442f9									
Direcciones IP elásticas (1/1)									
<input type="text" value="Filtrar direcciones IP elásticas"/>									
Dirección IPv4 pública: 44.205.87.240									
<input checked="" type="checkbox"/>	Name	Dirección IPv4 asig...	Tipo	ID de asignación					
<input checked="" type="checkbox"/>	-	44.205.87.240	IP pública	eipalloc-00d943de09fb908d2					

35. IP elástica asociada al servidor

Ahora queda acceder al servidor mediante PuTTY e instalar el entorno de trabajo de Symfony (PHP, mariadb y hacer segura su instalación, composer y symfony, revisando que se cumplen los requerimientos para ejecutarlo) y Nginx.

Se crea la carpeta para subir el proyecto en /var/www/html/mandopedia

Se instala Nginx y se crea el archivo de configuración del sitio web en /etc/nginx/sites-available/mandopedia, indicándole el nombre del servidor, la ruta de la carpeta raíz, y el comando que ejecutará el socket php-fpm.

Para averiguar este último, se escribe sudo service php8.1-fpm status, y se busca el nombre que aparezca en la línea /run/php/

```
ubuntu@ip-172-31-23-72:/etc/nginx/sites-available$ sudo service php8.1-fpm
Usage: /etc/init.d/php-fpm8.1 {start|stop|status|restart|reload|force-reload}
ubuntu@ip-172-31-23-72:/etc/nginx/sites-available$ sudo service php8.1-fpm status
● php8.1-fpm.service - The PHP 8.1 FastCGI Process Manager
   Loaded: loaded (/lib/systemd/system/php8.1-fpm.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-06-10 09:35:19 UTC; 1h 24min ago
     Docs: man:php-fpm8.1(8)
  Process: 3890 ExecStartPost=/usr/lib/php/php-fpm-socket-helper install /run/php/php-fpm.sock /etc/php/8.1/fpm/pool.d/www.conf 81 (code=exited, status=0/0)
 Main PID: 3878 (php-fpm8.1)
   Status: "Processes active: 0, idle: 2, Requests: 0, slow: 0, Traffic: 0req/sec"
    Tasks: 3 (limit: 1145)
   Memory: 10.3M
   CGroup: /system.slice/php8.1-fpm.service
           └─3878 php-fpm: master process (/etc/php/8.1/fpm/php-fpm.conf)
             └─3888 php-fpm: pool www
               └─3889 php-fpm: pool www

Jun 10 09:35:19 ip-172-31-23-72 systemd[1]: php8.1-fpm.service: Succeeded.
Jun 10 09:35:19 ip-172-31-23-72 systemd[1]: Stopped The PHP 8.1 FastCGI Process Manager.
Jun 10 09:35:19 ip-172-31-23-72 systemd[1]: Starting The PHP 8.1 FastCGI Process Manager...
Jun 10 09:35:19 ip-172-31-23-72 systemd[1]: Started The PHP 8.1 FastCGI Process Manager.
```

36. Socket php8.1-fpm

```
GNU nano 4.8 mandopedia
server {
    server_name mandopedia.granadev.es;
    root /var/www/html/mandopedia/public;

    location / {
        # try to serve file directly, fallback to app.php
        try_files $uri /index.php$is_args$args;
    }

    # DEV
    # This rule should only be placed on your development environment
    # In production, don't include this and don't deploy index_dev.php or config.php
    location ~ ^/(index|config)\.php(/|$) {
        fastcgi_pass unix:/run/php/php-fpm.sock ;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
        # When you are using symlinks to link the document root to the
        # current version of your application, you should pass the real
        # application path instead of the path to the symlink to PHP
        # FPM.
        # Otherwise, PHP's OPcache may not properly detect changes to
        # your PHP files (see https://github.com/zendtech/ZendOptimizerPlus/issues/126
        # for more information).
        # Caveat: When PHP-FPM is hosted on a different machine from nginx
        # $realpath_root may not resolve as you expect! In this case try using
        # $document_root instead.
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        fastcgi_param DOCUMENT_ROOT $realpath_root;
    }

    # PROD
    # return 404 for all other php files not matching the front controller
    # this prevents access to other php files you don't want to be accessible.
    location ~ \.php$ {
        return 404;
    }

    listen 80;
    error_log /var/log/nginx/mandopedia_error.log;
    access_log /var/log/nginx/mandopedia_access.log;
}

[ Read 39 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Und
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line   M-E Rec
```

37. Archivo de configuración de mandopedia

Este archivo es el usado en aplicaciones symfony para ser ejecutados en servidores con Nginx.

Antes de poner el nombre del servidor en el archivo previo, hay que asociar la IP del servidor a un nombre de dominio. Este se puede hacer en la página no-ip.

Nombre de host ▲	Last Update	IP / Objetivo	Type	
 mandopedia.sytes.net <small>Active</small>	Jun 10, 2022 06:41 PDT ⓘ	44.205.87.240	A	 Modificar 

38. Creación del nombre del servidor web

Por último, hay que instalar un certificado gratuito de let's encrypt, para poder acceder al sitio web de forma segura usando el protocolo HTTPS.

10. BIBLIOGRAFÍA

<https://materializecss.com/>

<https://sass-lang.com/>

<https://animate.style/>

<https://ned.im/noty/#/>

Diseño (tarjetas 3d)

<https://www.youtube.com/watch?v=grqUusS5ulg>

Transform 3d con perspectiva

<https://www.youtube.com/watch?v=xlixFvHqjSY>

Vídeo al principio

<https://www.youtube.com/watch?v=hCmHXh0ASwo>

Symfony:

<https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/query-builder.html>

➤ Webpack

<https://symfony.com/doc/current/frontend/encore/simple-example.html>

➤ Formularios

<https://symfony.com/doc/current/reference/forms/types.html>

➤ Enviar email:

<https://symfonycasts.com/screencast/symfony-security/verify-email>

<https://symfonycasts.com/screencast/mailer/config-mailcatcher>

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

➤ Easy-admin

<https://symfony.com/bundles/EasyAdminBundle/current/dashboards.html>

<https://symfony.com/bundles/EasyAdminBundle/current/crud.html>

<https://symfony.com/bundles/EasyAdminBundle/current/fields.html>

<https://symfony.com/bundles/EasyAdminBundle/current/actions.html>

Despliegue

<https://www.nginx.com/resources/wiki/start/topics/recipes/symfony/>

<https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-lets-encrypt-on-ubuntu-20-04-es>

GIT

<https://docs.github.com/es/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

Repositorio: <https://github.com/juandemarr/Mandopedia>

Copyright: Juan de Dios C. Todos los derechos reservados.