



Proyecto DAW

Gestión de artículos literarios

Fecha exposición: 9 de Junio de 2020

Alumno: Gloria Prieto Ortega

Tutor del proyecto: Francisco Martín



Índice

1. Introducción.....	3
2. Descripción general.....	4
3. Objetivos.....	5
4. Ideas principales.....	6
5. Camino escogido.....	7
6. Recursos necesarios para su desarrollo.....	8
7. Desarrollo del proyecto.....	10
8. Manual básico de utilización.....	40
9. Dificultades encontradas.....	46
10. Posibles mejoras.....	48
11. Fuentes de Información y recursos utilizados. Webgrafía:.....	50
Anexos.....	52



1. Introducción

En este proyecto expongo los conocimientos adquiridos como estudiante del Grado Superior de Desarrollo de Aplicaciones Web y obtenidos a través de otros medios mencionados en el presente documento, mediante la realización de una web de gestión de artículos literarios.



2. Descripción general

El proyecto va dirigido a un grupo de escritores que buscan una web propia para compartir sus artículos literarios.



3. Objetivos

Objetivo principal: Proporcionar un medio para que escritores (usuarios) compartan escritos, borradores, o cualquier tipo de artículo.

Objetivos específicos:

- La creación de los artículos pueden estimular y desarrollar la habilidad de escritura del usuario.
- A partir de la web se podría dar a conocer al autor.
- Se podrían comercializar los artículos.



4. Ideas principales

Backend:

Como servidor de base de datos escogí mysql.

Para llevar a cabo el proyecto haré uso de Symfony, un framework con PHP.

Frontend:

Con respecto al desarrollo de la parte cliente haré uso del lenguaje de marcado HTML5, hoja de estilos css3 y bootstrap 4. También añadiré algunas imágenes e iconos creados por mí.

Despliegue:

Para este proyecto usaré Heroku (y Travis para revisar que todo va bien). Como servidor de base de datos usaré GearHost.



5. Camino escogido

PHP/Symfony

Para llevar a cabo el proyecto haré uso de Symfony, un framework de PHP que ofrece funcionalidades atrayentes: crear de forma automática entidades, repositorios, formularios, etc, gestionar permisos de forma sencilla.

MySQL

Como servidor de base de datos escogí mysql pues para una aplicación web no es necesario manejar gran cantidad de datos. Otras razones por la que elegí este software:

- Es openSource
- Es uno de los gestores de base de datos con mejor rendimiento.
- Tiene escasos requerimientos, lo cual permite que se use en máquinas virtuales (como este caso).

El proyecto tendrá algunas entidades, de las cuales, las principales harán referencia al autor o usuario y al artículo, el resto servirán para hacer consultas tipo 'filtrar por tipo de artículo', 'mostrar los últimos añadidos al inicio', ...En un principio usaré Apache y phpmyadmin para generar la base de datos.

Bootstrap

Para mejorar la visualización de las respuestas dadas por el servidor usaré bootstrap pues ofrece gran variedad de etiquetas, y me parece un buen complemento para la hoja de estilos. Además, permite hacer diseños responsive.

Despliegue

Para el despliegue usaré Heroku como servidor del proyecto y travis para conectar Heroku con el repositorio de github donde lo alojaré. Al tratarse de un proyecto web pequeño consideré que era la mejor opción.

Al optar por usar mysql como gestor de base de datos opté por usar GearHost como servidor de base de datos. Para introducir datos en la base de datos generada en GearHost, opté por descargarme heidi, un software libre sencillo de usar.

Para usar heidiSQL me descargué un archivo portable, por lo que no tuve que instalar nada.



6. Recursos necesarios para su desarrollo










Hardware:

El equipo del que dispongo es un ordenador portátil que, entre otras características presenta:

- Sistema operativo Windows 10 de 64 bits.
- Procesador Intel® Core™ i5-5200U CPU @ 2.20GHz.
- Memoria RAM (DDR3) de 16GB.
- Disco duro Crucial_CT525MX300SSD1 489GB.

Máquina virtual:

Figura 1: Características máquina virtual

Device	Summary
 Memory	3.0 GB
 Processors	1
 Hard Disk (SCSI)	20 GB
 CD/DVD (SATA)	Auto detect
 Network Adapter	NAT
 USB Controller	Present
 Sound Card	Auto detect
 Printer	Present
 Display	Auto detect

Software:

- Sistemas operativos:
 - Máquina virtual de Ubuntu 18.
 - Windows 10.
- Software de aplicación:
 - Terminal de ubuntu
 - Editor de código: Visual Studio
 - Navegadores de internet: Chrome y Firefox principalmente, entre otros para pruebas puntuales.
 - Gestor de base de datos: PhpMyadmin.
- Software de despliegue:
 - Servidor de proyecto en Heroku.



- Servidor de base de datos en GearHost y gestor de base de datos HeidiSQL.
- El resto de herramientas necesarias para el despliegue son servidores web (Heroku con respecto al proyecto y GearHost para la base de datos).



7. Desarrollo del proyecto

Implantación de software:

Para la realización del proyecto hice uso de una máquina virtual con Ubuntu 18. Dentro de esta, tuve que ir instalando cierto software para facilitar su creación.

Las herramientas que fui instalando son las siguientes:

- **Instalación LAMP**

Como me decidí por usar una máquina virtual de linux para realizar mi proyecto y, habiendome decidido por usar tanto PHP como mySQL opté por instalar aquello necesario para tener una plataforma LAMP.

- **Instalación Git**

Como quiero trabajar con GitHub y heroku, instalé git y creé mi proyecto de symfony en GitHub.

- **Instalación Symfony**

- **Instalación composer**

Permite instalar otros componentes que permitan otorgar distintas funcionalidades a la aplicación.

- **Base de datos y Doctrine**

Symfony no tiene un componente propio que trabaje con bases de datos, sino que está integrado con Doctrine. Este último se encarga de almacenar y extraer objetos de una base de datos, modificarlos y borrarlos.

- **Componentes de Symfony:**

Assets: recurso (una imagen, una hoja de estilo css, un script javascript,...).

Form: Para la creación de formularios symfony provee de un componente propio que facilita su generación y procesamiento.

Security: Para otorgar seguridad.

- **Librerías externas:**

Dompdf: Para la generación de archivos pdf.

- **Despliegue:** Instalación de Heroku y yarm.

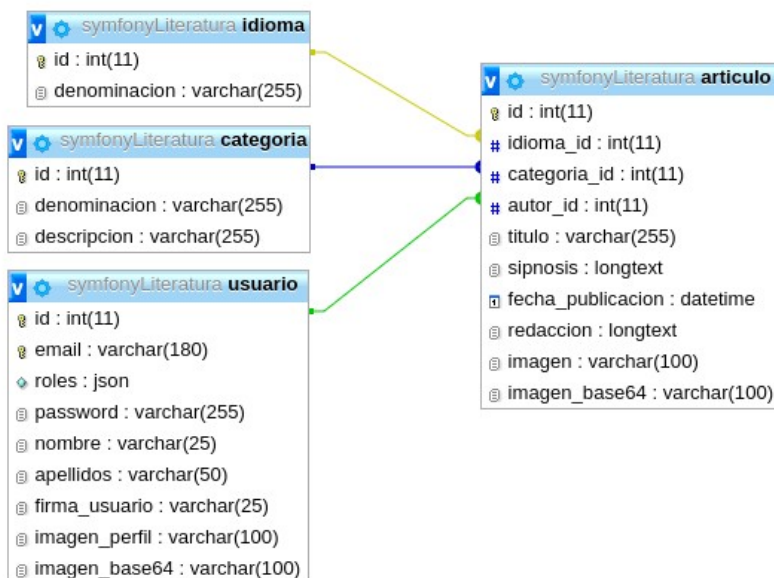


Base de datos: Diagrama E/R y/o modelo relacional. Script SQL de creación de tablas.

El modelo de entidad-relación final por el que opté es el siguiente:

Diagrama E/R (phpmyadmin)

Figura 2: Diagrama E/R



Script

```
-- phpMyAdmin SQL Dump
-- https://www.phpmyadmin.net/
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `symfonyLiteratura`
--
--
```



```
--  
-- Estructura de tabla para la tabla `articulo`  
--  
CREATE TABLE `articulo` (  
  `id` int(11) NOT NULL,  
  `idioma_id` int(11) NOT NULL,  
  `categoria_id` int(11) NOT NULL,  
  `autor_id` int(11) NOT NULL,  
  `titulo` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,  
  `sipnosis` longtext COLLATE utf8mb4_unicode_ci,  
  `fecha_publicacion` datetime NOT NULL,  
  `redaccion` longtext COLLATE utf8mb4_unicode_ci NOT NULL,  
  `imagen` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `imagen_base64` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;  
-- -----  
--  
-- Estructura de tabla para la tabla `categoria`  
--  
CREATE TABLE `categoria` (  
  `id` int(11) NOT NULL,  
  `denominacion` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,  
  `descripcion` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;  
--  
-- Volcado de datos para la tabla `categoria`  
--  
  
INSERT INTO `categoria` (`id`, `denominacion`, `descripcion`) VALUES  
(1, 'Poema', NULL),  
(2, 'Relato corto', 'Relato de no más de 10.000 palabras'),  
(3, 'Terror', 'Relato corto. Busca provocar el escalofrío, la inquietud o el
```



```
desasosiego en el lector.']),  
(4, 'Ficción', NULL);  
  
-----  
  
--  
  
-- Estructura de tabla para la tabla `idioma`  
--  
CREATE TABLE `idioma` (  
  `id` int(11) NOT NULL,  
  `denominacion` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;  
  
--  
  
-- Volcado de datos para la tabla `idioma`  
--  
INSERT INTO `idioma` (`id`, `denominacion`) VALUES  
(1, 'Español'),  
(2, 'English');  
  
--  
  
-- Estructura de tabla para la tabla `usuario`  
--  
CREATE TABLE `usuario` (  
  `id` int(11) NOT NULL,  
  `email` varchar(180) COLLATE utf8mb4_unicode_ci NOT NULL,  
  `roles` json NOT NULL,  
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,  
  `nombre` varchar(25) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `apellidos` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `firma_usuario` varchar(25) COLLATE utf8mb4_unicode_ci NOT NULL,  
  `imagen_perfil` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,  
  `imagen_base64` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;  
  
--  
  
-- Volcado de datos para la tabla `usuario`
```



```
--

INSERT INTO `usuario` (`id`, `email`, `roles`, `password`, `nombre`,
`apellidos`, `firma_usuario`, `imagen_perfil`) VALUES

(1,
                        'admin@gmail.com',
                        '["ROLE_ADMIN"]',
'$argon2id$v=19$m=65536,t=4,p=1$0v7f/4FVioVCv3lSjl7Peg$pCMLZy03sj0BUql3XzBapyx
UZ8WHRlG2iJ0JCbk3H7E', NULL, NULL, 'Admin', NULL);

--

-- Indices de la tabla `articulo`

--

ALTER TABLE `articulo`

  ADD PRIMARY KEY (`id`),

  ADD KEY `IDX_69E94E91DEDC0611` (`idioma_id`),

  ADD KEY `IDX_69E94E913397707A` (`categoria_id`),

  ADD KEY `IDX_69E94E9114D45BBE` (`autor_id`);

--

-- Indices de la tabla `categoria`

--

ALTER TABLE `categoria`

  ADD PRIMARY KEY (`id`);

--

-- Indices de la tabla `idioma`

--

ALTER TABLE `idioma`

  ADD PRIMARY KEY (`id`);

--

-- Indices de la tabla `usuario`

--

ALTER TABLE `usuario`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `UNIQ_2265B05DE7927C74` (`email`);

--

-- AUTO_INCREMENT de las tablas volcadas

--
```



```
--  
  
-- AUTO_INCREMENT de la tabla `articulo`  
  
--  
  
ALTER TABLE `articulo`  
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;  
  
--  
  
-- AUTO_INCREMENT de la tabla `categoria`  
  
--  
  
ALTER TABLE `categoria`  
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;  
  
--  
  
-- AUTO_INCREMENT de la tabla `idioma`  
  
--  
  
ALTER TABLE `idioma`  
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;  
  
--  
  
-- AUTO_INCREMENT de la tabla `usuario`  
  
--  
  
ALTER TABLE `usuario`  
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;  
  
--  
  
-- Restricciones para tablas volcadas  
  
--  
  
-- Filtros para la tabla `articulo`  
  
--  
  
ALTER TABLE `articulo`  
    ADD CONSTRAINT `FK_69E94E9114D45BBE` FOREIGN KEY (`autor_id`) REFERENCES  
`usuario` (`id`),  
    ADD CONSTRAINT `FK_69E94E913397707A` FOREIGN KEY (`categoria_id`) REFERENCES  
`categoria` (`id`),  
    ADD CONSTRAINT `FK_69E94E91DEDC0611` FOREIGN KEY (`idioma_id`) REFERENCES  
`idioma` (`id`);
```



```
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Programación:

El código de mi aplicación se encuentra alojado en un repositorio de GitHub cuya dirección es: https://github.com/GIPrieto/literatura_daw.

Guía de estilo de codificación

Para mi proyecto usé las guías de estilo PSR-2 y PSR-4.

La **guía de estilo PSR-2** es una extensión de PSR-1, estándar básico de estilos de código en php.

De entre sus normas, nos encontramos con:

- El código **debe** seguir una "guía de estilo de codificación" PSR [PSR-1].
- El código **debe** usar 4 espacios para sangría.
- Las líneas deben tener 80 caracteres o menos.
- **Debe** haber una línea en blanco después de la declaración 'namespace', y tras el bloque de declaraciones 'use'.
- Las llaves de apertura y cierre para las clases y los métodos **deben** ir en otra línea.
- **Debe** declararse la visibilidad en todas las propiedades y métodos, y, en caso de ser necesaria su utilización, las palabras clave 'abstract' y 'final' **deben** declararse antes de esta, mientras que 'static' **debe** ser declarada después de la visibilidad.
- Las palabras clave de la estructura de control **deben** tener un espacio después de ellas; las llamadas a métodos y funciones **no pueden tenerlo**.
- Las llaves de apertura para las estructuras de control **deben** ir en la misma línea, y las llaves de cierre **deben** ir en la línea siguiente después del cuerpo.
- Los paréntesis de apertura para estructuras de control **no deben** tener un espacio después de ellos, y los paréntesis de cierre para estructuras de control **no deben** tener un espacio antes.

La guía de estilo PSR-4 es el estándar de Autocarga. Está orientado a los nombres de los Namespaces, Clases y Ficheros. Describe una especificación para las clases de carga automática desde rutas de archivos.



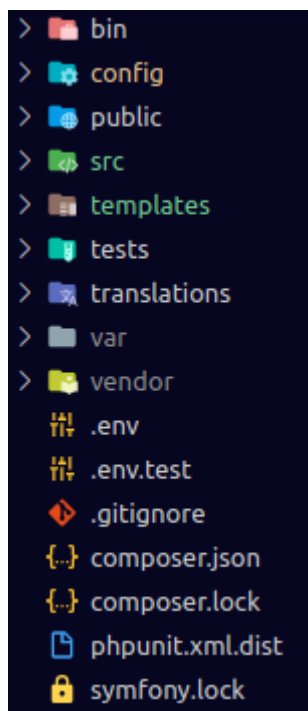
Según lo descrito en esta guía, debe cumplirse:

- Las rutas completas de los Namespaces **deben** tener la estructura Proveedor\Namespace\NombreDeLaClase. (VendorName\Namespace\ClassName)
- Todas las rutas **deben** tener un Proveedor (Vendor Name).
- Los Namespaces pueden estar compuestos de todos los Sub-Namespaces que se desee.
- Todas las rutas **deben** acabar en un Nombre de Clase (Class Name).
- Los guiones bajos no tienen ningún significado especial (modificación sobre PSR-0).
- Se pueden combinar mayúsculas y minúsculas.
- Todos los nombres de clase **deben** usar un estilo 'case-sensitive'.
- Cada namespace puede tener tantos sub-namespaces como se quiera.
- Todos los archivos deben tener la extensión .php.
- Los nombres de los namespaces o clases deben ser ordenados alfabéticamente.

Organización física del código (ficheros)

Los proyectos en Symfony tienen la siguiente estructura:

Figura 3: Estructura



De los cuales, con los que trabajé principalmente son:

- **config**: contenedor de la configuración de rutas, servicios y paquetes.
- **public**: directorio raíz del proyecto. Todo lo de esta carpeta es accesible al usuario.
- **src**: código fuente PHP.
- **templates**: plantillas twig.
- **bin**: archivos binarios (programas), incluyendo bin/console.
- **var**: ficheros que se crean automáticamente, como la caché o los logs.
- **vendor**: librerías externas (frameworks, extensiones...) descargadas con composer.

Clases y funciones

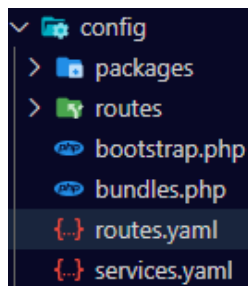


En este apartado hago una breve introducción al código de mi web, indicando de que clases esta compuesto y haciendo referencia a los códigos que contiene.

Para crear una página es necesario definir una ruta (url de la página) y un controlador.

Las rutas hacen referencia a una función dentro del controlador y estas deben definirse en `config/routes.yaml`.

Figura 4: Rutas



Artículo:

- index
- app_articulo_ver
- app_articulo_pdf
- app_articulo_categoria
- app_articulo_idioma
- app_articulo_autor
- app_nuevo_articulo
- app_articulo_creado
- app_articulo_editar
- app_articulo_borrar

Usuario:

- app_perfil_ver
- app_perfil_editar

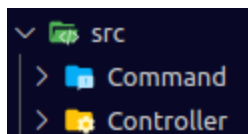
Administrador:

- app_ver_articulos
- app_borrar_articulo_admin
- app_ver_usuarios
- app_borrar_usuario_admin
- app_nueva_categoria
- app_nuevo_idioma

En las clases 'controlador' se establecen las funciones PHP que construyen la página. Recibe la petición del usuario y crea un objeto Response de Symfony, que, en este caso, son archivos '.html.twig'.

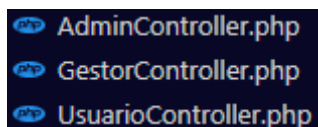


Figura 5: Directorio src



Para la realización de mi proyecto opté por crear controladores para manejar artículos y usuarios. Los controladores de este proyecto son:

Figura 6: Controladores



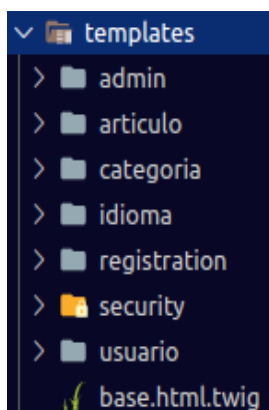
Para la subida de archivos generé un servicio, SubidaArchivos, que es usado tanto en 'GestorController' como en 'UsuarioController'.

Figura 7: Servicios



Para generar la respuesta que se enviará al usuario, Symfony incluye Twig, que es una herramienta para construir plantillas. Estas se crean en el directorio templates, dentro de esta creé distintos directorios para distinguir a cuales hace referencia cada controlador.

Figura 8: Plantillas



Usuarios

Comando 'crear usuario': Para que un administrador pueda crear cualquier tipo de usuario, mediante línea de comandos.

Formulario de registro: Para que se puedan registrar usuarios no administradores en la web.

Formulario de login: Para que los usuarios registrados (administradores o no), puedan



acceder a páginas restringidas de la web.

UsuarioController:

- **Register.** Llama al formulario de registro y guarda los datos introducidos en base de datos.
- **Login.** Llama al formulario de inicio de sesión y, si los datos de usuario introducidos se corresponden con los de un usuario registrado, le permite acceso a otras zonas de la web.
- **Logout.** Permite la desconexión del usuario.
- **Mostrar perfil.** Muestra los datos del usuario y los artículos creados por este. De momento solo está implementado para que el usuario logueado pueda acceder a su perfil.
- **Editar usuario.** En relación con la función anterior, el usuario podrá editar sus datos (email, imagen de perfil, etc).

Plantillas:

- **Registration.** Contiene la plantilla que genera el formulario de registro.
- **Security.** Contiene la plantilla que genera el formulario de inicio de sesión.
- **Usuario.** Contiene las siguientes plantillas:
 - editarPerfil.html.twig: genera el formulario de edición de perfil.
 - verPerfil.html.twig: genera la vista del perfil del usuario.

Artículos

ArticuloVoter: Para restringir acceso a ciertas funciones (editar y borrar), otorgando permisos solo a los dueños del artículo.

GestorController:

En este controlador indiqué cual sería el índice de la web.

- **Index:** Como página principal decidí que se mostrase una lista de artículos, mencionandose la categoría e idioma de cada uno de ellos y su autor.
- Funciones para **filtrar los artículos**, por categoría, por idioma y por autor(usuario). Esto lo hice añadiendo consultas en el repositorio del artículo.
- **Vista de un artículo:** Página en la que, aparte de mostrar otros atributos de la entidad, aparece la redacción del artículo.
- **Ver artículo como archivo pdf:** Genera un pdf a partir de un documento que contiene código html, en el cuál aparece la información de la entidad artículo



mostrada en el apartado anterior.

- **Creación de un nuevo artículo:** Muestra un formulario en el cual se piden datos sobre el artículo.
- **Edición de un artículo:** Muestra un formulario similar al anterior, pero con los campos de texto rellenos.
- **Eliminación de artículo:** Borra el artículo de la base de datos.

Plantillas:

- **Artículo.** Contiene las siguientes plantillas:
 - `editarArticulo.html.twig`: genera el formulario de edición de un artículo.
 - `listaArticulo.html.twig`: genera una vista con la lista de artículos.
 - Filtros:
 - `listaArticuloPorAutor.html.twig`: genera una vista con la lista de artículos de un autor concreto.
 - `listaArticuloPorCategoría.html.twig`: genera una vista con la lista de artículos filtrando por una categoría.
 - `listaArticuloPorIdioma.html.twig`: genera una vista con la lista de artículos que están escritos en el idioma seleccionado.
 - `nuevoArticulo.html.twig`: genera el formulario de creación de un artículo.
 - `verArticulo.html.twig`: genera una vista con el contenido de un artículo.
 - `verArticuloPDF.html.twig`: archivo que contiene la vista con la que se generará el pdf.

Administrador

Formularios para crear categorías e idiomas.

AdminControler:

- **Índice de la página de administrador:** Se muestra los artículos creados recientemente, nuevos usuarios registrados, todas las categorías y todos los idiomas que se han guardado en la base de datos.
- **Vista de artículos:** Se muestra una tabla con todos los artículos que hay en la base de datos.
- **Eliminar artículo:** Un administrador tiene opción de eliminar un artículo de cualquier usuario.
- **Vista de usuarios:** Se muestra una tabla con todos los usuarios que hay



registrados.

- **Eliminar usuario:** Un administrador puede eliminar un usuario desde esta página.
- **Métodos nuevaCategoría y nuevoldioma:** Para poder añadir a base de datos nuevos campos en las tablas categoria e idioma.

Plantillas:

- **Admin.** Contiene las siguientes plantillas:
 - index.html.twig: genera la página de gestión a la que puede acceder un administrador.
 - vistaAdminArticulos.html.twig: genera una vista con una tabla que lista todos los artículos.
 - vistaAdminUsuarios.html.twig: genera una vista con una tabla que lista todos los usuarios.
- **Categoría:**
 - nuevaCategoría.html.twig: genera una vista con el formulario de inserción de categorías.
- **Idioma:**
 - nuevoldioma.html.twig: genera una vista con el formulario de inserción de nuevos idiomas.

Pruebas

Visualizar la página

Para realizar las pruebas iniciaba el servidor de Symfony mediante el comando `symfony server:start`.

Figura 9: Inicio del servidor de Symfony

```
usuario@ubuntu:~/Symfony/literatura_daw$ symfony server:start
May 24 18:45:35 [DEBUG] PHP Reloading PHP versions
May 24 18:45:36 [DEBUG] PHP Using PHP version 7.4.6 (from default version in $PATH)
May 24 18:45:36 [INFO] PHP listening path="/usr/bin/php7.4" php="7.4.6" port=35109
May 24 18:45:36 [DEBUG] PHP started
May 24 18:45:36 [DEBUG] PHP PHP 7.4.6 Development Server (http://127.0.0.1:35109) started

[OK] Web server listening on https://127.0.0.1:8000 (PHP CLI 7.4.6)
```

Si los datos de localización del servidor están bien, se muestra este mensaje, y entrando en la URL que proporciona al iniciarlo, podemos ver lo generado por las plantillas.



Vistas antes de insertar bootstrap

Inicialmente solo creé la vista de los artículos. Copié la carpeta public de un proyecto hecho en clase para poder ver como iba quedando la web con unos estilos específicos.

Figura 10: Página de inicio (sin haber insertado artículos)

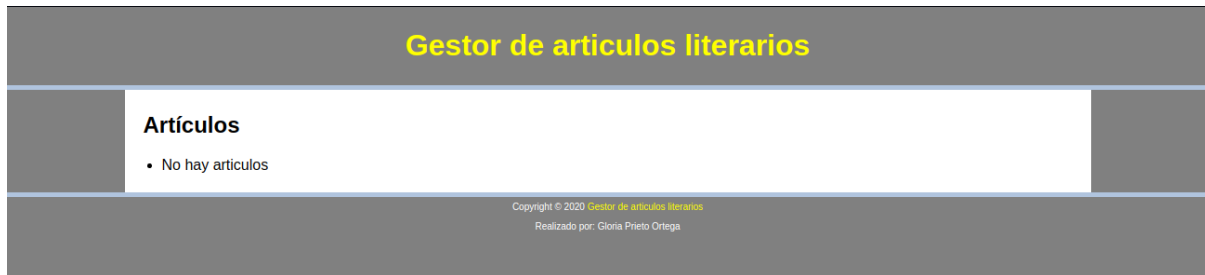


Figura 11: Página de inicio (1 artículo)

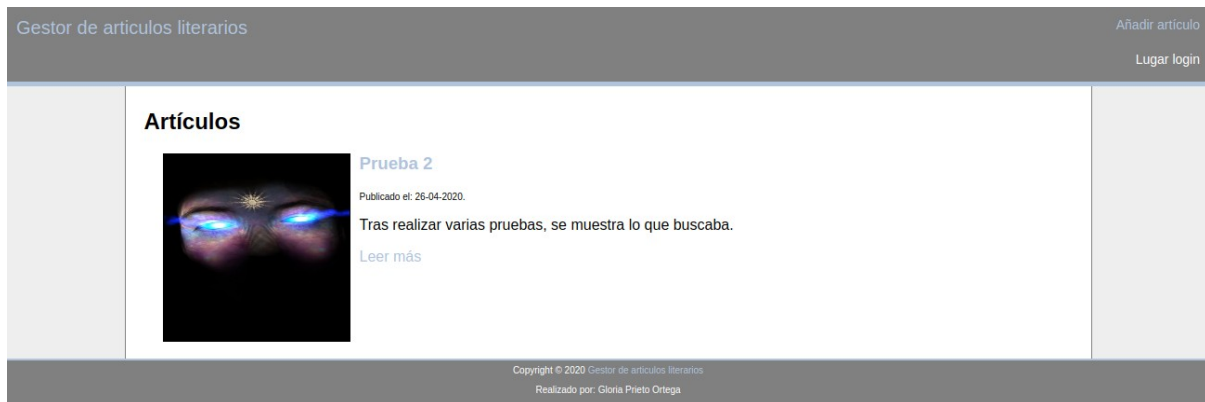
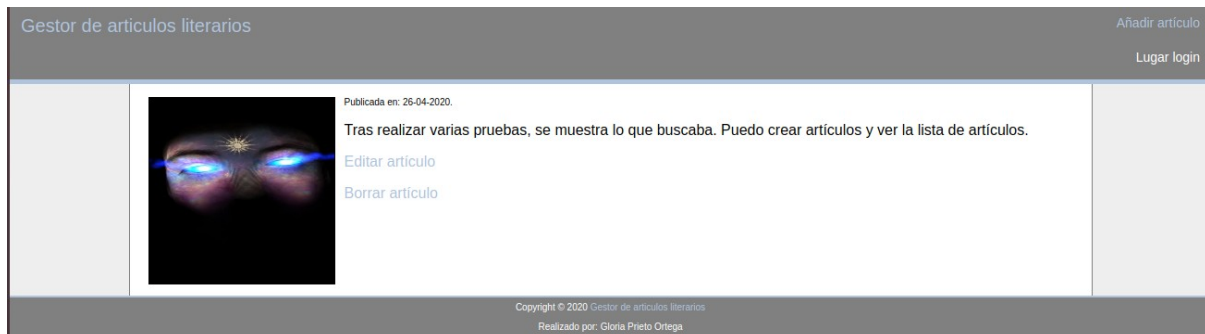


Figura 12: Vista de un artículo



Como esta no me convenció, prové con las siguientes:



Figura 13: Vista de un artículo (mostrando categoría e idioma)

Lorem Ipsum



Categoría: Relato corto. Idioma: Español.
Publicada en: 27-04-2020.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin et scelerisque arcu. Curabitur iaculis ut ligula in congue. Curabitur id libero ut nunc feugiat fermentum in a eros. Vestibulum at leo et quam imperdiet scelerisque. Morbi tincidunt posuere quam vitae dictum. Curabitur dictum eu nulla ut interdum. Aliquam erat volutpat. Nunc fringilla venenatis ante porta fringilla. Suspendisse gravida libero sed felis suscipit, a efficitur lacus scelerisque. Pellentesque lobortis mauris vitae lectus venenatis aliquam. Nunc eget sem viverra, ornare ante vitae, consectetur leo. Donec eu metus venenatis sem lacinia gravida. Cras risus odio, egestas et vulputate quis, imperdiet id odio. In cursus turpis sed aliquam hendrerit. Mauris ultrices arcu felis, eget aliquam metus luctus at. Duis lacinia ac odio a eleifend. Proin semper quam vel pulvinar faucibus. Phasellus in neque eu velit dictum blandit pretium in justo. Donec interdum nulla a rhoncus volutpat. Nulla molestie mi ac lectus efficitur malesuada. Donec interdum ut neque in pharetra. Quisque enim risus, auctor eget ante quis, finibus pulvinar ligula. Nulla tincidunt nisi vel leo rhoncus interdum. Nulla facilisi. Morbi ipsum turpis, convallis sed neque suscipit, vulputate fringilla ipsum. Nam fringilla lacinia dui, quis auctor justo hendrerit a. Duis nulla ipsum, euismod eu est eget, commodo tempor turpis. Fusce feugiat, arcu a auctor vulputate, lorem eros venenatis metus, quis commodo est lacus et mi. Duis sed eros sit amet est dictum molestie non in urna. Vivamus nec felis erat. Maecenas semper arcu auctor mauris portitor consequat. Vestibulum sed nulla mauris. Quisque eget ante non ligula facilisis finibus id quis justo. Proin tempus blandit nunc, at molestie magna iaculis nec. In sagittis, dolor maximus vehicula luctus, diam felis portitor velit, vel pulvinar ligula urna vitae lectus. Donec placerat arcu dignissim vulputate molestie. Cras tristique a dolor ac pharetra. Cras aliquet massa sit amet efficitur tempor. Pellentesque finibus eu elit ac sagittis. Sed vitae ex justo. Nullam interdum accumsan condimentum. Praesent consequat euismod varius. Morbi accumsan augue id cursus luctus. Curabitur tincidunt quam eget ligula tincidunt pretium. Sed venenatis mattis neque, in euismod risus ultrices vitae. In euismod felis et ipsum volutpat, a rutrum dolor sollicitudin. Vestibulum pellentesque erat libero, vel varius metus blandit ac. Suspendisse non nisi in libero consectetur dictum. Curabitur nec leo et lacus vestibulum pretium. Mauris convallis eros facilisis, efficitur ipsum at, semper tortor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilla curae; Curabitur vitae enim a mi efficitur congue ut a orci. Fusce sed finibus sapien. Cras porta lorem posuere ultricies pellentesque. Donec vehicula at tellus eget laoreet. Nam consequat vel nulla et interdum. Nam nec tortor vitae risus malesuada fringilla et id felis. Aliquam erat volutpat. Duis interdum leo at tellus mattis imperdiet.

[Editar artículo](#) [Borrar artículo](#)

Copyright © 2020 Gestor de artículos literarios
Realizado por: Gloria Prieto Ortega

Formulario de creación:

Figura 14: Formulario de crear artículos (sin categoría e idioma)

Gestor de artículos literarios

Añadir artículo

Lugar login

Rellene los campos mostrados a continuación

Título

Prueba 2

Sipnosis

Tras realizar varias pruebas, se

Redaccion

Tras realizar varias pruebas, se

Fecha publicacion

Apr 26 2020

Añadir artículo

Copyright © 2020 Gestor de artículos literarios
Realizado por: Gloria Prieto Ortega

Tras varias pruebas conseguí añadir en el formulario otros dos campos, para que me mostrase que categorías e idiomas había en la base de datos y poder elegir una.



Figura 15: Formulario de crear artículos (con categoría e idioma)

The screenshot shows a web application interface for managing literary articles. At the top, there is a header bar with the title 'Gestor de artículos literarios' on the left and two links, 'Añadir artículo' and 'Lugar login', on the right. Below the header, the main content area has a heading 'Rellene los campos mostrados a continuación'. Under this heading, there is a form with several input fields: 'Titulo' (a text box), 'Sinopsis' (a text box with a diagonal line icon), 'Redaccion' (a text box with a diagonal line icon), 'Fecha publicacion' (a date picker showing 'Jan 1 2015'), 'Idioma' (a dropdown menu showing 'Español'), and 'Categoría' (a dropdown menu showing 'Poema'). Below these fields is a button labeled 'Añadir artículo'. At the bottom of the form, there is a footer bar with the text 'Copyright © 2020 Gestor de artículos literarios' and 'Realizado por: Gloria Prieto Ortega'.

Tras esto conseguí que mostrase el nombre del autor:

Figura 16: Inicio (mostrando lista de artículos y su autor)

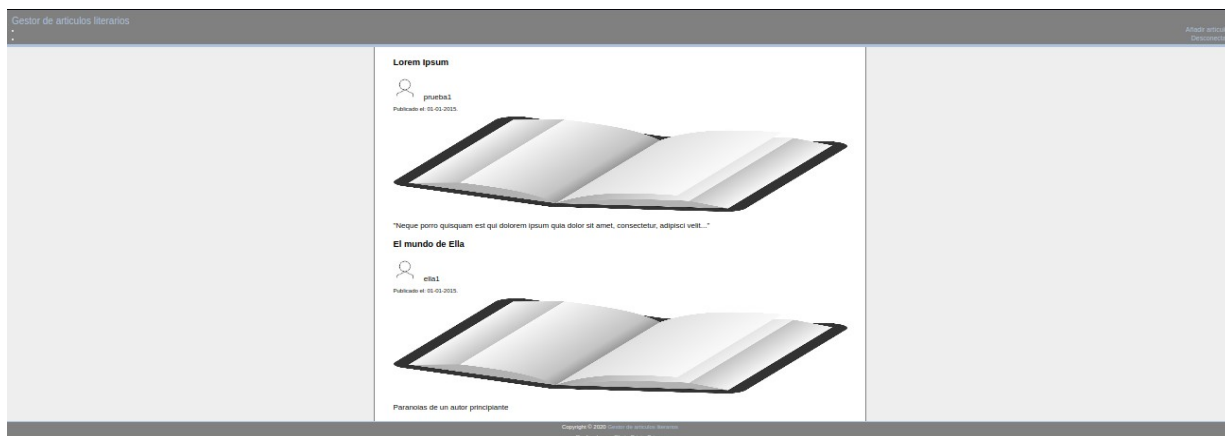


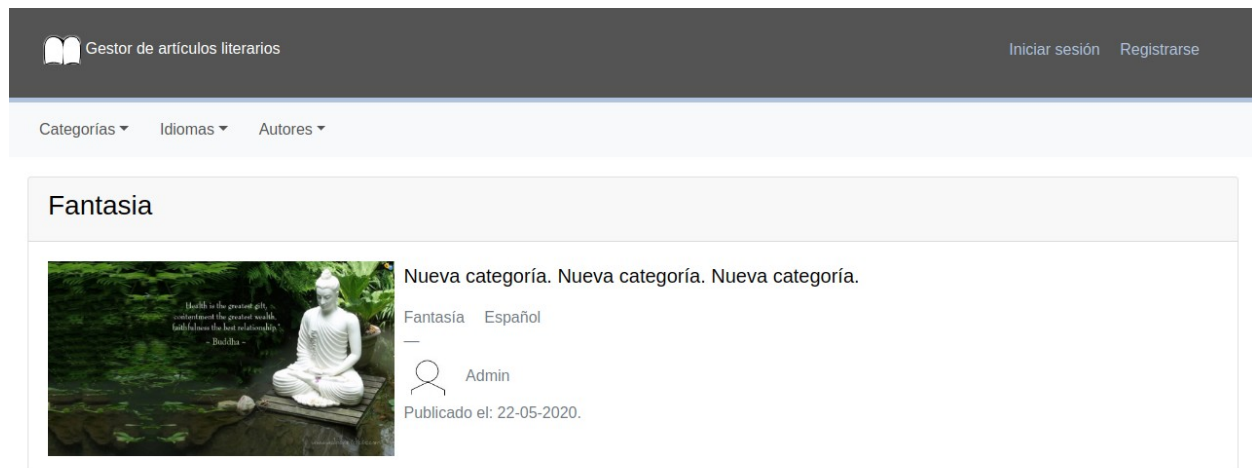


Figura 17: Vista de un artículo (Con autor)



Cuando estuve conforme con las clases y los métodos, me centré más en el frontend. Busqué otorgarle una apariencia más elegante y ordenada a toda la página, insertando en la plantilla base un assert con el enlace a los recursos de bootstrap.

Figura 18: Vista página de inicio (con bootstrap)



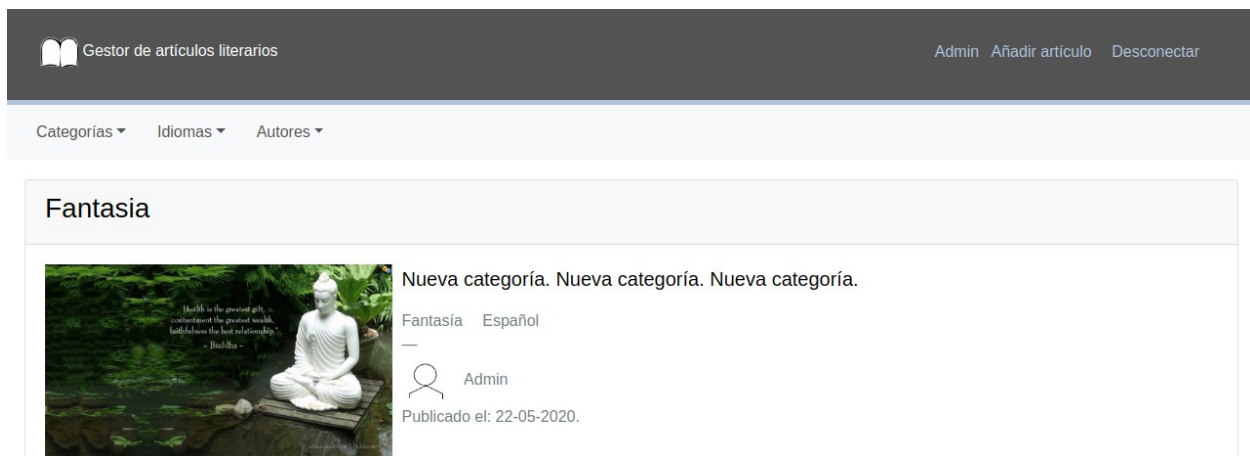
Para ver si se creó el usuario administrador inicio sesión con los datos introducidos:



Figura 19: Inicio de sesión (con bootstrap)

Una vez iniciada la sesión podemos ver que cambió la barra de navegación superior.

Figura 20: Vista página principal tras inicio de sesión (con bootstrap)



Más detalles de esta última versión en el manual de uso.

Crear usuario administrador

Para crear un usuario en el terminal, llamo al comando que creé: **php bin/console app:create-user email contraseña firma [--admin]**

Figura 21: Creación usuario administrador

```
usuario@ubuntu:~/Symfony/proyecto_literatura$ php bin/console app:create-user admin@gmail.com administrador admin --admin
Creador de usuario
El usuario fue creado
```

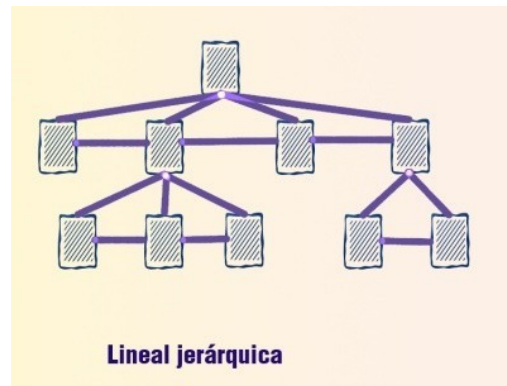
Desarrollo web:

Estructura del sitio web.

Como estructura de sitio web opté por una lineal jerárquica, pues permite que desde distintos sitios se pueda acceder a otros, incluso al índice.



Figura 22: Estructura de la web



Maqueta del sitio web.

La estructura de navegación variará en cada página. Según la página de la web que se visite, se mostrarán leves modificaciones.

Para las páginas de inicio, perfil, las que muestran artículos y las de administración de usuarios y artículos tienen un contenido similar, sufriendo variaciones leves en el contenido de la etiqueta nav y la etiqueta container:

Tabla 1: Maquetación 1

Header
Nav
Contenedor
Footer

Mientras que las páginas que contienen formularios y la página principal de administrador no tendrá etiqueta nav fuera del header, quedando de la siguiente forma:

Tabla 2: Maquetación 2

Header
Contenedor
Footer

Manual de diseño.



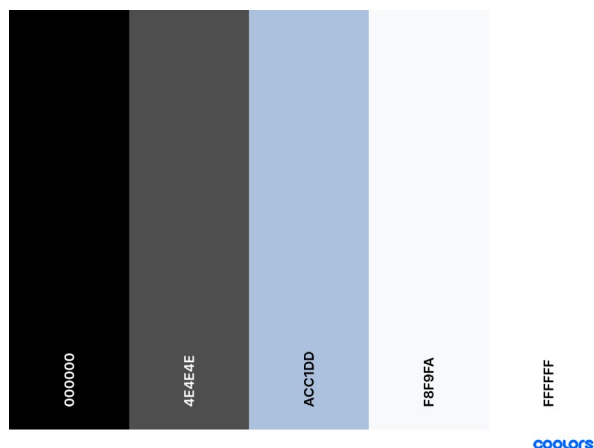
Tipografía

Como tipografía opté por fuentes sin serifa, pues son más recomendables a la hora de leer una web. Las fuentes que indiqué al proyecto que debía coger son: Verdana, sans-serif para la web y Arial para cuando se generan archivos pdf.

Colores

Mi objetivo con esta página era mostrar seriedad, para ello busqué usar, a parte del blanco, tonalidades de grises y negro. Por darle unas gotas de color opté por coger un gris azulado. No quise pasarme en este sentido, pues quiero que destaquen los artículos y no los colores de la web.

Figura 23: Código de colores



Imágenes

Creación del icono.

Tamaño: 16x16px.

Figura 24: Creación de icono en <https://www.favicon.cc/>

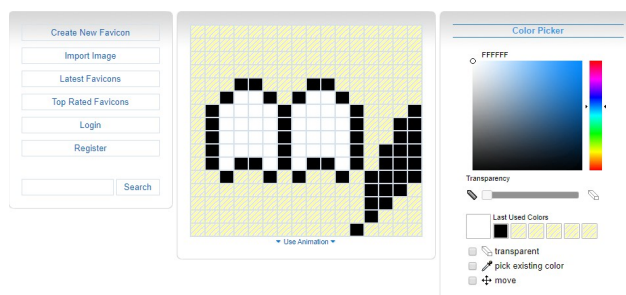


Figura 25: Icono (favicon.ico)





Creación del logo.

A partir de la transformación de una imagen png a svg, obtuve un logo. Pero como quería que tuviera fondo transparente tuve que desechar el vector y crear una nueva imagen, quitándole el fondo.

Figura 26: Creación logo en <https://quitarfondo.com/index.html>

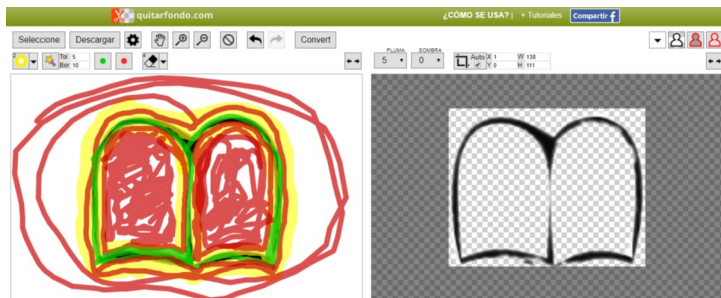


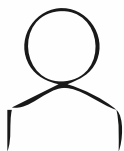
Figura 27: Logo



Tamaño: 39x31px.

Imágenes de perfil. Con respecto al usuario no puse varias opciones de imágenes, pero creé una que apareciera por defecto.

Figura 28: Usuario



Tamaño: 73x73px.

Imágenes de artículos. Para otorgar algo de distinción entre artículos, puse una imagen provisional por defecto que saliese según la categoría elegida.

- Tamaño: 170x280.
- Extensión: svg.



Figura 29: Terror



Figura 30: Relato corto

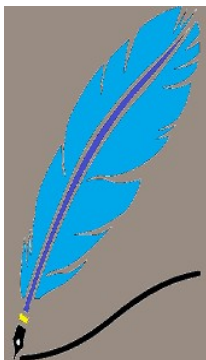


Figura 32: Poema



Figura 31: Fantasía



Licencias:

Licencia de imágenes: Opté por licencias de reconocimiento, permitiendo su uso comercial y su compartición.

Figura 33: CC imágenes



Licencia de la web: Para la web opté por una licencia para permitir que se comparta, pero no permitir un uso comercial.

Figura 34: CC web

¿Tiene una página web?



Esta obra está bajo una licencia de
Creative Commons Reconocimiento-
NoComercial 4.0 Internacional.

¡Copie este código para informar a sus visitantes!

```
<a rel="license"
href="http://creativecommons.org/licenses
/by-nc/4.0/"><img alt="Licencia de Creative
Commons" style="border-width:0"
src="https://i.creativecommons.org/l/by-
```

☒ Icono
normal

☐ Icono
compacto

Usabilidad y accesibilidad.

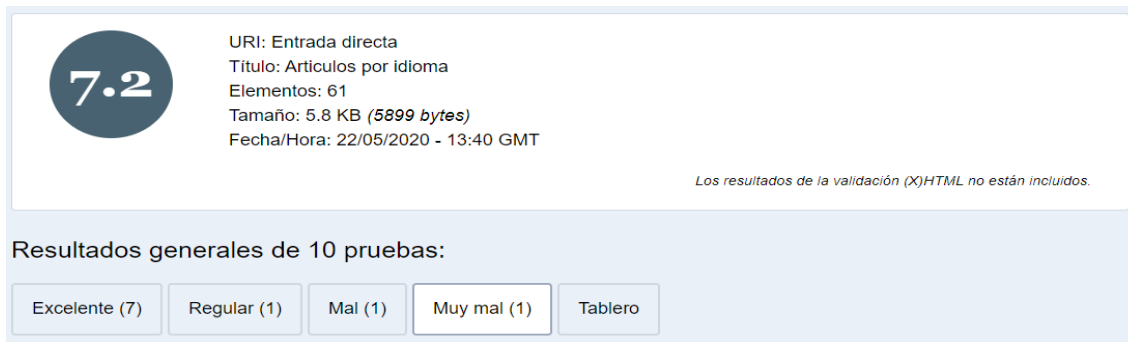
Para estudiar la usabilidad y accesibilidad de mi aplicación lo más detalladamente posible, pasé las plantillas a html y fui analizandolas una a una en



<http://examinator.ws/>.

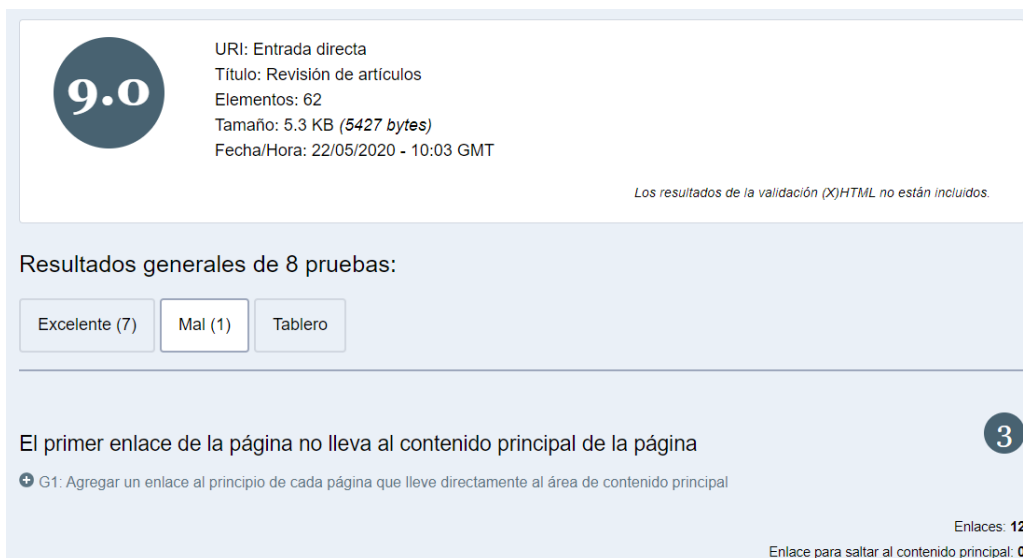
Los resultados que me dieron fue que los encabezados estaban mal, que faltaban enlaces que llevaran al contenido principal, o que le faltaba el atributo title a algunas referencias.

Figura 35: Nota test HTML (antes de las correcciones)



Finalmente conseguí resolver estos errores hasta quedarme solo con uno.

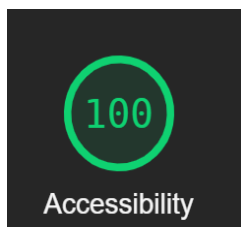
Figura 36: Test tras correcciones



Pasé otro test, con una extensión de chrome.



Figura 37: Test de accesibilidad con Chrome



Al ser en ambos casos test gratuitos no realizan un análisis en profundidad, por lo que tuve que poner de mi parte para revisar que podría mejorarse.

Proceso de despliegue.

Antes de intentar desplegar la aplicación, paso la base de datos a un servidor de base de datos, en este caso GearHost.

Figura 38: Base de datos en el servidor GearHost

The screenshot shows the GearHost database management interface. At the top, there's a header with the database name 'literaturadaw' and a button 'Add Database User'. Below this is a green banner for 'Free Database' with a database icon and a message: 'You are using free database which is limited. It's recommended to upgrade to Standard.' with an 'Upgrade Now' button. The main content area is divided into three sections: 'database server' (den1.mysql3.gear.host), 'plan' (Free (upgrade)), and 'data utilization' (0.00B). Below these is a 'database users' section with a table showing the user 'literaturadaw' with a masked password and 'Read & Write' permissions. There are also icons for eye and gear settings.

Username	Password	Permission
literaturadaw	*****	Read & Write

Una vez creada en el servidor, accedo a ella a través de heidi e inserto el contenido.



Figura 39: Inicio HeidiSQL

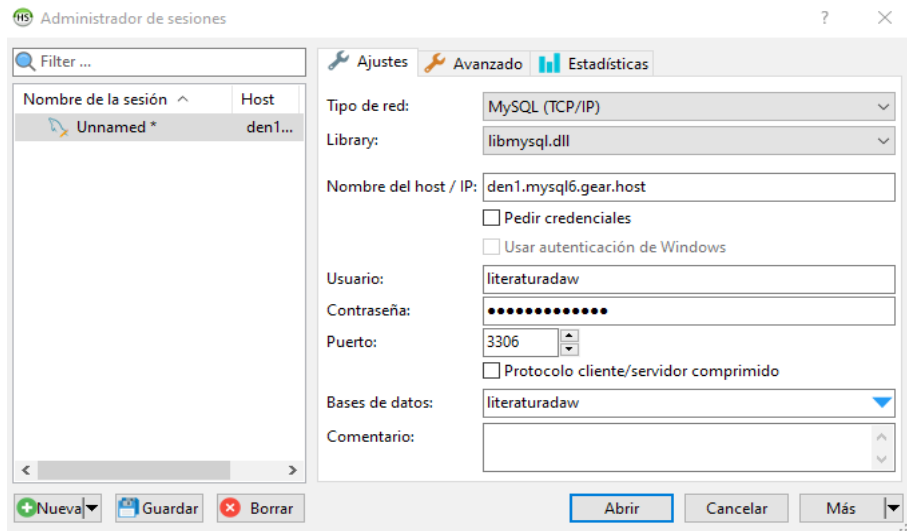
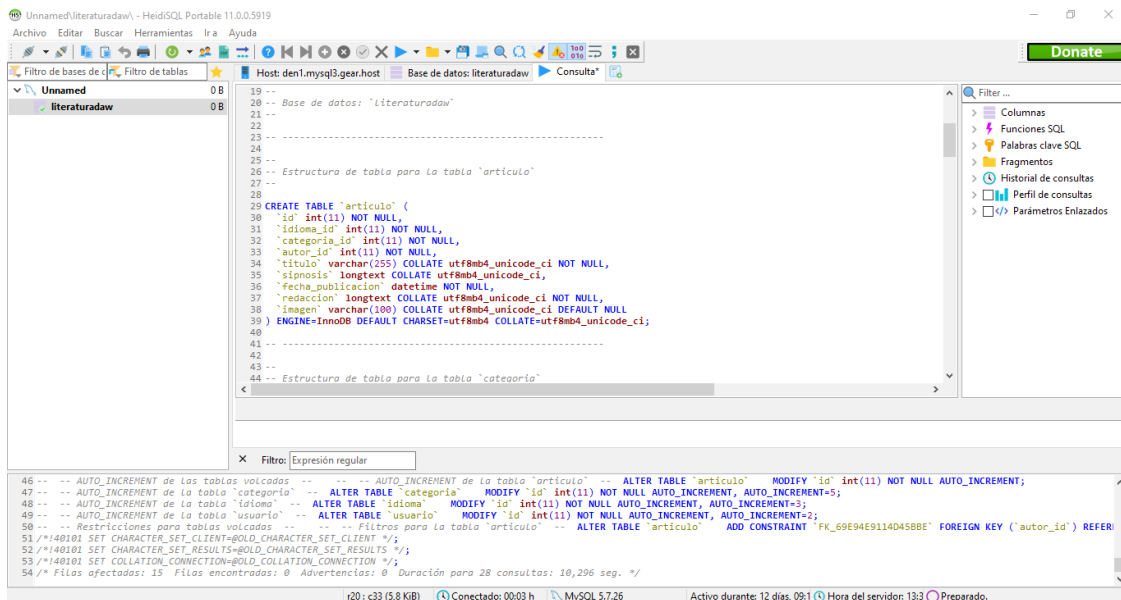
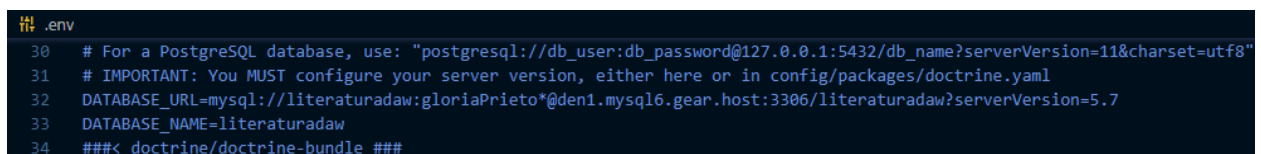


Figura 40: Inserción de base de datos mediante SQL



Tras esto debo indicar en el proyecto la nueva ruta de base de datos. Esto se hace en el archivo '.env'.

Figura 41: Llamada a BBDD



Después compruebo que tengo heroku y creo un proyecto en el servidor.



Figura 42: Instalación heroku

```
usuario@ubuntu:~/Symfony/proyecto_literatura$ sudo snap install --classic heroku
heroku v7.41.1 from Heroku✓ installed
usuario@ubuntu:~/Symfony/proyecto_literatura$ heroku login --interactive
heroku: Enter your login credentials
Email: gloriapr@r@gmail.com
Password: *****
```

Figura 43: Creación proyecto heroku

```
usuario@ubuntu:~/Symfony/proyecto_literatura$ heroku create --region eu gestor-articulos
Creating gestor-articulos... done, region is eu
https://gestor-articulos.herokuapp.com/ | https://git.heroku.com/gestor-articulos.git
```

Para el despliegue de la aplicación opté por usar el repositorio que tengo en github, para ello, entro en mi cuenta de heroku y, dentro de deploy, elijo dicha opción.

Figura 44: Configuración de proyecto heroku 1

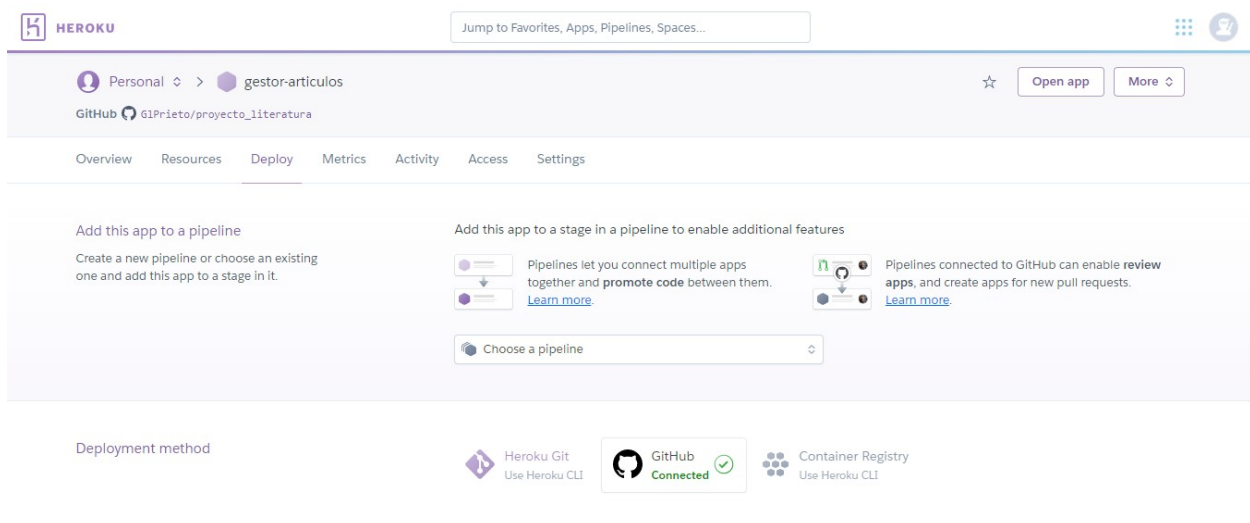
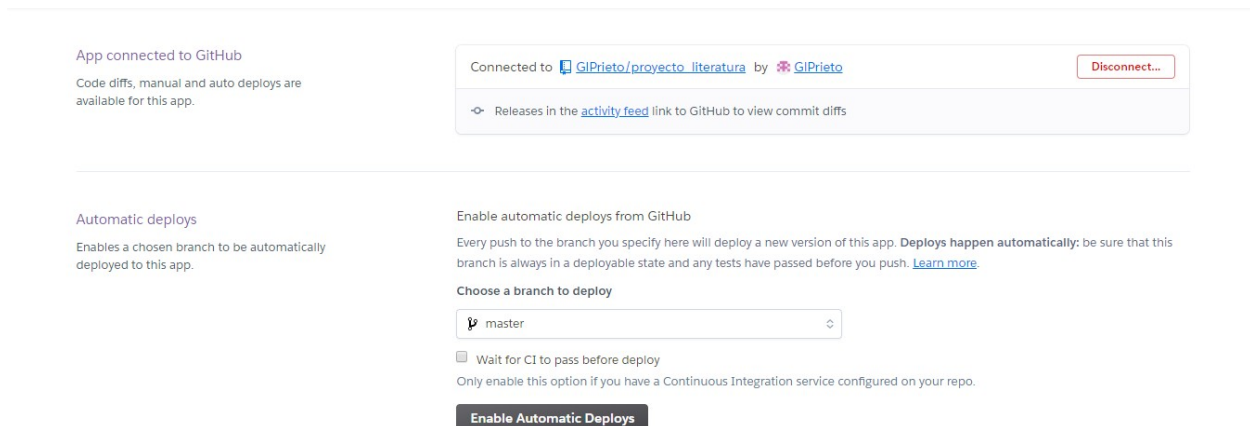


Figura 45: Configuración de proyecto heroku 2



Para conectar heroku con mi proyecto de github, uso TravisCI.



Para que TravisCI haga test de mi proyecto tuve que crear un archivo `travis.yalm`, cuyo contenido es:

```
# Project language

language: php


# Start mysql service

services:

  - mysql


# Cache composer packages so "composer install" is faster

cache:

  directories:

    - $HOME/.composer/cache/files


# Matrix to test in every php version

matrix:

  # Fast finish allows to set the build as "finished" even if the
  "allow_failures" matrix elements are not finished yet.

  fast_finish: true

  include:

    - php: 7.4


# Define an environment variable

env:

  - SYMFONY_VERSION="5.0.*" DB=mysql


# Update composer

before-install:

  - composer self-update


# Install composer dependencies,
```



```
# Create database, schema and fixtures

install:

    - composer install

# Run script

script:

    - phpunit

# After a build, send email notification with the build results

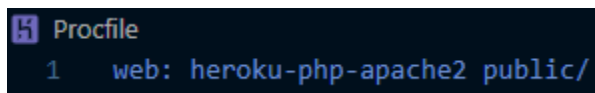
notifications:

    email: gloriaprora@gmail.com
```

Con este, cada vez que realizo un push al repositorio en GitHub, TravisCI procede a realizar la construcción (build).

Antes de desplegar la aplicación mediante Heroku creé un archivo Procfile, el cual le dirá a heroku que usar para cargar el servidor web de forma correcta. Es necesario porque heroku tiene entendido que el directorio principal de un proyecto Symfony es otro distinto al que es realmente, por lo que hay que indicar la ruta correcta en este archivo.

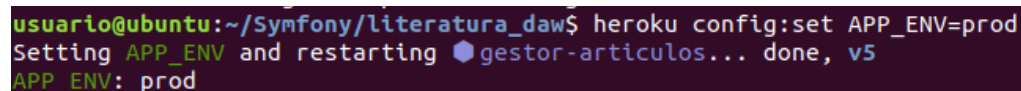
Figura 46: Contenido Procfile



```
1 web: heroku-php-apache2 public/
```

A parte de añadir este archivo, es necesario indicar que variables de entorno debe usar:

Figura 47: Indicación de variables de entorno



```
usuario@ubuntu:~/Symfony/literatura_daw$ heroku config:set APP_ENV=prod
Setting APP_ENV and restarting ● gestor-articulos... done, v5
APP_ENV: prod
```

Como tengo implantada en el proyecto la opción de inicio de sesión, debo indicar otra ruta para el registro. Para esto solo hay que cambiar la ruta indicada en el archivo `config/packages/prod/monolog.yaml`.



Figura 48: Edición de monolog.yalm

```
config > packages > prod > {..} monolog.yaml > {} monolog > {} handlers > {} nested
1  monolog:
2      handlers:
3          main:
4              type: fingers_crossed
5              action_level: error
6              handler: nested
7              excluded_http_codes: [404, 405]
8              buffer_size: 50 # How many messages should be saved? Prevent memory leaks
9          nested:
10             type: stream
11             path: 'php://stderr'
12             level: debug
13         console:
14             type: console
15             process_psr_3_messages: false
16             channels: ["!event", "!doctrine"]
```

Para obtener las urls de las páginas creadas es necesario instalar un paquete de apache que genera un archivo `.htaccess` en la carpeta public.

Figura 49: Instalación de paquete de apache

```
usuario@ubuntu:~/Symfony/proyecto_literatura$ composer require apache-pack
Using version ^1.0 for symfony/apache-pack
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "5.0.*"
Nothing to install or update
Generating autoload files
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
Executing script cache:clear [OK]
Executing script assets:install public [OK]
```

Por último es necesario indicar a Symfony que confíe en el enrutador de heroku.

Heroku enruta las solicitudes a través de una capa de servidores de proxy inversos.

Para que se genere dicha confianza es necesario editar el archivo `index.php`.

Figura 50: Configuración index.php

```
index.php x
public > index.php
14
15 if ( $trustedProxies = $_SERVER['TRUSTED_PROXIES'] ?? $_ENV['TRUSTED_PROXIES'] ?? false ) {
16     Request::setTrustedProxies( explode( ',', $trustedProxies ), Request::HEADER_X_FORWARDED_ALL ^ Request::HEADER_X_FORWARDED_HOST );
17 }
18
19 if ( $trustedHosts = $_SERVER['TRUSTED_HOSTS'] ?? $_ENV['TRUSTED_HOSTS'] ?? false ) {
20     Request::setTrustedHosts( [$trustedHosts] );
21 }
```

Tras esto ya se puede realizar el despliegue:



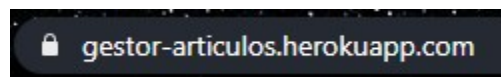
Figura 51: Despliegue en heroku

```
usuario@ubuntu:~/Symfony/literatura_daw$ git push heroku master
Contando objetos: 714, listo.
Comprimiendo objetos: 100% (358/358), listo.
Escribiendo objetos: 100% (714/714), 7.90 MiB | 3.06 MiB/s, listo.
Total 714 (delta 351), reused 624 (delta 305)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> PHP app detected
remote: -----> Bootstrapping...
remote: -----> Installing platform packages...
remote:          - php (7.4.5)
remote:          - apache (2.4.43)
remote:          - nginx (1.18.0)
remote: -----> Installing dependencies...
remote:          Composer version 1.10.5 2020-04-10 11:44:22
remote:          Loading composer repositories with package information
remote:          Installing dependencies from lock file
remote:          Package operations: 97 installs, 0 updates, 0 removals
remote:            - Installing ocrapius/package-versions (1.8.0): Downloading (100%)
remote:            - Installing symfony/flex (v1.6.2): Downloading (100%)
remote:
remote:          Prefetching 95 packages
```

Figura 52: Resultado despliegue

```
remote:          Released v6
remote:          https://gestor-articulos.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/gestor-articulos.git
 * [new branch]      master -> master
usuario@ubuntu:~/Symfony/literatura_daw$ heroku open
```

Figura 53: URL página





8. Manual básico de utilización.

<https://gestor-articulos.herokuapp.com/>

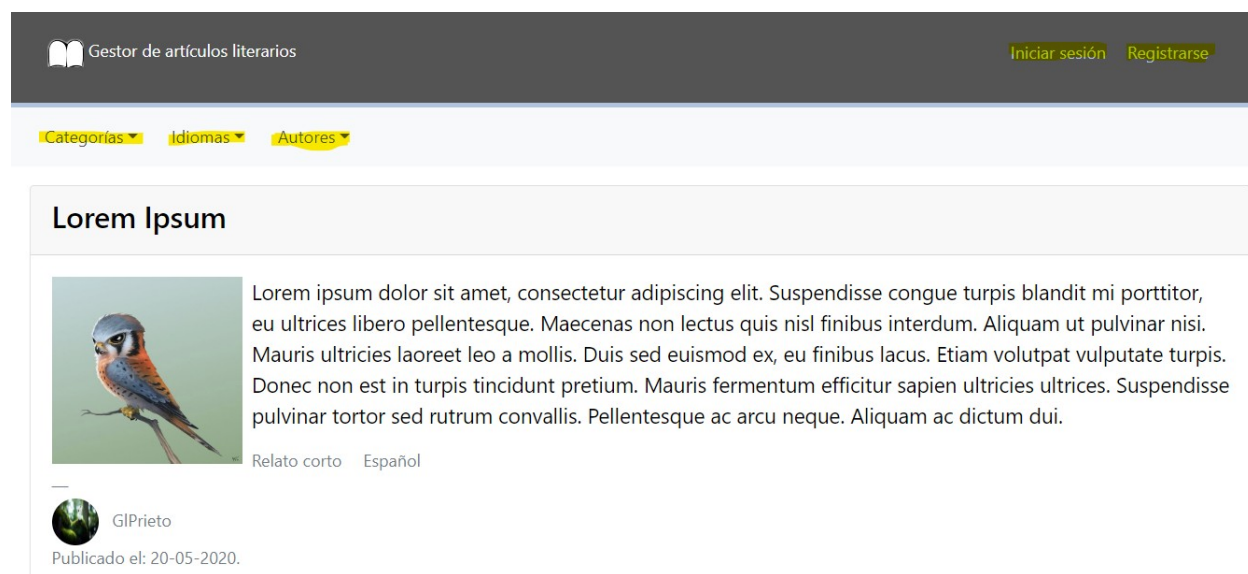
Usuario anónimo

Para leer artículos en la página web no es necesario tener cuenta. Esta clase de usuarios se le denominan usuario anónimo.

Las operaciones que puede realizar un usuario anónimo las puede realizar cualquier otro tipo de usuario.

La primera página que se ve al entrar a la web es la siguiente:

Figura 54: Inicio

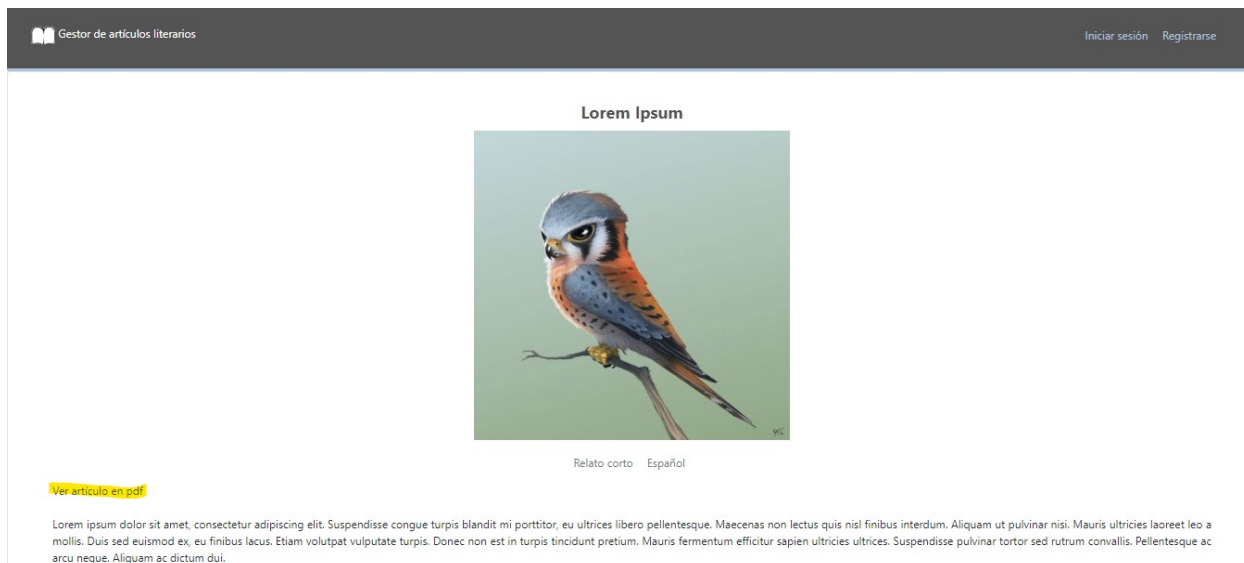


Se aprecian, además de una lista de artículos, una serie de enlaces, de los cuales, aquellos que están justo encima de la lista de artículos, son para filtrarlos y mostrar solo los artículos de una categoría concreta, de un idioma en particular o de un autor concreto.

Para ver un artículo en concreto de la lista se puede clicar en el título, la imagen o en la descripción que lo acompaña.

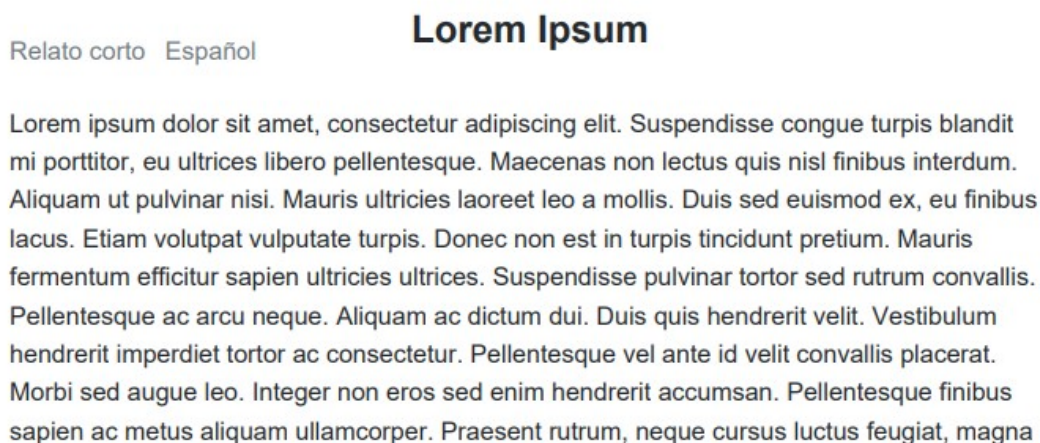


Figura 55: Vista de un artículo



Dentro de la vista del artículo, podemos proceder a su lectura en esa misma pantalla o bien generar un pdf clicando en el enlace que hay justo encima del texto.

Figura 56: pdf de un artículo



Para salir del pdf no es necesario ir a la pantalla anterior con las flechas de navegación, pues se puede ir al inicio de la página clicando en el enlace que aparece en la parte superior del documento.

Las dos últimas acciones que podría realizar un usuario anónimo son la de registrarse y la de iniciar sesión:



Figura 57: Navegador inicio de sesión y registro



Aquel usuario que no tenga cuenta y quiera crear artículos en la web deberá rellenar el formulario de registro, en el cual solo son obligatorios aquellos campos marcados con *.

Figura 58: Formulario de registro

Una vez registrado debería iniciarse sesión automáticamente, pero si no es el caso, ya que se trata de un usuario con cuenta, puede acceder al formulario de login e iniciar sesión.

Figura 59: Formulario de login



Usuario “autor” (tiene la posibilidad de crear artículos)

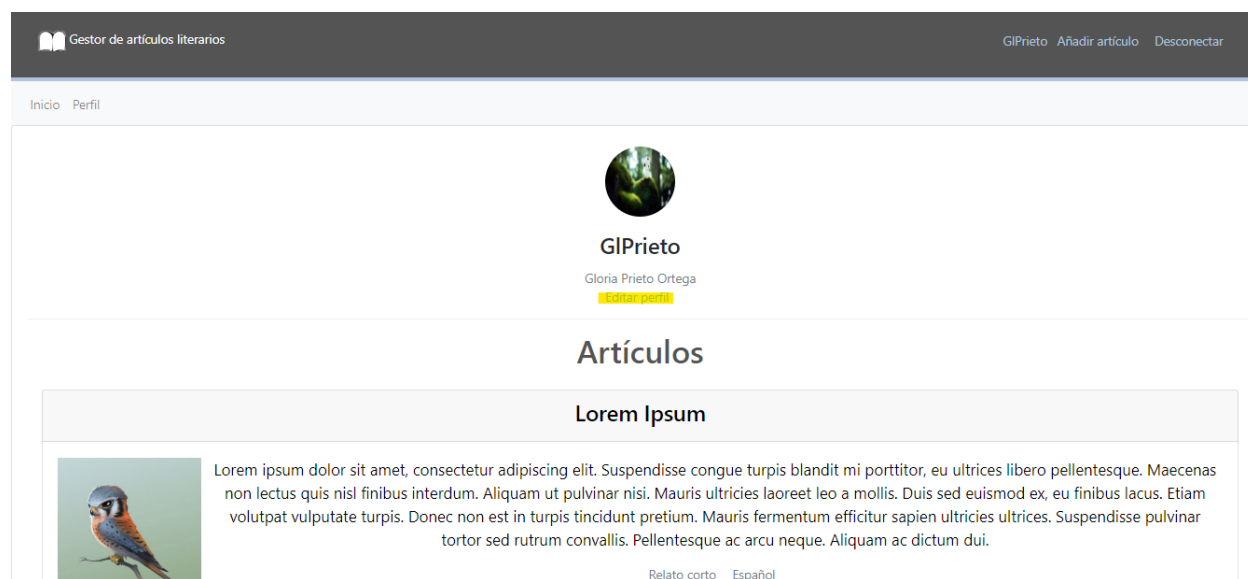
Una vez iniciada la sesión, cambia la barra de navegación principal, mostrando esta vez tres enlaces nuevos, de los cuales, el último, es para cerrar la sesión.

Figura 60: Navegación: perfil, nuevo artículo, desconexión



De los otros dos, el primero muestra la firma del usuario o nombre de perfil. Este enlace lleva a la página del perfil del usuario autenticado, en la cual puede ver los artículos que haya realizado y ver los datos que haya introducido a la hora del registro.

Figura 61: Vista perfil



En esta página se le da opción al usuario de editar su perfil. Este enlace llevará a un formulario similar al de registro, que permitirá añadir información, o solo cambiarla.

El siguiente enlace lleva a un formulario que permite crear un artículo, introduciendo datos obligatorios como son el título, la descripción y la redacción de este.



Figura 62: Formulario de creación

Gestor de artículos literarios

GPrieto Añadir artículo Desconectar

Rellene los campos mostrados a continuación

Título:

Descripción:

Redacción:

Administrador

Un usuario administrador puede realizar todo lo anterior, pero en vez de tener un enlace que lleve a su perfil, dicho enlace le lleva a la página donde puede administrar los elementos que fueron creados en la página (artículos, usuarios) y, crear nuevas categorías e introducir nuevos idiomas.

Figura 63: Vista página administrador

Gestor de artículos literarios

Admin Desconectar

Subidas recientes

Lorem Ipsum Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse congue turpis blandit mi porttitor, eu ultrices libero pellentesque. Maecenas non lectus quis nisl finibus interdum. Aliquam ut pulvinar nisi. Mauris ultricies laoreet leo a mollis. Duis sed euismod ex, eu finibus lacus. Etiam volutpat vulputate turpis. Donec non est in turpis tincidunt pretium. Mauris fermentum efficitur sapien ultricies ultrices. Suspendisse pulvinar tortor sed rutrum convallis. Pellentesque ac arcu neque. Aliquam ac dictum dui.

Todas las subidas

Nuevos usuarios

Prueba
prueba@gmail.com

Todos los usuarios

Categorías

Fantasia

Para ver todos los artículos o, de la misma manera, todos los usuarios (menos la figura del administrador) debe clicar en los enlaces que estan en esos respectivos grupos.

Una vez hecho esto, se accede a una página similar a la siguiente donde, en el caso de los artículos, hay dos enlaces. El título del artículo lleva a la página que muestra el



artículo, mientras que el enlace de la última columna lo que hace es borrar el elemento.

Figura 64: Administrador vista artículos

Gestor de artículos literariosAdminDesconectar

InicioArtículos

Artículos

Título	Fecha de publicación	Categoría	Idioma	
Lorem Ipsum	20-05-2020	Relato corto	Español	Eliminar
Lorem Ipsum	19-05-2020	Poema	Español	Eliminar
Desarrollo web	19-05-2020	Poema	Español	Eliminar
Uno escritos	19-05-2020	Ficción	Español	Eliminar
Lorem Ipsum	19-05-2020	Terror	Español	Eliminar

Copyright © 2020 | Gestor de artículos literariosRealizado por: Gloria Prieto Ortega | Arriba

Este último enlace está también para el caso de los usuarios.

En el caso de los otros dos grupos, los enlaces llevan a un formulario para crear una nueva categoría o añadir un nuevo idioma, respectivamente.

Figura 65: Administrador vista categorías e idiomas

Categorías	
Fantasia	
Ficción	
Poema	
Relato corto	
Terror	
Añadir categoría	
Idiomas	
English	
Español	
Añadir idioma	



9. Dificultades encontradas

Al inicio planteé el modelo relacional de una forma compleja, lo cual me llevó a dedicar más tiempo del previsto originalmente, hasta llegar al modelo final deseado.

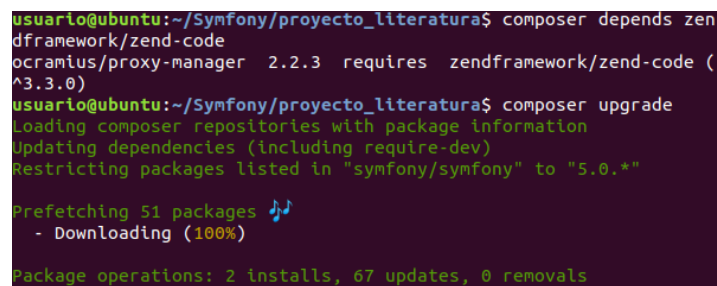
Symfony tiene su propia documentación, en donde explica de forma detallada, y con ejemplos, varias funcionalidades (entidad usuario, login, registro,...), pero cuadrar esos ejemplos con un proyecto concreto puede llegar a ser un proceso largo, pues en ocasiones no se entiende dónde puede estar el error cometido.

Avisos al instalar via terminal con composer:

```
'Package zendframework/zend-code is abandoned...'
```

Para resolver esto tuve que actualizar a una versión más reciente.

Figura 66: Actualización composer



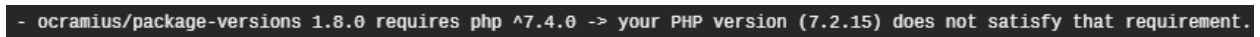
```
usuario@ubuntu:~/Symfony/proyecto_literatura$ composer depends zen
dframework/zend-code
ocramius/proxy-manager 2.2.3 requires zendframework/zend-code (
^3.3.0)
usuario@ubuntu:~/Symfony/proyecto_literatura$ composer upgrade
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "5.0.*"

Prefetching 51 packages 🎵
- Downloading (100%)

Package operations: 2 installs, 67 updates, 0 removals
```

A raíz de actualizar composer, la versión de php que había instalado no servía para este proyecto, pues, a la hora de intentar pasar los test de travis me salió el siguiente error:

Figura 67: Error de fallo de versión



```
- ocramius/package-versions 1.8.0 requires php ^7.4.0 -> your PHP version (7.2.15) does not satisfy that requirement.
```

Esta versión de composer requiere de la versión 7.4 de php.

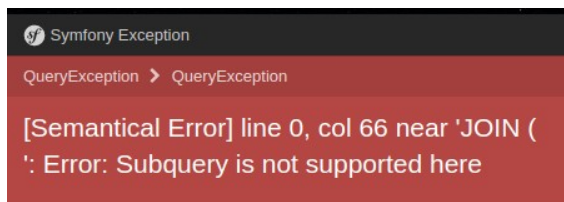
Para corregir este problema tuve que cambiar la versión de php tanto en el archivo de `.travis.yml`.

Problemas con sentencias mysql en Symfony:

A la hora de hacer consultas con Doctrine en Symfony, no es aceptada cualquiera. A la hora de hacer una consulta a mi base de datos mediante HeidiSQL, me mostraba como resultado una tabla cuyo contenido se aproximaba a lo que pedí. A la hora de pasar dicha consulta al proyecto, me saltó el siguiente error:



Figura 68: Error en la consulta



Con la guía de Alfonso opté por reescribir la consulta, de tal forma que, aun realizando la petición usando otros datos, me diera el resultado esperado.

Despliegue:

Al desplegar tuve varios problemas aparte de tener que actualizar la versión de php.

No me aparecía el archivo **.htaccess**, por lo que tuve que reinstalar el paquete.

Figura 69: Eliminación de paquete

```
usuario@ubuntu:~/Symfony/literatura_daw$ composer remove apache-pack
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "5.0.*"
Package operations: 0 installs, 0 updates, 1 removal
- Removing symfony/apache-pack (v1.0.1)
Writing lock file
Generating autoload files
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
Executing script cache:clear [OK]
Executing script assets:install public [OK]
```

Figura 70: Reinstalación paquete

```
usuario@ubuntu:~/Symfony/literatura_daw$ composer require symfony/apache-pack
Using version ^1.0 for symfony/apache-pack
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Restricting packages listed in "symfony/symfony" to "5.0.*"
Package operations: 1 install, 0 updates, 0 removals
- Installing symfony/apache-pack (v1.0.1): Loading from cache
Writing lock file
Generating autoload files
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
Symfony operations: 1 recipe (ef91cb48c1444b6d98871c6fe88b33f4)
- [WARNING] symfony/apache-pack (>=1.0): From github.com/symfony/recipes-contrib:master
  The recipe for this package comes from the "contrib" repository, which is open to community contributions.
  Review the recipe at https://github.com/symfony/recipes-contrib/tree/master/symfony/apache-pack/1.0

  Do you want to execute this recipe?
  [y] Yes
  [n] No
  [a] Yes for all packages, only for the current installation session
  [p] Yes permanently, never ask again for this project
  (defaults to n): y
- Configuring symfony/apache-pack (>=1.0): From github.com/symfony/recipes-contrib:master
Executing script cache:clear [OK]
Executing script assets:install public [OK]

Some files may have been created or updated to configure your new packages.
Please review, edit and commit them: these files are yours.
```



10. Posibles mejoras

El proyecto no es perfecto, la idea sería que cada cierto tiempo se le añadiesen nuevas funcionalidades, para así adaptarse a las necesidades de los usuarios. Aparte, la tecnología va evolucionando a diario, por lo que podrían surgir novedades atrayentes, ya sean para mejorar lo que ya hay o para implementar alguna nueva función.

Entre las posibles mejoras que se podrían llevar a cabo y nuevas funciones a implementar, destaco las siguientes:

General:

- **Añadir más elementos (nuevas consultas)** y mejorar las existentes. Por ejemplo en el caso de mostrar el último artículo de cada categoría, implementar que se haga con el elemento de bootstrap, carousel.
- **Traducción de la página:** Implementar la posibilidad de traducir la página en varios idiomas.
- **Traducción de artículos:** Habilitar una función que traduzca automáticamente cualquier artículo a otro idioma (en principio solo estarían los idiomas inglés y español).
- Para un correcto funcionamiento de la web, y que no se publicase cualquier cosa, los **artículos** deberían ser **revisados** por los administradores antes de su publicación, por lo cual habría que añadir un paso intermedio entre la creación y la visualización del artículo en la web.
- **Visualización de perfiles de otros usuarios:** Esto abriría la posibilidad de añadir nuevas funcionalidades, como la de seguir a aquellos autores que más les atraigan.
- **Guardar favoritos:** Implementar una funcionalidad que permita a los usuarios guardar aquellos artículos que les gustan.

Creación y edición de artículos y usuarios:

- **Formateo de texto:** Añadir la posibilidad de que el propio usuario formatee el texto de su artículo a su gusto, tal que sea más atrayente para el lector.
- **Más opciones de filtro para los artículos:** Permitir seleccionar más de una categoría o idioma, ordenar por fecha de publicación, por artículos más populares (según las veces que fue guardado por otros usuarios).
- **Edición de imágenes:** Implementar una funcionalidad que permita recortar las imágenes antes de su subida, para que, entre otras cosas, el usuario pueda ver



previamente como quedaría en la página.

- Implementación de **cambio de contraseña** y envío de **email de confirmación** para activar la cuenta.

Página de administrador:

- **Lista de artículos en PDF:** Añadir la opción de crear un archivo pdf donde se encuentre la lista de las obras publicadas por un usuario.

Seguridad:

Añadir **aviso legal, política de privacidad y normas de uso propias.**

Protección de obras:

- Otorgar **licencias** a los artículos de los usuarios.
- Nuevas funciones:
 - Impedir que se pueda marcar el texto.
 - Hacer que el documento pdf generado no se pueda descargar a no ser que el usuario propietario de su permiso, o que solo se genere si el usuario desea esa opción.

Posibilidad de venta:

Teniendo en cuenta la protección de los artículos, se podría añadir la opción de que los usuarios autores puedan vender sus artículos.



11. Fuentes de Información y recursos utilizados. Webgrafía:

Medium (blog):

<https://medium.com/@ger86/symfony-como-crear-un-sistema-de-usuarios-desde-cero-dbb616b536f>

Bootstrap 4 (09 de mayo)

<https://symfony.com/doc/current/form/bootstrap4.html>

getbootstrap.com: <https://getbootstrap.com/docs/4.4/getting-started/download/>

Ejemplos (Examples)

blog: <https://getbootstrap.com/docs/4.4/examples/blog/>

album: <https://getbootstrap.com/docs/4.4/examples/album/#>

sign up: <https://getbootstrap.com/docs/4.4/examples/sign-in/>

pricing: <https://getbootstrap.com/docs/4.4/examples/pricing/>

Mdbootstrap (ejemplos más detallados): <https://mdbootstrap.com/> (10 de mayo)

Corrección de advertencias (advertencia zendframe)

[:https://stackoverflow.com/questions/59923243/composer-warning-package-zendframework-zend-code-is-abandoned](https://stackoverflow.com/questions/59923243/composer-warning-package-zendframework-zend-code-is-abandoned) (09 de mayo)

Frontend (13 de mayo)

<https://diego.com.es/operadores-y-expresiones-en-twig>

Establecimiento de colores de la web

<https://coolers.co/000000-4e4e4e-acc1dd-eeeeee-ffffff>

Distintas versiones favicon

<https://www.favicon.cc/>

Despliegue (16 de mayo)

<https://devcenter.heroku.com/articles/deploying-symfony4>

<https://github.com/jamj2000/tiendaw>



HeidiSQL: <https://www.heidisql.com/download.php>

17 de mayo

Investigación para corrección de errores:

https://symfony.com/doc/current/controller/upload_file.html

https://symfony.com/doc/2.6/cookbook/doctrine/file_uploads.html

Generar pdf en Symfony (19 de mayo)

<https://ourcodeworld.com/articles/read/799/how-to-create-a-pdf-from-html-in-symfony-4-using-dompdf>

20 de mayo

Repetir contraseña (En symfony tienen un campo de tipo 'repeated')

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

Tipos en Symfony:

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

Tests de accesibilidad y consulta de estilos css4 (21 de mayo)

Test de accesibilidad:

https://googlechrome.github.io/lighthouse/viewer/?psiurl=https%3A%2F%2Fgestor-articulos.herokuapp.com%2F&strategy=desktop&category=performance&category=accessibility&category=best-practices&category=seo&category=pwa&utm_source=lh-chrome-ext#performance

Estilos: https://www.w3schools.com/css/css_list.asp

Estándares de codificación de PHP (24 de mayo):

<https://www.php-fig.org/psr/psr-2/>

<https://www.php-fig.org/psr/psr-4/>

Licencias creative commons (27 de mayo): <https://creativecommons.org/choose/>

Fracmentos de artículos (30 de mayo): DIEZ NEGRITOS - Agatha Christie, La Historia Interminable - Michael Ende, LIBRO DE POEMAS - Federico García Lorca.



Anexos

Clases y funciones

Todo el código se encuentra en el repositorio de GitHub https://github.com/GIPrieto/literatura_daw.git.

Subida de archivos

Para la subida de las imágenes creé el servicio `SubidaArchivos`.

En el método `upload` se le añade al nombre del archivo dado por el cliente una extensión y luego se sube al directorio de destino, el cuál es dado en el método `get` de `directorioDestino`.

Para guardar los archivos subidos tuve que indicar que se creara un directorio que los contuviera.

Para ello, añadí en el 'archivo `config/services.yaml`', en el apartado de argumentos, dentro de servicios, el nombre del directorio de destino y, en el apartado de parámetros, especifiqué la ruta de dicho directorio.

```
parameters:
    directorioImágenes: '%kernel.project_dir%/public/subidas/imagenes'
services:
...
# add more service definitions when explicit configuration is needed
# please note that last definitions always *replace* previous ones
App\Service\SubidaArchivos:
    arguments:
        $directorioDestino: '%directorioImágenes%'
```

Usuarios

Comando 'crear usuario'

Para crear usuarios se puede, o bien crear un controlador de usuario para crear usuarios o bien hacer uso de comandos. Opté por este último con idea de que un administrador no tenga que acceder a la página para generar un usuario, por otro lado me parece más cómodo crear usuarios con rol administrador introduciendo en el



terminal los datos requeridos.

Creé una carpeta Commands donde guardé el archivo `ComandoCrearUsuario`.

El comando a introducir en el terminal para crear un usuario será el siguiente:

`php bin/console app:create-user email contraseña firma [--admin]` (si está, indicará que es admin).

En el método `configure` indiqué que datos eran obligatorios proporcionar y cuál era opcional:

```
protected function configure()
{
    $this
        ->setDescription('Este comando permite crear un usuario')
        ->setHelp('Este comando permite crear un usuario')
        ->addArgument('email',
            InputArgument::REQUIRED,
            'email'
        )
        ->addArgument('password',
            InputArgument::REQUIRED,
            'La contraseña del nuevo usuario'
        )
        ->addArgument('firmaUsuario',
            InputArgument::REQUIRED,
            'El nombre con el que le verán los demás usuarios'
        )
        ->addOption('admin', null, InputOption::VALUE_NONE, 'Crea usuario
como admin')
    ;
}
```

Y para ejecutar el comando, creé el método `execute`.

Formulario de registro

Para permitir que cualquier tipo de usuario se registre, creé con MakerBundle el



formulario de registro `RegistrationFormType`.

Figura 71: Formulario registro

```
usuario@ubuntu:~/Symfony/proyecto_literatura$ bin/console make:registration-form

Creating a registration form for App\Entity\Usuario

Do you want to add a @UniqueEntity validation annotation on your Usuario class to make sure duplicate accounts aren't created? (yes/no) [yes]:
> yes

Do you want to automatically authenticate the user after registration? (yes/no) [yes]:
> yes

updated: src/Entity/Usuario.php
created: src/Form/RegistrationFormType.php
created: src/Controller/RegistrationController.php
created: templates/registration/register.html.twig

Success!

Next: Go to /register to check out your new form!
Make any changes you need to the form, controller & template.
```

Formulario de login

De esta misma forma, para que un usuario registrado pueda acceder a la página generé el formulario de login, introduciendo en el terminal: `bin/console make:auth`.

Figura 72: Formulario login

```
usuario@ubuntu:~/Symfony/proyecto_literatura$ bin/console make:auth

What style of authentication do you want? [Empty authenticator]:
[0] Empty authenticator
[1] Login form authenticator
> 1

The class name of the authenticator to create (e.g. AppCustomAuthenticator):
> FormularioLogin

Choose a name for the controller class (e.g. SecurityController) [SecurityController]:
> SecurityController

Do you want to generate a '/logout' URL? (yes/no) [yes]:
> yes

created: src/Security/FormularioLoginAuthenticator.php
updated: config/packages/security.yaml
created: src/Controller/SecurityControllerController.php
created: templates/security/login.html.twig

Success!
```



Con esto se crearon:

- Ruta de /login y /logout.
- Controlador SecurityController.php. (Se unirá al controlador de registro)
- Plantilla para el formulario de iniciar sesión en login.html.twig.
- Un GuardAuthenticator encargado de procesar el formulario de login dentro de la clase LoginFormAuthenticator.php en la nueva carpeta Security.

Y se actualizó el archivo security.yaml para establecer tanto el formulario de login como la ruta de logout dentro del firewall main.

```
security:

    encoders:

        App\Entity\Usuario:
            algorithm: auto

    providers:

        app_user_provider:
            entity:

                class: App\Entity\Usuario
                property: email

    firewalls:

        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        main:
            anonymous: true

            guard:
                authenticators:
                    - App\Security\FormularioLoginAuthenticator
```

Configuración del logout:

```
logout:

    path: app_logout

    # where to redirect after logout

    target: index
```



En este mismo archivo, `config/packages/security.yaml`, para permitir que los usuarios anónimos puedan siempre acceder a la ruta `/login`, añadí:

```
access_control:

    ...

    - { path: ^/login$, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

Las rutas a estos formularios se guardan en una carpeta distinta a las otras (`config/routes/security.yaml`).

```
app_login:
    path: /login
    controller: App\Controller\UsuarioController::login

app_logout:
    path: /logout
    controller: App\Controller\UsuarioController::logout
```

Configuración del formulario de login:

Lo único que tuve que modificar en el formulario de login fué el método `'onAuthenticationSuccess'` de cara a redirigir al usuario hacia donde queramos cuando inicie sesión correctamente.

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $providerKey))
    {
        return new RedirectResponse($targetPath);
    }

    return new RedirectResponse($this->urlGenerator->generate('home'));
}
```




Finalmente opté por fusionar los dos controladores generados creando así un único controlador para los métodos que realiza la entidad usuario.

Permisos

Como existen dos tipos de usuarios, o tres, si se cuenta al anónimo, necesité otorgar permisos, para que no puedan acceder todos a las mismas páginas.

Por defecto, los usuarios registrados mediante el formulario de registro tienen rol `ROLE_USER`.

Para indicar permisos en las plantillas hay que añadir `{% if is_granted('...') %}`

Entre paréntesis se indica el rol deseado. Por ejemplo, `is_granted('IS_AUTHENTICATED_FULLY')` solo mostrará el contenido a los que hayan iniciado sesión.

Artículos

La opción de editar y borrar artículos únicamente debería tenerla el propietario del archivo, desde la vista principal de la web (para los administradores habrá una página en concreto en la que puedan eliminar artículos).

Para otorgar permisos individuales, en este caso, indicar si tiene permisos el usuario logueado, será necesario crear 'voters' que es como se le denominan a los permisos que crearé.

En el controlador, en el método en el cuál deben comprobarse los permisos, se incluirá `$this->denyAccessUnlessGranted('view', $post);`.

En `src/Security/ArticuloVoter.php` se creará la clase que establece todos los permisos que queramos (hecho a mano, siguiendo una plantilla), llamando a las clases implicadas y a `use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;`

`use Symfony\Component\Security\Core\Authorization\Voter\VoterInterface;`

`GestorController.php`. Controlador para la gestión de artículos.

Como página principal decidí que se mostrase una lista de artículos, mencionandose la categoría e idioma de cada uno de ellos y su autor.

Para generar pdf de un artículo creé el método `verArticuloPDF` (previamente instalé `Dompdf`). Las líneas para generar el pdf son las siguientes:



// Configuración de Dompdf

```
$pdfOptions = new Options();  
  
$pdfOptions->set('defaultFont', 'Arial');
```

// Instancia de Dompdf con las opciones

```
$dompdf = new Dompdf($pdfOptions);  
  
// Recibir la respuesta HTML en la plantilla dada  
$html = $this->renderView('articulo/verArticuloPDF.html.twig', [  
    'titulo' => $titulo,  
    'articulo' => $articulo,  
]);
```

// Cargar el html en Dompdf

```
$dompdf->loadHtml($html);
```

// Indicación de las especificaciones de tamaño

```
$dompdf->setPaper('A4', 'portrait');
```

// Renderización del html como pdf

```
$dompdf->render();
```

// Mostrar el pdf en el navegador

```
$dompdf->stream("mypdf.pdf", [  
    "Attachment" => false  
]);  
  
}
```

En los métodos `editarArtículo` y `eliminarArtículo` se le llama a la clase 'voter' que comenté anteriormente, indicando que solo tendrá acceso a esta funcionalidad el usuario dueño del artículo. Las líneas que lo indica son las siguientes:

```
$this->denyAccessUnlessGranted('edit', $articulo);  
  
$this->denyAccessUnlessGranted('delete', $articulo);
```

Consultas

Con respecto a los controladores de usuario y artículos, añadí referencias a consultas realizadas en sus respectivos repositorios, para así mostrar una serie de elementos en cada página. Gracias a Doctrine, Symfony provee al proyecto todas las herramientas



necesarias para trabajar con base de datos. Algunas de las consultas que realicé:

Artículos:

Introducí unos cambios en los repositorios, en los métodos `findAll()`, para que, mediante consultas, me mostrase los datos en un orden específico.

```
$articulos = $entityManager->getRepository( Artículo::class )->findAll();
$categorias = $entityManager->getRepository( Categoria::class )->findAll();
$idiomas = $entityManager->getRepository( Idioma::class )->findAll();
$autores = $entityManager->getRepository( Usuario::class )->findAll();
```

```
public function findAll()
{
    return $this->findBy(array(), array('...'));
}
```

Donde ‘...’ variará según el repositorio en el que se encuentre.

Las siguientes consultas las hice en el repositorio de artículos.

Artículos por categoría:

```
$articulos = $entityManager->getRepository( Artículo::class )->
>mostrarArticulosPorCategoria ($idCat);
```

Artículos por idioma:

```
$articulos = $entityManager->getRepository( Artículo::class )->
>mostrarArticulosPorIdioma ($id);
```

Artículos por autor:

```
$articulos = $entityManager->getRepository( Artículo::class )->
>mostrarArticulosPorAutor ($id);
```

Usuarios:

Mostrar perfil. Como en el perfil quería que se mostrase tanto el usuario como los artículos que ha creado, llamo a ambas entidades. En esta función hago uso de una consulta que hice para mostrar los artículos de un usuario en concreto en el repositorio de Artículo (`mostrarArticulosPorAutor`), la cual está indicada en el apartado Artículo. Una vez obtenida dicha información, hago que se muestre en la plantilla `usuario/verPerfil.html.twig`

```
$articulos = $entityManager->getRepository( Artículo::class )->
```



```
>mostrarArticulosPorAutor($id);
```