

MEMORIA SISTEMAS INFORMÁTICOS: PRÁCTICA 1

Guía detallada de ficheros

Dentro del directorio principal nos encontramos los siguientes directorios y archivos:

- start.wsgi: que nos permite ejecutar con Apache la aplicación web.
- Memoria-P1: este fichero.
- /app: fichero con los datos y programas de la aplicación.

Dentro de este último repositorio /app tenemos lo siguiente:

- catalogue/catalogue.json: json con el catálogo de películas de la aplicación.
- static/css: contiene el fichero .css común para todos los templates html.
- static/js: que a su vez contiene:
 - js/ajax.js: código ajax que nos permite comunicar asíncronamente con el servidor el número de clientes online.
 - js/historial-toggle.js: JQuery que nos permite desplegar pedidos en el historial.
 - js/password-strength.js: JQuery que calcula la fuerza de la contraseña y la muestra en la barra de progreso dinámicamente.
 - js/register.js: JavaScript que valida los campos de registro del usuario.
- static/img: imágenes que cargar.
- templates/: directorio que contiene todos los templates que se muestran como capturas más adelante, junto con su estilo.
- usuarios/: carpeta donde se crearán los directorios con los nombres de usuario, dentro de estos directorios se encontrará el fichero datos.dat, que contiene el usuario, el salt y password encriptados, el email, la tarjeta y el saldo. También se creará el historial.json de cada usuario con los pedidos que ha realizado.
- __init__.py: inicializa Flask y Flask Sessions.
- __main__.py: ejecuta la app.
- routes.py: fichero python con las funciones views que se enlazan a las urls, y funciones auxiliares que se explican y documentan en el propio código.

Instrucciones de uso e Implementación:

Para poder ejecutar la plataforma necesitaremos crear un entorno virtual de python3 en el directorio principal, que tiene que ser public_html, llamado si1pyenv en el que tengamos python 3.6 y flask instalados para poder ejecutar la aplicación.

La aplicación se puede ejecutar desde la terminal en public_html con el siguiente comando:

```
$python3 -m app
```

Y accediendo a la url <http://0.0.0.0:5001/index> podemos ir a la página principal de la aplicación y hacer lo que deseemos.

Otra opción es mediante el servidor Apache, antes de ejecutarlo necesitamos darle permisos a los directorios de nuestra aplicación para que no de error al crear los directorios de los usuarios. Desde la terminal en public_html podemos ejecutar

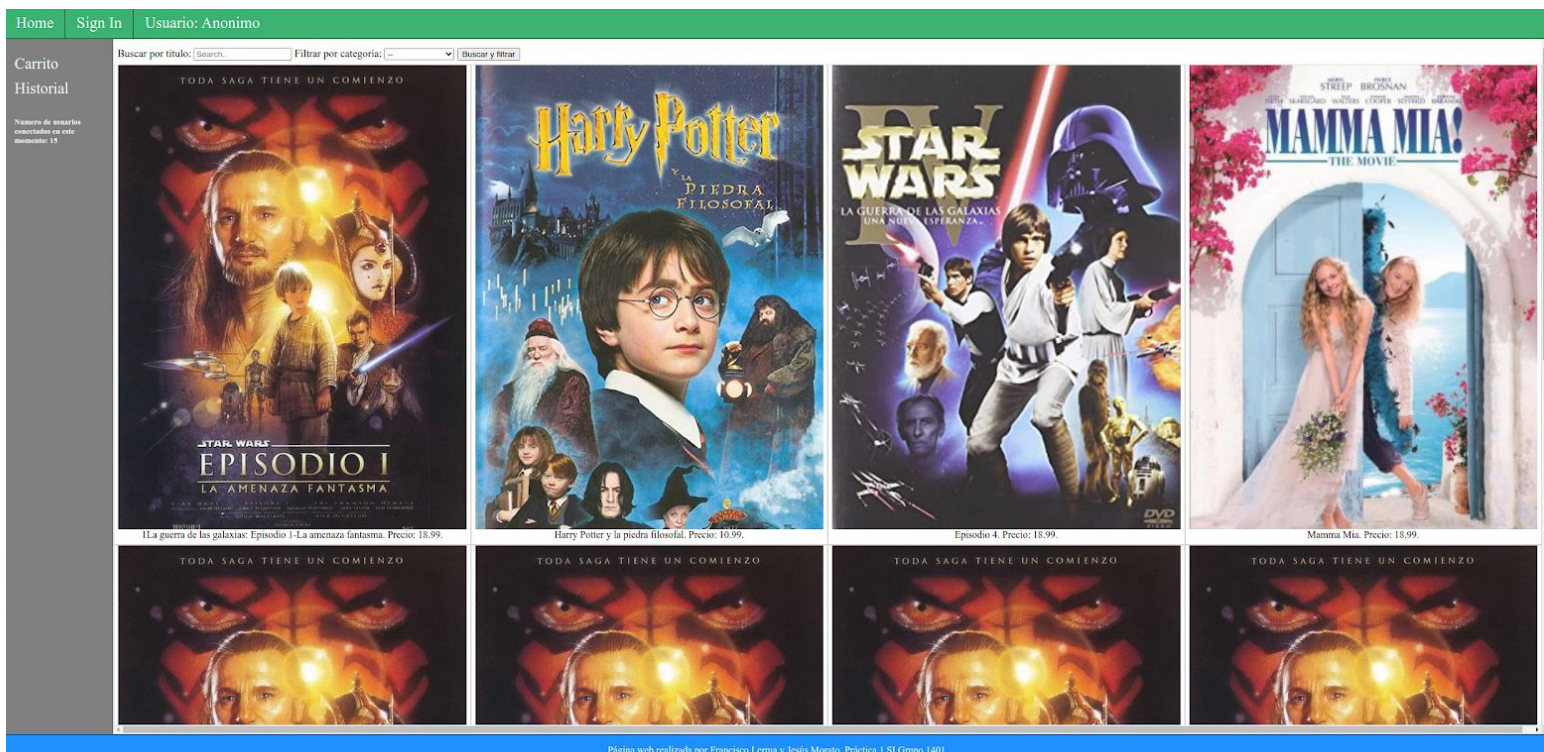
```
$chmod -R 777 ./
```

comando con el damos los permisos necesarios. Una vez hecho si accedemos la url <http://localhost/~eps/start.wsgi/> accederemos mediante Apache a la aplicación.

Detalles de implementación de las páginas:

INDEX (/index)

En esta página se muestra la página principal que contiene diferentes partes. En el header se encuentran enlaces para el home(esta misma página), otro para iniciar sesión y por último el nombre de usuario de la sesión o “Anónimo” si no hay sesión iniciada. En la barra lateral se encuentran enlaces para el carrito, para el historial y una sección de texto que se actualiza de manera dinámica mediante ajax (fichero js/ajax.js) el número de clientes online en la aplicación. También tiene un foot que muestra la información de los autores. Por último hay una sección de contenido que varía en cada página. En esta página se muestran las imágenes, que hacen de enlace a para su página de detalles, para ver las diferentes películas que tiene de catálogo la aplicación. La página es generada por index() en routes.py.



Encima de las películas tenemos la opción de buscar por título en las películas y de filtrar por categoría mediante un select. Se puede buscar y filtrar a la vez. Cuando se hace submit se llama a las funciones `busqueda_titulo()`, `filtrar()` y `search()` de routes.py mediante la url “/search”.

SIGN IN (/login)

Esta página es generada por login() en routes.py. Contiene un formulario que permite a los usuarios iniciar sesión. Tiene un campo con el username y otro con la contraseña. Si el usuario no existe o la contraseña es incorrecta nos notificará el error. Si las credenciales son válidas nos redirigirá al index con el usuario ya iniciado en sesión. También contiene un link que permite a los usuarios registrarse.

Si hemos iniciado sesión previamente se guardará el usuario en una cookie que auto completará el input del username en las próximas sesiones.

REGISTER (/register)

Es generada por registert() en routes.py. Nos devuelve un formulario que permite a los clientes registrarse en la web. Nos pide los siguientes campos que son validados por JavaScript en el cliente, que se encuentra en el fichero register.js:

- Username: comprueba que no esté vacío, no tenga espacios y sea alfanumérico.
- Password y Repeat Password: es la contraseña, se pide que tenga al menos 8 caracteres, no esté vacía y que coincidan las dos.
- Password strength: es una barra de progreso que mediante JQuery(password-strength.js), se va completando según la fortaleza de la contraseña (longitud, minúsculas, mayúsculas, números, símbolos especiales, etc).
- CreditCard: se pide que sea numérica y tenga una longitud de entre 13 y 18 dígitos.
- Email: solo se pide que sea de la forma string@string.
- Condiciones: checkbox que acepta las condiciones.

Si todos los parámetros son validados se crea el usuario, y se crea su directorio con su .dat su historial.json. Si el usuario ya existe nos dará error.

La página de Register no inicia sesión, así que tendremos que iniciar sesión después de registrarnos.

The screenshot shows a web application interface. At the top, a green navigation bar contains links for 'Home', 'Sign In', and 'Usuario: Anonimo'. On the left, a dark grey sidebar displays 'Carrito' and 'Historial', along with a statistic: 'Numero de usuarios conectados en este momento: 18'. The main content area is titled 'Registration menu' and contains several form fields: 'Username:' with a placeholder 'Enter username (Only alphanumerical, no special characters)', 'Password:' with 'Enter password', 'Password Strength:' with a progress bar, 'Repeat Password:' with 'Enter password again', 'Credit Card:' with 'Enter your credit card number', and 'Email Address:' with 'Enter your email address'. Below these fields is a checkbox labeled 'By selecting the box, I agree to the Terms and Conditions.' and a green 'Register' button. A blue footer at the bottom reads 'Página web realizada por Francisco Lerma y Jesús Morato. Práctica 1 SI Grupo 1401.'

DETALLES (/details-<id>)

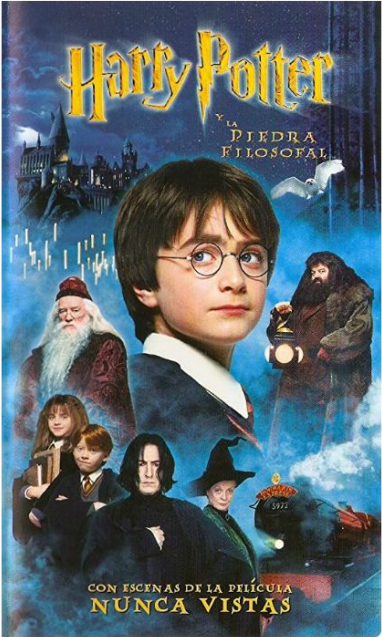
En esta página se muestran los detalles de la película que se ha seleccionado en la página de index. y vemos en la captura que se muestra una foto de la película, así como el título, director, categoría, precio, año y actores (con su respectivo nombre, personaje e imagen del mismo). Además la página se interpreta con el archivo html deatils.html, y se carga a través de la función details() en routes.py, esta función recibe como argumento el id de la película a visualizar que se mapeara en la URL en la que se muestra la página.

[Home](#) | [Sign In](#) | Usuario: Anonimo


[Carrito](#)
[Historial](#)



Numero de usuarios
conectados en este
momento: 11

Harry Potter y la piedra filosofal



CON ESCENAS DE LA PELÍCULA
NUNCA VISTAS

- **Director:** George Lucas
- **Categoría:** Aventura
- **Precio:** 10.99
- **Año:** 1999
- **Actores:**
 - Nombre: Daniel Radcliffe
 - Personaje: Harry Potter
- Nombre: Emma Watson
- Personaje: Hermione Granger



Vemos que esta página cuenta con un único botón, “*Añadir al carrito*”, el cual al pulsarlo añade automáticamente al carrito la película de la cual se muestran los detalles, esto se hace a través de la función `add_carrito()` en `routes.py` la cual cogera el id de la película a través de un request get.



CARRITO (/carrito)

En esta página se muestra el carrito de la compra tanto del usuario registrado como del usuario anónimo, aunque con diferentes funcionalidades entre ambos. Vemos que se muestran las diferentes películas que hay en el carrito y la cantidad de ellas. El código HTML de la página que se muestra en se encuentra definido en `carrito.html`, además para una vista previa del carrito al dar al botón carrito de la base se crea con la función `carrito()`.

[Home](#)
[Log out](#)
Usuario: roberto

Carrito
Historial

Numero de usuarios conectados en este momento: 68

Carrito de películas

Saldo: 23 €

<p>1.La guerra de las galaxias: Episodio 1-La amenaza fantasma Precio Unidad: 18.99</p> <p>Cantidad: 3</p> <div> <div>Actualizar cantidad</div> <input type="text" value="Enter amount of films"/> </div> <div>Eliminar Película</div>
<p>1.Harry Potter y la piedra filosofal Precio Unidad: 10.99</p> <p>Cantidad: 1</p> <div> <div>Actualizar cantidad</div> <input type="text" value="Enter amount of films"/> </div> <div>Eliminar Película</div>
<p>1.Mamma Mia Precio Unidad: 18.99</p> <p>Cantidad: 2</p> <div> <div>Actualizar cantidad</div> <input type="text" value="Enter amount of films"/> </div> <div>Eliminar Película</div>

Precio total: 105.94

Comprar

En esta captura podemos observar que tenemos un input donde se puede actualizar directamente el número de películas de un título que queremos comprar. Esta funcionalidad se implementa con la función `act_carrito()`, la cual hará un get de la cantidad que se introduce en el text input. esta función se llama al pulsar el botón “Actualizar cantidad”.

Cantidad: 3

Actualizar cantidad

Además tenemos un botón de “*Eliminar película*”, el cual, si tenemos más de una película de ese tipo bajará la cantidad en una unidad y si tenemos únicamente una la eliminará del carrito. Desarrollada en la función `rmv_carrito()` que hará un get de la película a la cual restará una unidad su cantidad.

Además también se muestra el saldo del usuario en caso de que esté registrado (y 0 euros si no lo está) y el precio que tiene el total del carrito.

Saldo: 23 €

Es necesario registrarse para esta funcionalidad

Registrarse

Por último, tenemos el botón de “*Comprar*”, el cual si el usuario no está registrado, saldrá un mensaje que dice que es necesario registrarse y un botón con un link a la página de registro, si el carrito está vacío a la hora de dar a comprar saldrá un mensaje de carrito y vacío, y por último si el usuario está registrado y tiene películas le llevará a la página de historial habiendo procesado el carrito y convirtiéndolo en un nuevo pedido (actualizando

por correspondiente el saldo del usuario). Esta funcionalidad está desarrollada en la función `comp_carrito()`, la cual en el caso de que sea un usuario registrado y exista el carrito redireccionará a la página del historial una vez haya actualizado el json del usuario correspondiente añadiendo el nuevo pedido.

No hay carrito

HISTORIAL (/historial)

En esta página se mostrará el historial de pedidos que tiene el usuario registrado, se implementa con el HTML `historial.html` y la función que utilizamos para mostrar este historial y que llama a este archivo html es `historial()` en `routes.py`. Esta función tendrá acceso al correspondiente json de los pedidos del usuario registrado en la carpeta `usuarios/<usuario-registrado>/historial.json` la cual se encargará de leerlo y de mostrarlo a través del HTML por pantalla en la página.

[Home](#) [Log out](#) Usuario: prueba

Carrito
Historial

Saldo total: 768€

Add cash €:

Pedido número 1.

- Número del pedido: 1
- Fecha del pedido: 28 de Octubre del 2020
- Películas compradas:

Id	Título	Precio	Cantidad
2	Harry Potter y la piedra filosofal	10.99€	4
4	Mamma Mia	18.99€	5

- Precio total del pedido: 138.91€.

Pedido número 2.

- Número del pedido: 2
- Fecha del pedido: 28 de Octubre del 2020
- Películas compradas:

Id	Título	Precio	Cantidad
2	Harry Potter y la piedra filosofal	10.99€	1
4	Mamma Mia	18.99€	5

- Precio total del pedido: 105.94€.

Vemos que en esta página tenemos también la opción de actualizar el saldo del usuario registrado, en este caso se hará uso de la función `add_saldo()`. Mediante JQuery en `historial-toggle.js` hacemos que se pueda desplegar los diferentes pedidos.

Saldo total: 768€

Add cash €: