

Práctica 2. Programación

Web dinámica de un sistema de venta de DVDs

TuPelicul  N

Grupo: 1363

Tejada Paredes, Alejandro

alejandrom.tejada@estudiante.uam.es

Villa Gómez, Julián

julian.villa@estudiante.uam.es

ÍNDICE

- ENTRADA: PÁGINA PRINCIPAL
 - ACCESO DE UN USUARIO
 - REGISTRO DE USUARIO
 - DETALLE PELÍCULA
 - CARRITO DE LA COMPRA
 - FINALIZAR COMPRA DE DVD
- MOSTRAR HISTORIAL DEL USUARIO
 - OTRAS FUNCIONALIDADES
 - Desplegable
 - Fortaleza de la contraseña
 - Número de usuarios conectados
- FICHEROS QUE CONFORMAN EL PROYECTO
 - REFERENCIAS

Entrada: Página Principal

A la hora implementar todas las funcionalidades para la página principal, primero partimos del template estático que habíamos creado en la anterior práctica, introduciendo en esta nueva vista tanto el jinja como el catalogo.json, para que todas las películas que tiene nuestra aplicación sean mostradas en el inicio, de manera que, se muestren 5 películas por fila, en un principio la distribución no presentaba ningún problema, sin embargo, cuando decidimos añadir un par de películas a nuestro catálogo, nos dimos cuenta que, cuando se desplegaban, algunas no se iban a la izquierda y por tanto se quedaba en el espacio donde el contenedor pudiese caber, y esto se debía que dentro del contenedor que contenía las imágenes de las películas y su título, dependiendo de lo largo que era uno u otro título de la película este se hacía más grande o más pequeño, la solución que tomamos fue ponerle un tamaño definido al contenedor que tenía el título y así de esta manera todos los contenedores de las películas tendrían el mismo tamaño(Imagen1).

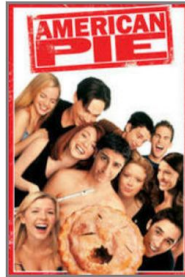
En nuestra implementación inicial y de la cual nos hemos ido guiando para hacer las vistas de nuestra aplicación, habíamos decidido que daría igual en la vista que te encontrases que siempre va a aparecer una barra de navegación, el logo de la página, la barra de búsqueda, la barra desplegable para filtrar por categoría, un botón para realizar la búsqueda, el icono del carrito y un enlace para iniciar sesión/registrarse, en caso de no estar logeado y en caso de estarlo un enlace para desloguearte y un mensaje de bienvenida. Por tanto y siendo fieles a nuestra idea base, tanto como la barra de búsqueda como la barra de filtro y el botón, han sido implementados dentro del fichero base.html, en el cual se encuentran los elementos comunes de todas la vistas de nuestra página. Para implementar la barra de búsqueda, el desplegable y el botón de búsqueda, hemos empleado la etiqueta "form", dentro de esta hemos utilizado la etiqueta "input" con type = texto para la barra de búsqueda y un "input" con type = submit para el botón de búsqueda, pero todo esto solo es la implementación visual, para la implementación que ejecuta la búsqueda, hemos definido un método dentro de nuestro fichero routes.py, que captura lo que se ha escrito en la barra de búsqueda y la categoría que se ha elegido, y con estos datos filtra entre todas las películas de nuestro catálogo(Imagen2), en caso de que no se introduzca ningún texto en la búsqueda pero si se seleccione alguna categoría con la que filtrar, busca todas las películas que tengan esa categoría como género y las muestras(Imagen3). Por último, también busca solo por la cadena de caracteres introducidas en el campo de la búsqueda(Imagen4).



El Rey León



Los Increíbles 2



American Pie



Vengadores: Infinity War



Titanic



Harry Potter y las reliquias de la muerte - Parte II



Érase una vez en... Hollywood



Star Wars VIII: Los últimos Jedi



Spider-Man: Into the Spider-Verse



Pokémon: Detective Pikachu



La Monja

Imagen 1



Titanic

Imagen 2

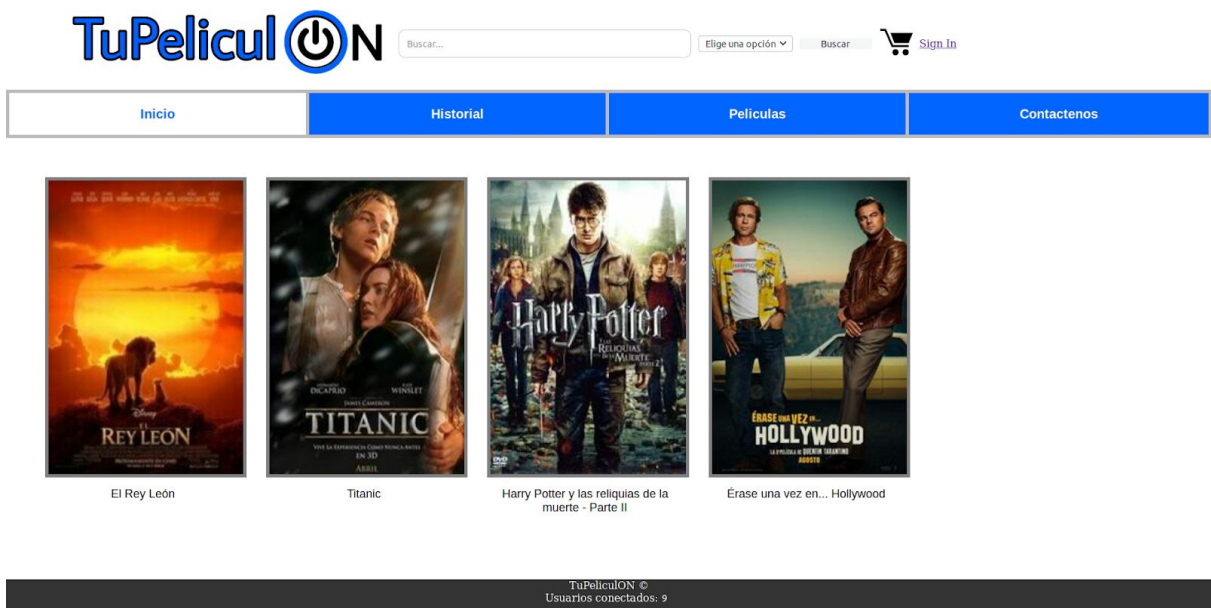


Imagen 3

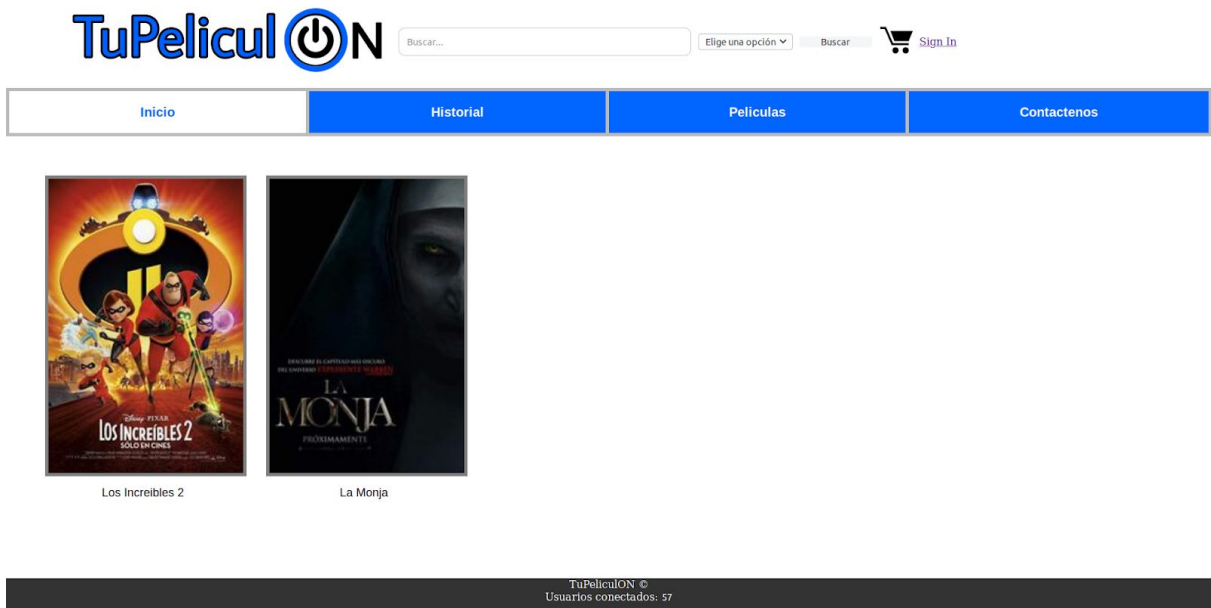


Imagen 4

Acceso de un Usuario

Para implementar el acceso del usuario dentro del sistema, la vista destinada para esta funcionalidad es compartida con la vista destinada para el registro. El acceso del usuario se encuentra en la zona derecha de la vista y cuenta con 2 campos para rellenar y un botón para hacer efectiva el inicio de sesión, el primer campo está destinado a que el usuario introduzca su nick(nombre dentro del sistema) y su contraseña. Para realizar esta implementación hemos utilizado la etiqueta “form” dentro de esta 2 inputs uno para cada campo y un último input de tipo submit para el botón de acceso(Imagen5). Dentro del routes.py hemos modificado el método llamado login que venía ya en el ejemplo de moodle dentro de este comprobamos si los campos que ha rellenado son los correctos, es decir, comprueba si la carpeta de nombre con el nombre introducido existe, en caso de que no exista lanza una alerta al usuario, en caso que exista, internamente capturamos la contraseña codificada y comparamos si es la misma que introducido, en caso de que no sean la misma, lanza una alerta al usuario(Imagen6), y en caso de que exista el sistema recupera los datos del usuario, es decir, su historial de compras, el carrito, sus datos y su depósito de dinero, una vez realizado todo esto te redirige a la página de inicio, donde en la parte superior derecha se verá un mensaje de bienvenida y un enlace para hacer el logout(Imagen7). Una vez logueado podrás comprar películas del sistema.

Cuando el usuario decida desloguearse se guarda en una cookie la información, para si luego quiero volver a loguearse aparezca su nombre dentro del campo de nick, a la hora de iniciar su sesión.

The screenshot shows the user interface for 'TuPeliculON'. At the top, there is a navigation bar with a logo, a search bar, and links for 'Inicio', 'Historial', 'Películas', and 'Contactenos'. Below this, the page is divided into two main sections: 'Regístrate' (Register) and 'Iniciar sesión' (Login). The 'Regístrate' section contains four input fields: 'Nombre de usuario', 'Contraseña', 'Repita su contraseña', and 'E-mail', followed by a 'Tarjeta de crédito' field. A red error message 'No permitida' is visible next to the password field. A blue 'Registrarse' button is at the bottom of this section. The 'Iniciar sesión' section contains two input fields: 'Nick' and 'Contraseña', followed by a blue 'Login' button. At the bottom of the page, a dark footer bar displays 'TuPeliculON ©' and 'Usuarios conectados: 31'.

TuPeliculON [Sign In](#)

Inicio	Historial	Películas	Contactenos
--------	-----------	-----------	-------------

Regístrate

Nombre de usuario:

Contraseña:

No permitida

Repita su contraseña:

E-mail:

Tarjeta de crédito:

Iniciar sesión

Nick:

Contraseña:

TuPeliculON ©
Usuarios conectados: 31

Imagen 5

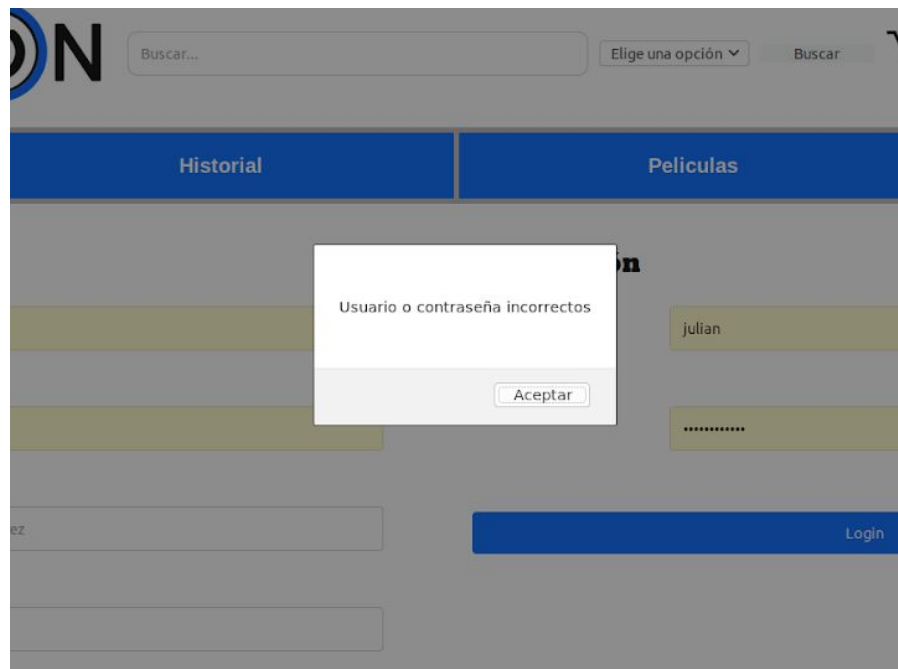


Imagen 6

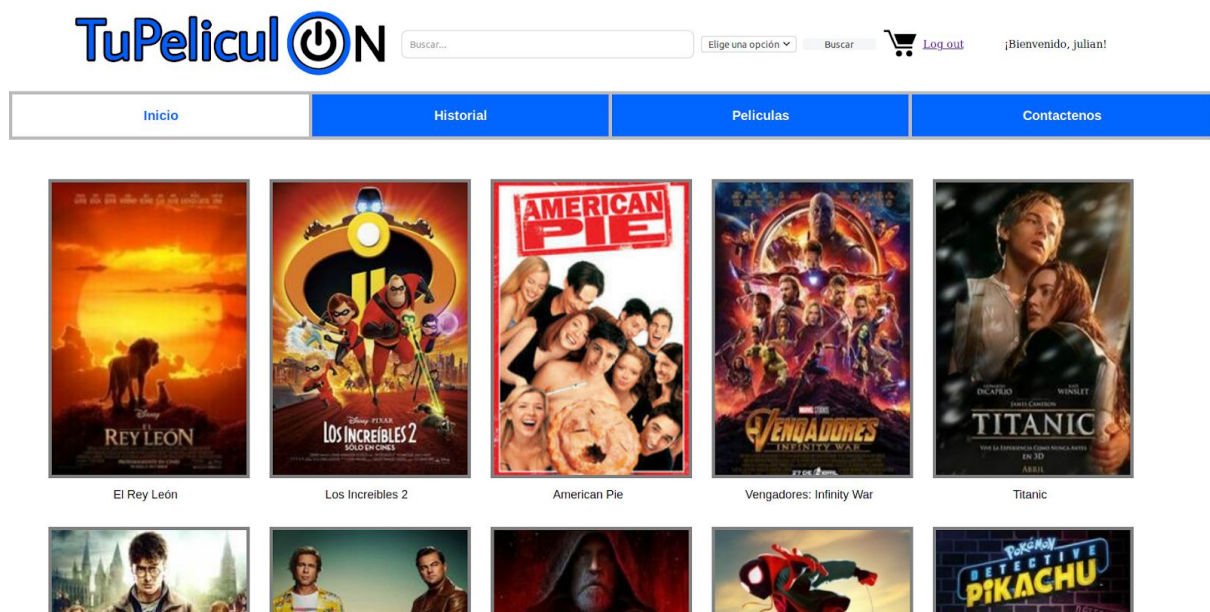



imagen 7

Registro de usuario


Para implementar el registro de usuario, como ya hemos mencionado antes decidimos compartir la misma vista tanto para el registro como para el inicio de sesión, por tanto el formulario dedicado al registro de un usuario se encuentra a la derecha de la vista. El contenedor del registro del usuario consta de 5 campos(nick, contraseña, repetir la contraseña, email, tarjeta de crédito) y un botón de acceso(Imagen8). Para registrarte en nuestro sistema los campos rellenos tienen que cumplir una serie de condiciones que serán verificadas una a una a la hora de pulsar el botón de acceso, las condiciones a cumplir son las siguientes(Imagen9):

1. Todos los campos tienen que estar rellenos.
2. La contraseña del usuario tiene que tener al menos 8 caracteres.
3. La 1ª contraseña introducida tiene que coincidir con la 2ª.
4. El nick tiene que tener caracteres válidos.
5. El número de la tarjeta tiene que tener caracteres válidos(sólo números) y tiene que tener 16 caracteres.

Para realizar estas comprobaciones tenemos un fichero .js, el cual recibe el nombre de functions.js y lo comprueba todo mediante javascript, en el método “verifyRegister”, donde capturamos los valores introducidos por el usuario y con sentencias condicionales y expresiones regulares verificamos que los campos tenga la longitud, el lenguaje admitido o ambas. Una vez realizadas estas comprobaciones satisfactoriamente, internamente en el método register del módulo routes.py, primero se comprueba si la carpeta ya existe con el nombre del nick introducido por el nuevo usuario, es decir, si el nick ya lo posee otro usuario del sistema, en caso de que sea así envía una alerta del usuario comunicándole que ya existe un usuario con ese nombre, en caso contrario, continúa, creando la carpeta para el nuevo usuario, dentro de esta carpeta creamos un fichero de nombre data.dat donde almacenamos la información del usuario, es decir, los campos introducidos por en el formulario, teniendo en cuenta que la contraseña se ha cifrado antes de la escritura de estos datos en el fichero, además se le asigna un saldo inicial que puede variar entre 0 y 100, posteriormente creamos también un fichero .json, el cual recibe el nombre de historial.json, donde guardamos en principio un diccionario vacío, ya que este posteriormente contendrá el historial de películas adquiridas por el usuario, ya por último informa al sistema de la presencia de un usuario conectado en línea(Imagen10).



Elige una opción


[Sign In](#)

Inicio	Historial	Películas	Contactenos
--------	-----------	-----------	-------------

Regístrate

Nombre de usuario:

Contraseña:

No permitida

Repita su contraseña:

E-mail:

Tarjeta de crédito:


Iniciar sesión

Nick:


Contraseña:

TuPelículON ©
 Usuarios conectados: 31

Imagen 8



Elige una opción


[Sign In](#)

Inicio	Historial	Películas	Contactenos
--------	-----------	-----------	-------------

Regístrate

Nombre de usuario:

Contraseña:

Repita su contraseña:

E-mail:

Tarjeta de crédito:

Iniciar sesión

Nick:

Contraseña:

TuPelículON ©
 Usuarios conectados: 66

Imagen 9

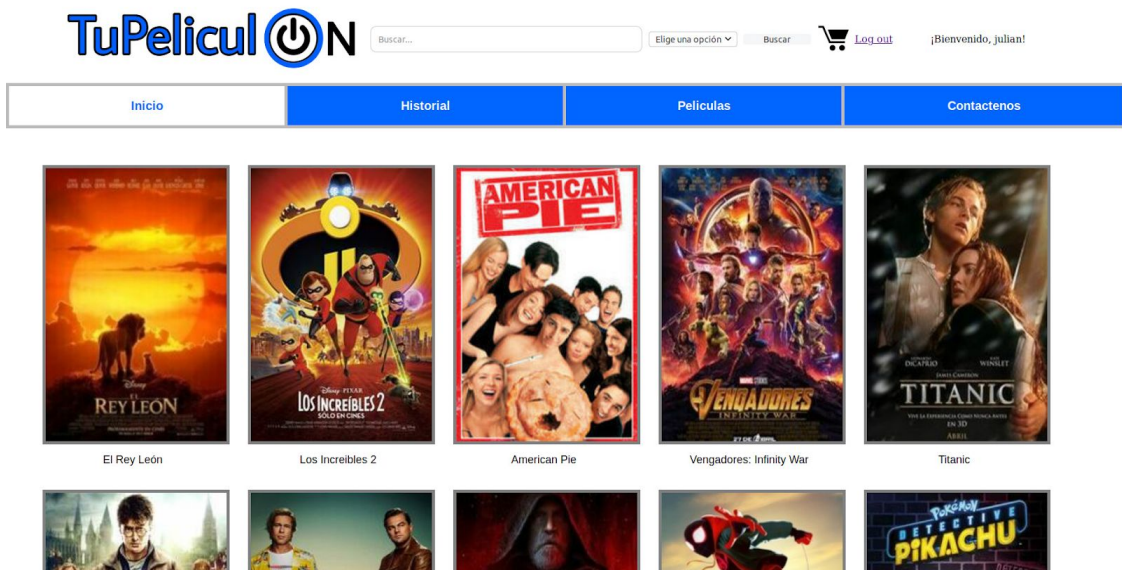


imagen 10

Detalle película

Para implementar el detalle de la película, en la vista hemos decidido colocar dentro del contenedor principal, tanto como el cartel de la película, el icono de añadir al carrito y la descripción de la película, es decir, su nombre, su sinopsis, sus actores, su precio, su duración; que se encuentran dentro del fichero catalogo.json(Imagen11). Para poder capturar todos estos atributos de la película de la cual queremos ver el detalle, desde el index por ejemplo, en el inicio cada imagen de la película es un formulario que tiene 2 inputs uno de tipo imagen que es el cartel de la película y otro input de es de tipo oculto, el cual tiene en su atributo valor el id de la película, llegar a cabo esta implementación que parece sencilla, nos llevó más tiempo del que esperábamos ya que, en ese entonces no teníamos muy claro cómo capturar ese id. Internamente en el sistema, dentro del módulo routes.py en el método detalle, el cual definimos para este propósito, cargamos catálogo y capturando el valor del campo oculto(que es el de la película seleccionada), iteramos en búsqueda de el id que le corresponde a ese ide, el mismo, y una vez encontrado lo devolvemos a la vista la película, para que ya en ella se pueda trabajar con el contenido de la película, mediante html y jinja.

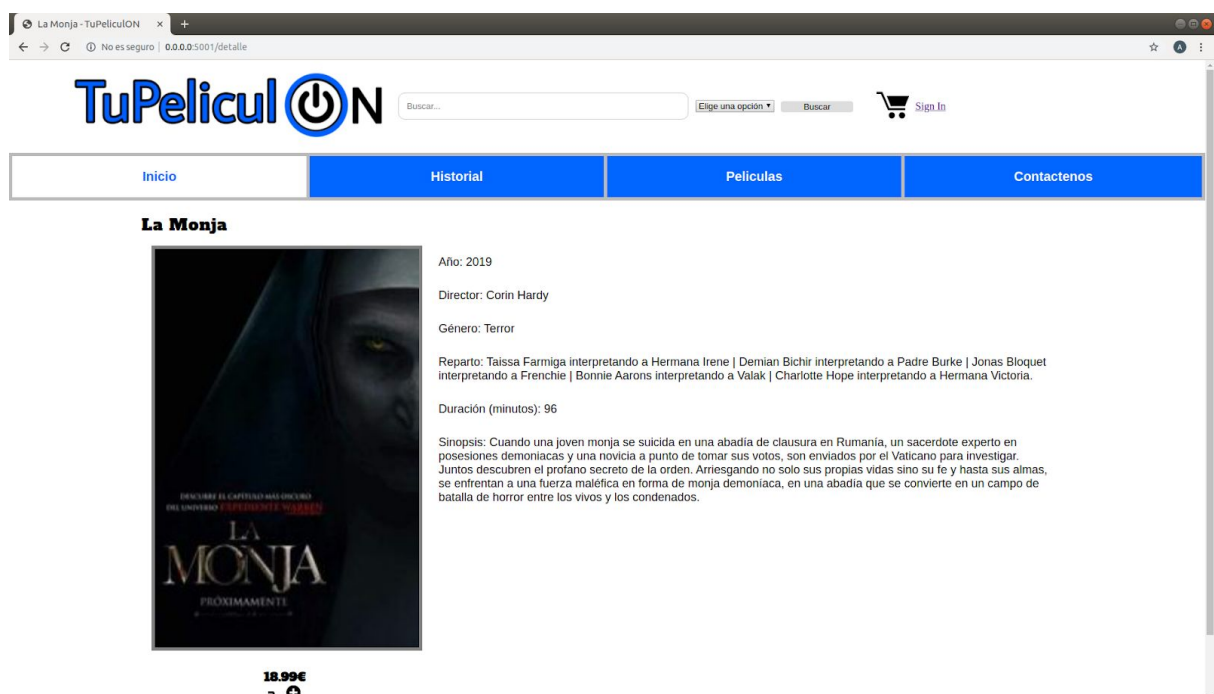


Imagen11

Carrito de la compra

Como ya se ha mencionado antes, el carrito de la compra está disponible desde todas las vistas, y la función para añadir una película al carrito solo se puede realizar accediendo al detalle de la película. Todo usuario bien sea anónimo o no, puede añadir películas a su carrito de compra y ver qué películas tiene añadidas al carrito de la compra sin embargo, los usuarios que no estén registrados es decir los anónimos no podrán efectuar la compra, en caso de que un usuario anónimo quiera efectuar una compra se le enviará la vista de registro/inicio de sesión para que se pueda identificar en el sistema, esto se explicará más detalladamente más adelante, una vez identificado el usuario podrá ver que las películas que ha añadido al carrito se mantienen, se explicará más detalladamente más adelante, y recién podrá efectuar la compra.

Empezaremos explicando la implementación de añadir al carrito, como ya se ha mencionado en el detalle hay un icono que corresponde a añadir película al carrito (Imagen12), este es un formulario que siguiendo la filosofía del form con imagen y campo oculto transfiere la id al método `add2car`, en el cual se crea una lista donde se irán guardando los ids de las distintas películas que han sido añadidas al carrito. Para ver el contenido del carrito, es necesario pulsar el icono que se encuentra en la parte superior de la página junto al botón de búsqueda (Imagen13), este internamente está vinculado con el método `carrito`, el cual captura la lista de ids de las películas añadidas al carrito y también, capturando las películas de sistema itera buscando coincidencias entre películas, mientras calcula el precio total de las películas del carrito, a las películas las mete en una lista que será devuelta a la vista del carrito donde se desplegará la tabla con los elementos del carrito (Imagen14), si por algún casual el usuario quiere eliminar algún elemento del carrito, tiene a su disposición un icono de cubo de basura (Imagen15), que internamente está implementado mediante el método `delitem` el cual captura el id de la película a eliminar y la elimina de la lista.

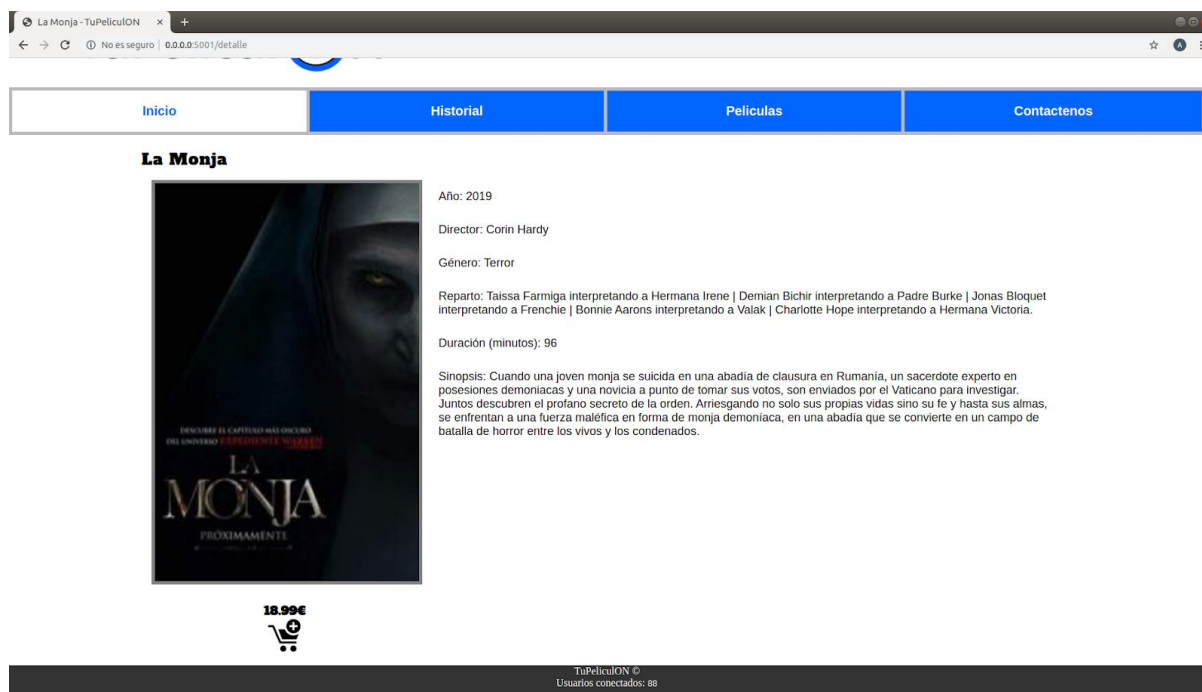


Imagen12

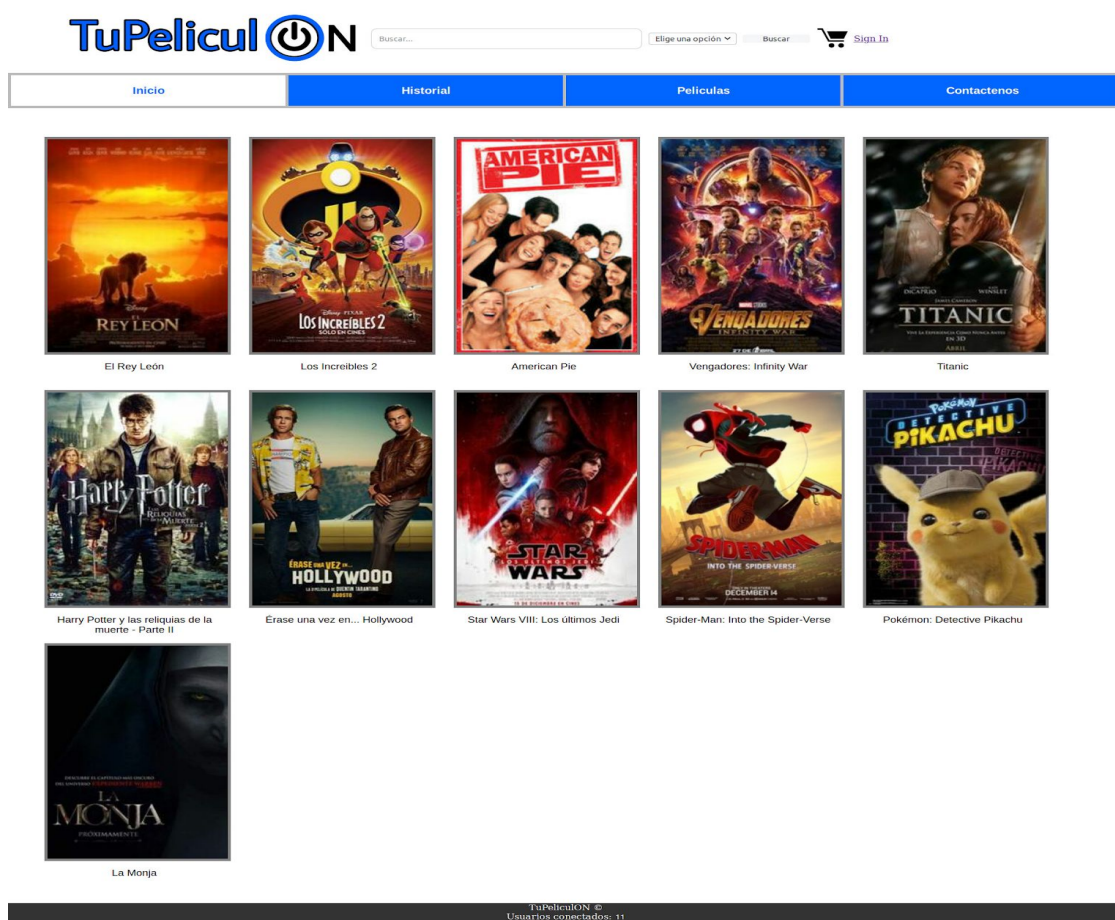


Imagen13

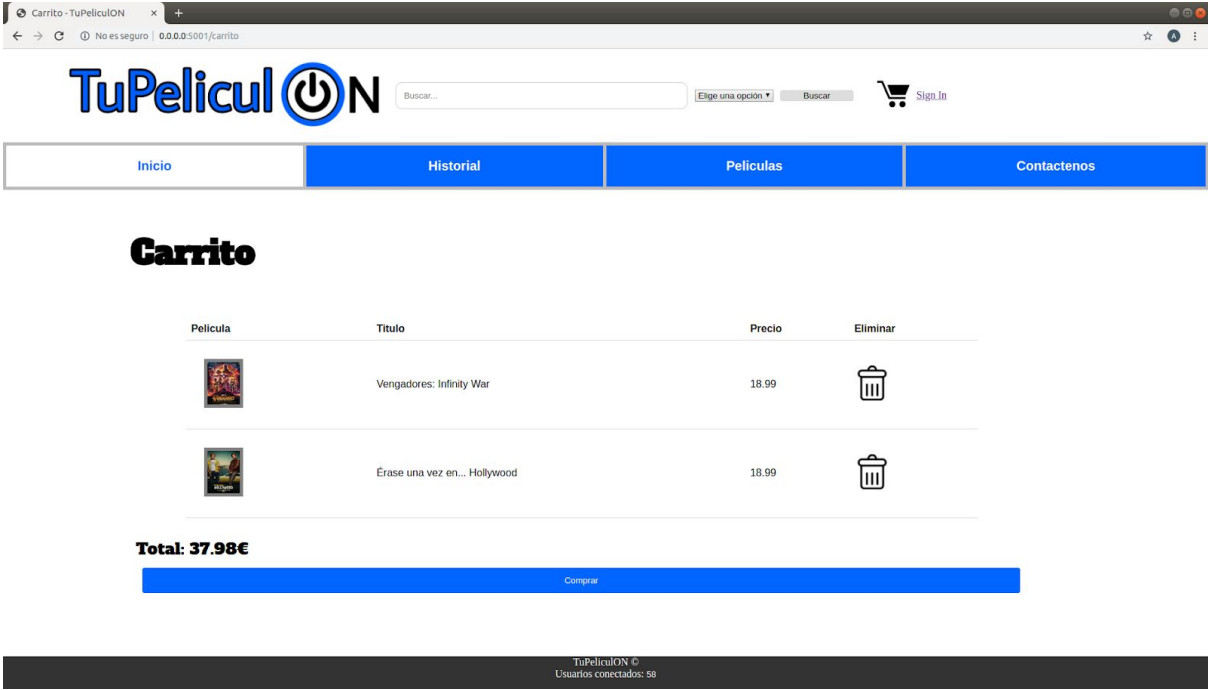


Imagen14

Finalizar compra de DVD

Como ya se ha mencionado antes para finalizar la compra de un carrito el usuario tiene que estar identificado en el sistema, si el sistema detecta que el usuario no está identificado en el sistema, el sistema lanzará una alerta al usuario y le enviará a la página de registro. Cuando un usuario envía la petición de finalizar compra al sistema este evalúa dicha petición mediante el método fincompra, donde se cargan los datos del usuario que quiere hacer la compra, en caso de que el usuario sea anónimo no los carga y le impide la compra como ya se ha mencionado anteriormente, capturamos su saldo y el precio total del carrito, comprobamos que el saldo sea igual o mayor al precio del carrito, y en caso afirmativo procedemos a descontar el precio de la compra del y guardar estos cambios en el fichero de datos del usuario, también, empezamos a escribir el fichero historial.json, en el que se guardan las películas que han sido compradas, por último limpiamos el carrito y se vuelve a la vista del carrito, mientras que se lanza una alerta al usuario de que su compra ha sido realizada(Imagen16).

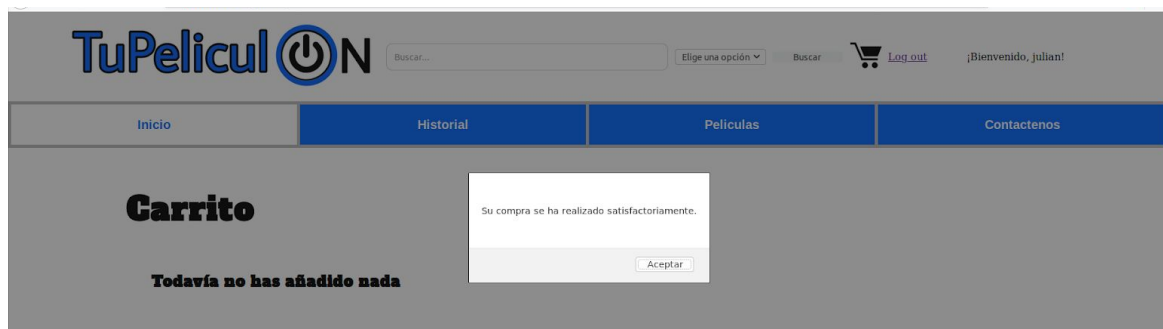


Imagen 16

Mostrar historial del usuario

Como ya hemos mencionado antes el historial, muestra el historial de compras, para acceder a solo hace falta estar registrado y en la barra de navegación hay un campo destinado al acceso al historial(Imagen17), cuando el usuario manda la solicitud para acceder al historial, internamente el sistema entra en el método historial, en el cual lo primero que comprueba si hay un usuario logueado, en caso de que no lo haya, el sistema enviará al usuario a la vista de registro, en caso afirmativo, es decir, que si exista un usuario, el sistema cargará su fichero historial.json en una lista y esta será enviada a la vista de historial donde será desplegada mediante una tabla(Imagen18), dentro de esta tabla se mostrará al usuario, el poster de la película, su nombre y su precio, además, de un cuarto campo que se un desplegable en el que se muestra más información de la compra, esto será explicado con mayor detalle más adelante. Por último al final de la tabla habrá un campo donde te informará de tu saldo actual, y un input donde podrás escribir la cantidad de dinero que quieras agregar a tu saldo, junto a un botón para hacer efectivo el incremento, internamente el sistema realiza esta operación en el método addmoney en el que se carga el fichero de datos del usuario y se modificará el campo saldo del usuario.

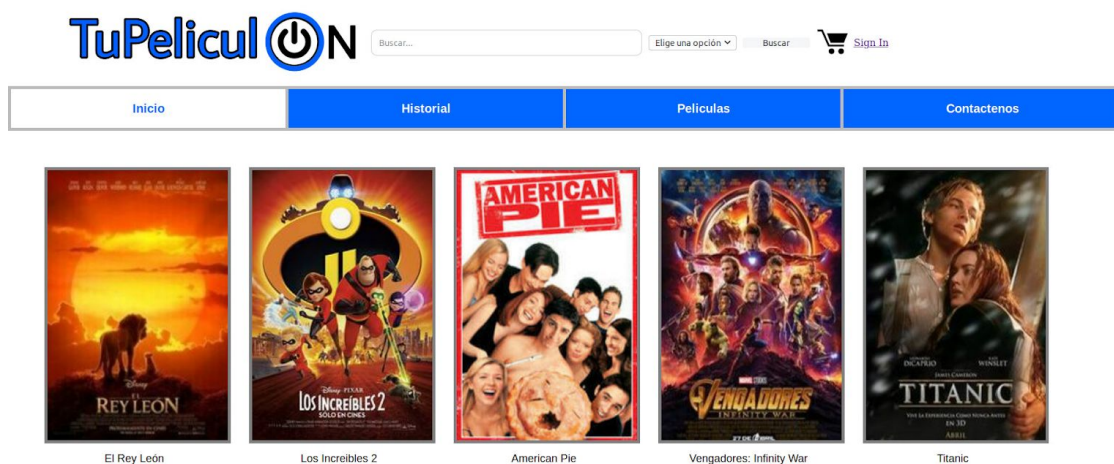


Imagen 17

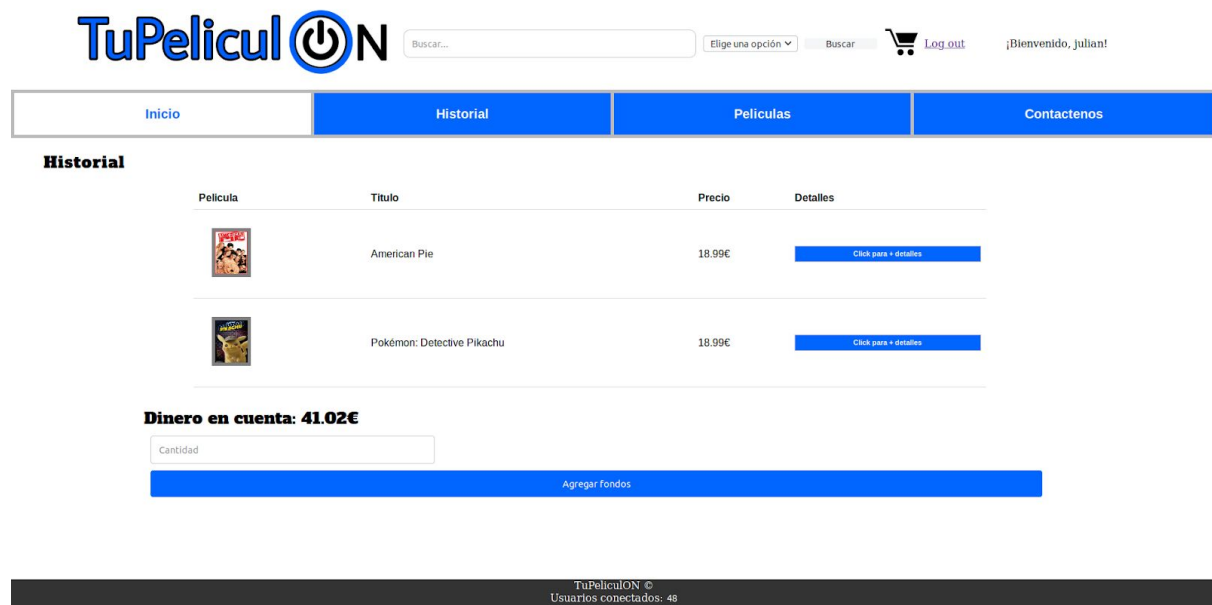


Imagen 18

Otras funcionalidades

1. Desplegable

Para implementar el desplegable que mostrará detalles ocultos propios de la compra de una película, hemos empleado jquery, la implementación se encuentra dentro del fichero del functions.js, este interactúa de tal manera que cuando el usuario ejecuta un click sobre el mensaje, se despliega un campo en el que aparecen algunos detalles añadidos de la compra(Imagen18).

2. Fortaleza de la contraseña

Cuando el usuario se quiere registrar en nuestro sistema en la vista propia para el registro e inicio de sesión, a la hora de completar el campo de su contraseña, el sistema le informa de la fortaleza de su contraseña, mediante un jquery que captura y analiza lo introducido como contraseña por el usuario, dependiendo de la longitud de ésta el mensaje a mostrar en pantalla puede ser: No permitida, Débil, Buena y Muy buena, con unos colores de apoyo donde el rojo es el peor y el verde el mejor(Imagen20).



Imagen 20

3. Número de usuarios conectados

Para la generación del número de usuarios conectados hemos añadido un contador en el footer. El valor se encuentra dentro de un campo de tipo text que no se puede editar. Para que esto funcione y se actualice cada 3 segundos hemos necesitado crear una variable en un script interno en el base.html a la cual le asignamos el valor de url_for de nuestra función python rand(). Una vez creada esta variable, podemos pasarle la variable a la función callrand() de nuestro functions.js desde nuestra llamada con el onload de body. Esta función va a establecer con setInterval cada cuanto tiempo se llama a la función rand de nuestro functions.js que se encarga de hacer la petición al servidor mediante AJAX y pone el numero aleatorio en el contador(Imagen21).

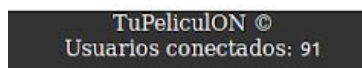


Imagen21

Ficheros que conforman el proyecto

- catalogue
 - Catalogue.json (Catálogo de las imágenes del sistema)
- static
 - Imagenes (Carpeta de todas las imágenes utilizadas por el sistema)
 - misestilos.css (Fichero css que guarda el diseño de los elementos html del sistema)
 - functions.js (Fichero javascript en el que se almacenan las funciones de javascript, ajax y jquery utilizadas en el sistema)
- templates
 - base.html (Fichero html que contiene los elementos comunes en todas las vistas del sistema como por ejemplo la barra de navegación o el pie de página)
 - busqueda.html (Fichero html que muestra la vista de una búsqueda solicitada por el usuario al sistema)
 - carrito.html (Fichero html que muestra la vista del carrito de compra de un usuario dentro del sistema)
 - detalle.html (Fichero html que muestra la vista del detalle de cada una de las películas del sistema)
 - historial.html (Fichero html que muestra la vista del historial de compras realizado por un usuario registrado)
 - index.html (Fichero html que muestra la vista de la página de inicio del sistema)
 - login.html (Fichero html que muestra la vista de inicio de sesión y registro del sistema)
- routes.py (Fichero python en el que se almacena el funcionamiento interno del sistema y cómo interactúa con las vistas)

- users (Carpeta que almacena carpetas individuales por cada usuario del sistema)

Referencias

- Referencia guia para el desplegable:
<https://stackoverflow.com/questions/16751241/how-to-have-slidetoggle-for-multiple-items-on-one-page>
- Referencia guia para el control de la contraseña:
<https://www.formget.com/password-strength-checker-in-jquery/>
- Referencia guia para capturar los valores de la etiqueta select:
<https://stackoverflow.com/questions/32019733/getting-value-from-select-tag-using-flask>
- Referencia guia para el uso de las etiquetas form y de etiquetas dentro de este mismo y los distintos inputs:
<http://www.mclibre.org/consultar/htmlcss/html/html-formularios.html>

Instrucciones de uso:

Para la ejecución solo es necesario seguir las directrices que se nos dan en moodle, no requiere nada adicional.