Enunciado

Para comenzar elaboramos la interfaz HTML y CSS, implementando tablas y listas en páginas como historial.html pero implementando la base.html con divisiones y siguiendo la cabecera, el pie de página… Toda la interfaz principal se muestra en la entrega intermedia.

En base.html se muestra como en todas las páginas principales (aquellas que no son utilizadas como "iframe") utilizamos cabecera, menú lateral y pie de página con nuestros respectivos correos. Además, las rutas son relativas como se exige en el enunciado. También se puede observar el uso de imágenes para representar las películas disponibles.

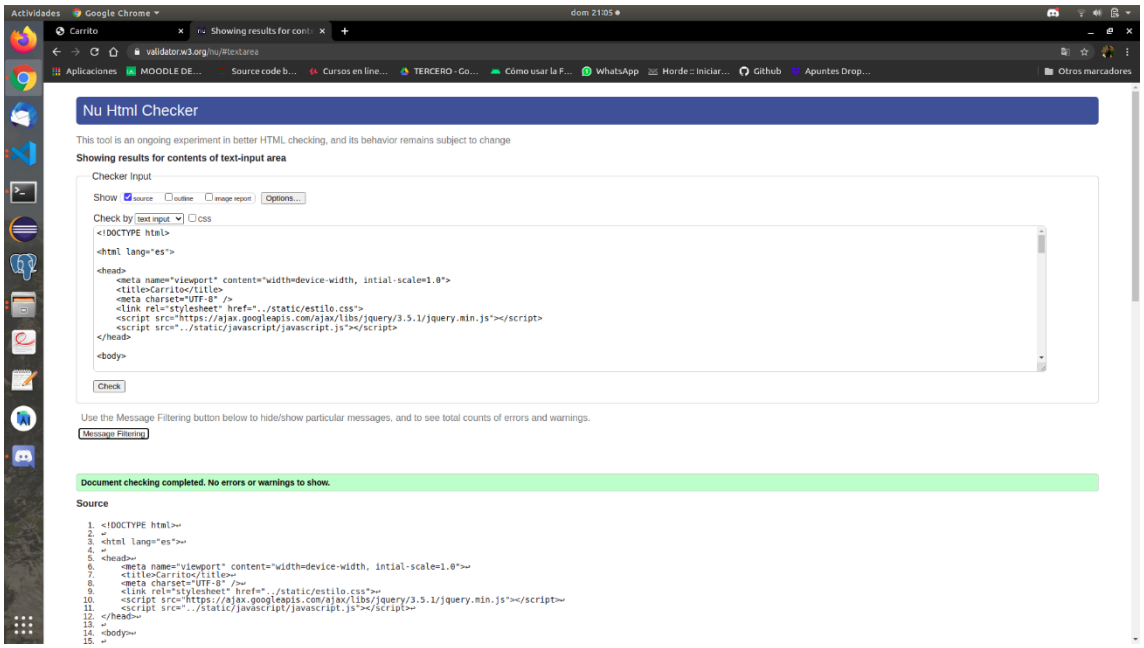
Todo el CSS está en el fichero "estilo.css" el cual busca uniformidad y que se reutilicen las clases modificadas en él lo máximo posible.

Para el correcto funcionamiento de la página, empleamos Python, jquery, HTML, CSS, Jinja2, Flask, Javascript, Json y Ajax. Json por ejemplo es empleado para la creación de un fichero "historial.json" por cada usuario, además del uso de catalogue.json para obtener la información básica de todas las películas.

Para la aplicación usamos una interfaz base denominada base.html en la que predefinimos una distribución de la interfaz como puede ser el "topnav" y el "sidenav" además de la cabecera y el pie de página. Todo esto se consigue empleando Flask.

Para que el código html sea lo más general posible usamos Jinja2, lo cual nos permite mostrar información especificada previamente mediante un bucle o una comparación Flask. Como usamos Jinja2 debemos validar el código generado, lo que quiere decir la comprobación de este en la web, de la cual obtenemos la manera de elaborar nuestro código fuente.

Las siguientes imágenes muestran la validación de un fichero css y uno html como ejemplo



Registro de un Usuario

Para la implementación de una página para Registro de Usuario o en nuestro caso signup.html empleamos tanto javascript como json. Lo primero de todo es la interfaz de la página en la que empleamos “<form>” el cual incluye variedad de inputs, de tipo “text”, de tipo “password”, de tipo “email” y de tipo “tel”. El caso de tipo “tel” se emplea para el número de la tarjeta de crédito. Para calcular la fortaleza de la contraseña usamos javascript en el que cuanto más compleja sea la contraseña más aumenta un “<meter>” que al llegar al máximo pasa a ser verde indicando que la contraseña es lo suficientemente segura. En el caso de que la contraseña no sea lo suficientemente segura con el código javascript nos encargamos de que no guarde los valores recibidos y que no permita al usuario registrarse, mostrando un mensaje en rojo indicando el motivo o los motivos. Además, hay un campo proporcionado para repetir la contraseña, y en el caso de que la contraseña original no coincida con esta, se mostrará un mensaje de error y no permitirá al usuario registrarse. Este proceso por el que no deja al usuario registrarse lo realizamos utilizando ‘onsubmit=”return(validation())”’ siendo validation() la función de javascript que comprueba la validez de la contraseña. Por último como parte de interfaz, tenemos un “<button>” de tipo submit.

Respecto a la creación de las carpetas de usuario por cada usuario registrado, lo conseguimos con el uso de la librería “os” la cual nos proporciona la posibilidad de crear un directorio y que en él se incluya el fichero “historial.json”, , en el que escribimos la información del usuario y “datos.dat”. En el caso de la contraseña, esta es encriptada empleando “hashlib.sha512” como se indica en el pdf.

Detalle de película

La interfaz de Detalle de película, en nuestro caso recibe el nombre de filmDetail.html. Es una de las páginas más visuales. En ella se emplean mucho las imágenes, para mostrar tanto al director de la película como los actores de esta. Imágenes las cuales están almacenadas en “../static/imágenes” es decir en un subdirectorio dentro del directorio static. En ella dependiendo de la imagen de película que haya sido pulsada en cualquier página, aparecerá la información de

esta, la cual se encuentra en catalogue.json. Para mostrar dicha información de forma general, en “routes.py” pasamos como argumento al render_template de filmDetail.html la película que va a ser mostrada y por tanto en el html con el empleo de jinja2, haciendo referencia a los campos de esta imagen podemos representar toda su información. (Un ejemplo sería mostrar la imagen de la película como {{ film.foto }}).

Además, en filmDetail.html proporcionamos la posibilidad de comprar o alquilar dicha película, añadiendo esta automáticamente en el carrito una vez el botón respectivo es pulsado. Tanto la acción de comprar como alquilar se ejecutan de la misma manera sin ofrecer ningún tipo de diferencia de gasto. Esta acción es llamada “añadir_carrito/{{film.id}}”.

Carrito de la compra

Esta página recibe el nombre de carrito.html. En ella se muestra una fila de las películas que han sido añadidas al carrito. Estas pueden ser tanto compradas de forma individual, como eliminadas de la misma forma. Para mostrar las películas se emplea un bucle creado por Flask que permite que se puedan mostrar las películas a comprar de forma uniforme, la aplicación permite que puedas comprar películas que habías comprado previamente. En el caso de querer eliminar las películas de carrito, se emplea la función “eliminar_carrito” haciendo que la película mostrada desaparezca automáticamente. Esto se consigue haciendo un pop de la película del diccionario session. En el caso de la compra de la película esta implica más complicación. Dicha función que permite la compra de películas recibe el nombre de “realizar_compra”. Esta función no siempre se llevará a cabo ya que no siempre el usuario tiene el saldo necesario para pagar la película. Para ello primero se comprobará que dentro de datos.dat del usuario el saldo sea mayor que el coste de la película. Una vez la comprobación está hecha, se añade la película al historial del usuario en historial.json. En el caso de que la compra no sea posible un error es mostrado en pantalla, esto también ocurre si intentas realizar una compra sin haber ingresado con tu cuenta. En el caso de que la compra se realice, al añadirla en historial.json también es eliminada automáticamente del carrito y el saldo del usuario es actualizado. Para la implementación de carrito empleamos memoria que va relacionada con la sesión. Esto quiere decir que con “import session” tenemos una memoria en la que se guardan las películas que quieren ser compradas y que se mantienen con el inicio de sesión del usuario.

Mostrar historial de un usuario

La interfaz de mostrar historial de un usuario se encuentra en el fichero historial.html. Interfaz la cual emplea el uso de tablas para mostrar las compras realizadas y la información básica de estas películas. En la implementación de esta página debemos mostrar los datos almacenados en historial.json. Para ello empleamos Jinja2 además de Flask para crear un bucle que muestre cada película en la tabla de forma uniforme. Además, con el uso de Javascript se proporciona la posibilidad de mostrar más información de la compra como la fecha de esta. En esta página también se le proporciona al usuario el saldo que tiene y la posibilidad de aumentarlo. Para la implementación de aumentar el saldo usamos la función “introducir_saldo()” la cual obtiene el saldo actual del usuario y le aumenta el introducido en forma de input.

Ficheros utilizados

En la carpeta **app** tenemos varios archivos los cuales utilizamos con el fin de hacer funcionar nuestra web.

En el archivo **routes.py**, definimos todas las rutas que se generan al navegar por la página web, podemos ver como para varias rutas hay veces que definimos una única función que renderiza el archivo html o nos redirige a este. Aparte en este archivo se encuentra toda la lógica detrás de las búsquedas, inicios de sesión registros ...

En el archivo **__main__** se define la dirección que se usa para poder usar el servidor.

En el archivo **__init__** se inicializa la sesión de flask.

Ahora podemos ver una carpeta llamada "**thesessions**", en esta se guardan todas las sesiones generadas mientras usamos la web.

En la carpeta **templates** tenemos todos los archivos html que forman las páginas, estos utilizan jinja2 por lo que son "plantillas" que se adaptan al contenido que les pasemos mediante la función render_template.

Si pasamos a comprobar cada uno de los archivos podemos ver lo siguiente:

- **base.html**, este es el archivo html que se usa como plantilla para los demás, aquí definimos la estructura básica de todas y cada una de las páginas.
- **busqueda.html**, category.html, index.html, estas páginas son bastante similares, la primera se usa para mostrar el resultado de nuestras búsquedas, la segunda para mostrar el resultado de nuestras búsquedas por categoría y la última es el menú principal de la web. Las tres tienen una estructura casi idéntica, lo único que cambia es que en la página de búsqueda se le pasan las películas que tiene que mostrar, mientras que en la página de categoría se usa un if en el mismo html para mostrar las películas de cada categoría.
- **historial.html**, en esta página se muestran las últimas compras del usuario en ese momento logeado.
- **login.html y signup.html**, ambos bastante parecidos, uno es para hacer login y el otro para registrarse. En el archivo signup podemos ver que mediante el uso de JQuery definimos la fortaleza de la contraseña.
- **sidenav.html y topnav**, estos son los dos últimos archivos que usamos, son archivos que definen como deben lucir la barra superior y lateral izquierda.

Pasamos ahora a la carpeta **static**, ahí tenemos todos los elementos estáticos de nuestra implementación.

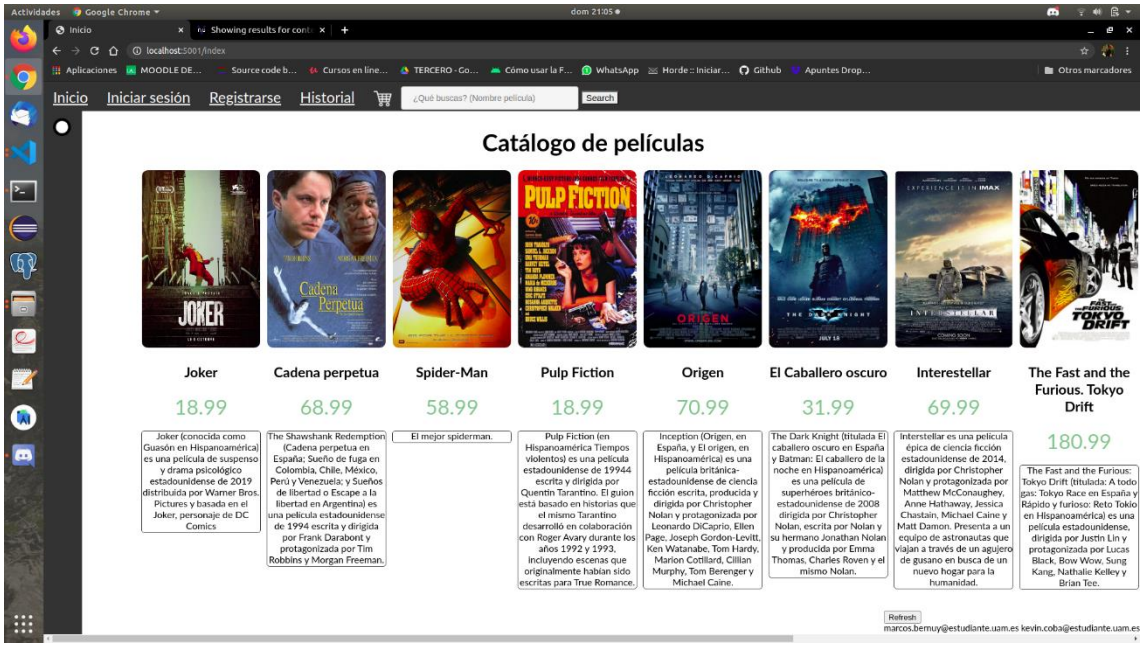
Tenemos la carpeta imágenes que esta subdividida en utils, imágenes como el carrito..., películas, posters de películas, y actores.

Luego también tenemos la carpeta javascript que únicamente contiene el fichero javascript.js, donde por ejemplo comprobamos que los campos introducidos para registrarse cumplan los requisitos establecidos, en caso de no ser así no dejaríamos que se enviase una petición al servidor.

Por último, en esta carpeta tenemos estilo.css, que es el lugar donde definimos todos los estilos de la página web.

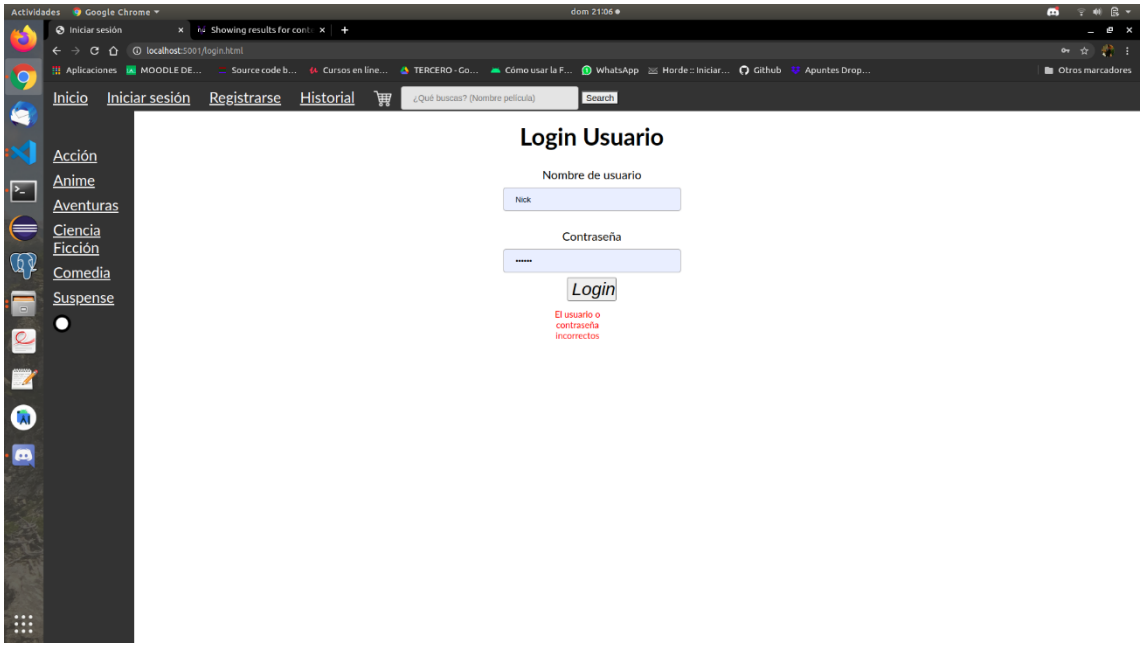
Para acabar tenemos el directorio catalogue, que contiene el fichero catalogue.json el cual tiene todas las películas de la página.

Tour por la Web



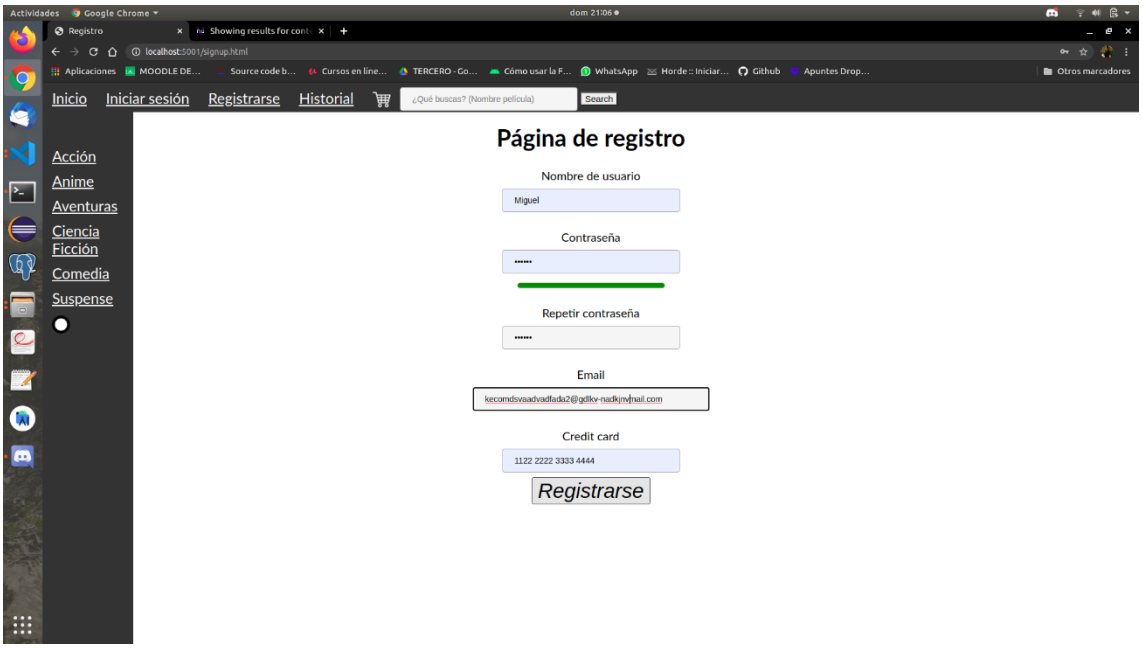
Como podemos ver en esta primera imagen, este es el menú principal de nuestra página. Desde aquí podemos buscar películas, filtrar películas por categorías, iniciar sesión…

Para empezar, comenzaremos iniciando sesión, nos moveremos a la barra superior y pulsaremos iniciar sesión.

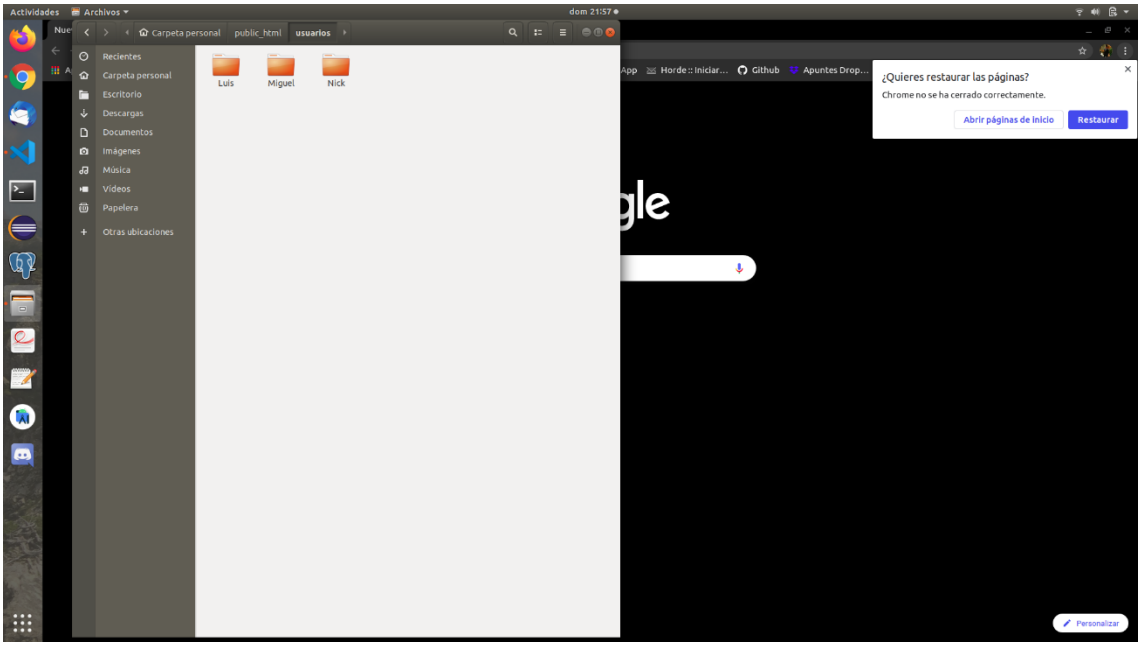


Pondremos nuestros datos y nos daremos cuenta de que esta cuenta no existe.

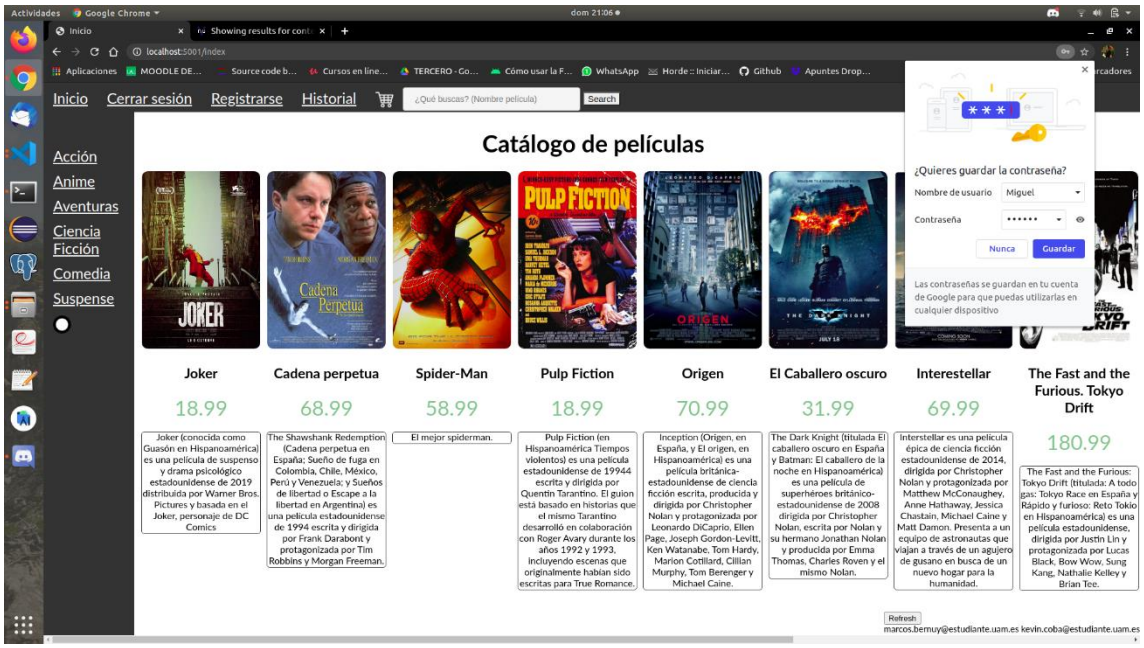
Ahora nos iremos a *registrarse* y desde ahí introduciremos todos los datos necesarios para registrarnos.



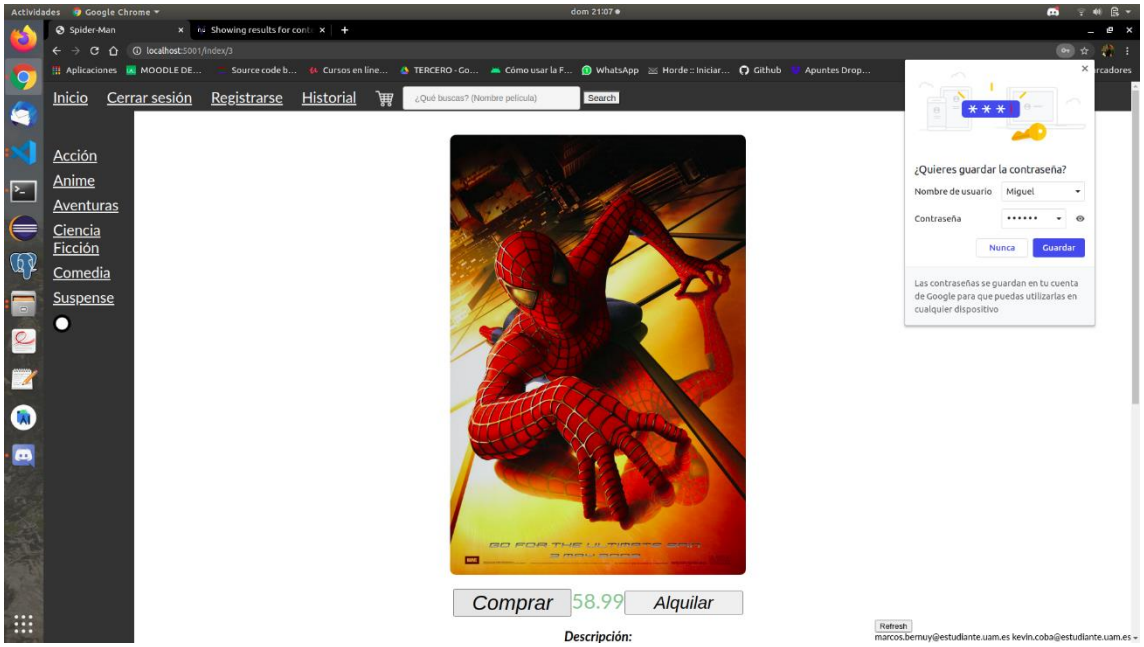
Una vez nos hemos registrado podemos ver que en nuestro ordenador que hace de servidor, se crea un directorio en la carpeta usuarios con el nombre de nuestro usuario.



Proseguimos con nuestro tutorial y ahora podemos ver que donde antes se iniciaba sesión ahora se cierra. El mensaje mostrado ahora es *Cerrar sesión.*

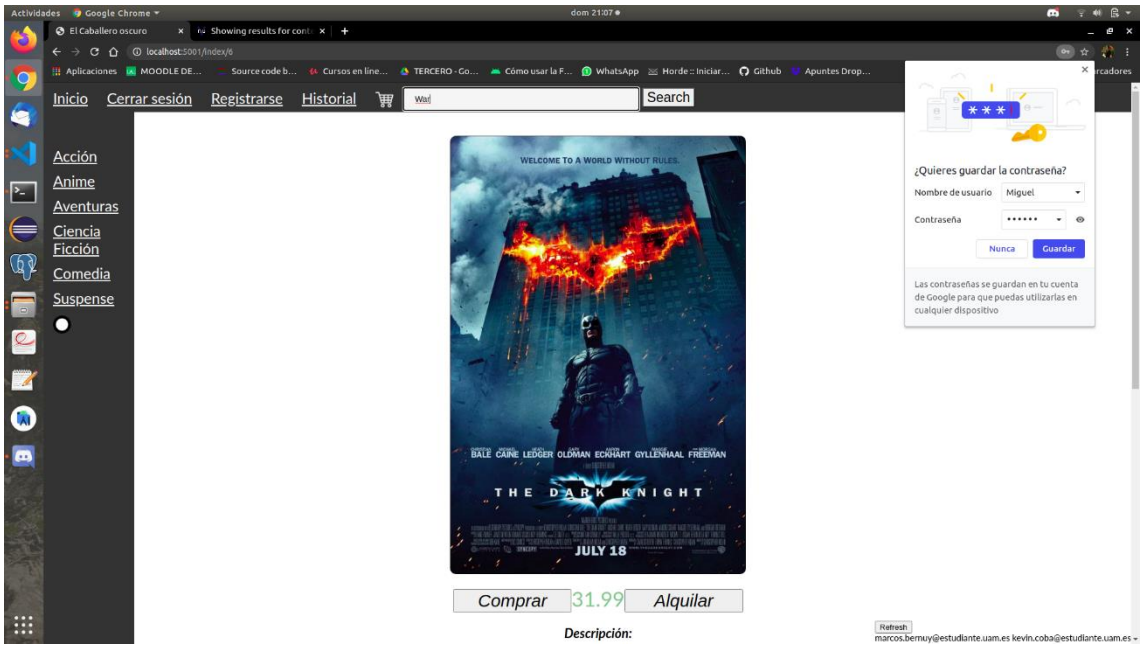


Pasaremos ahora a alquilar una película, para ello clicaremos en la imagen de esta y se nos redirigirá a la página que muestra en detalle la película. Para añadirla al carrito pulsaremos el botón alquilar o comprar.

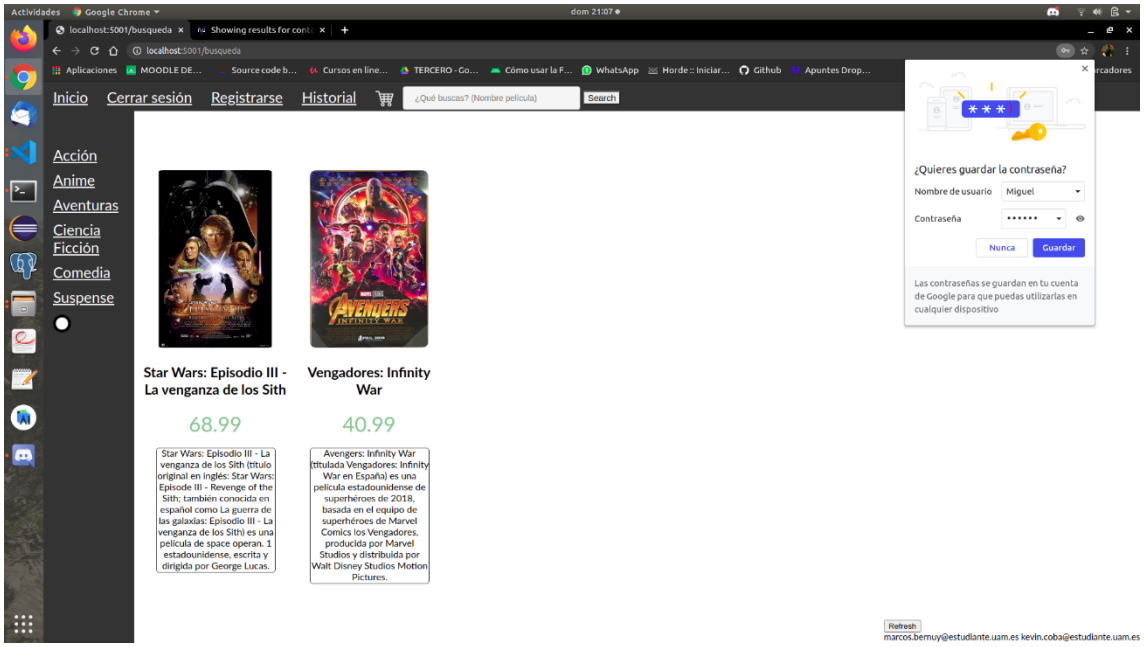


Si quisiéramos añadir otras películas iríamos a inicio y repetiríamos el proceso, pero en otra película.

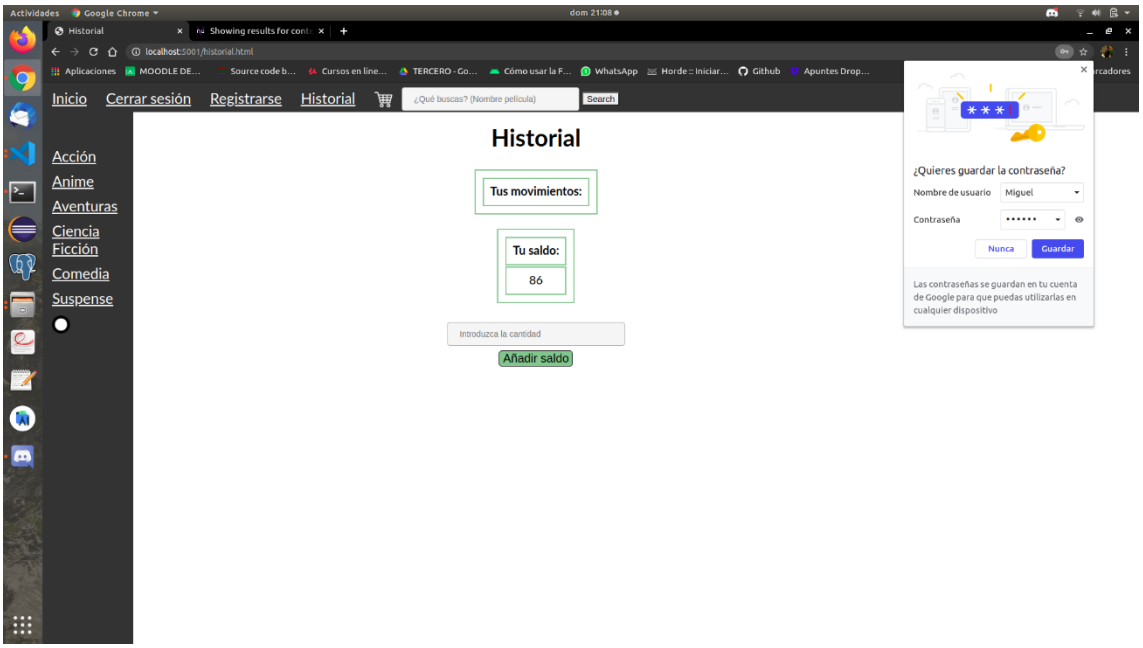
También podemos usar el buscador, para eso introduciremos parte del nombre de la película en campo de búsqueda y después pulsaremos el botón *Search*.



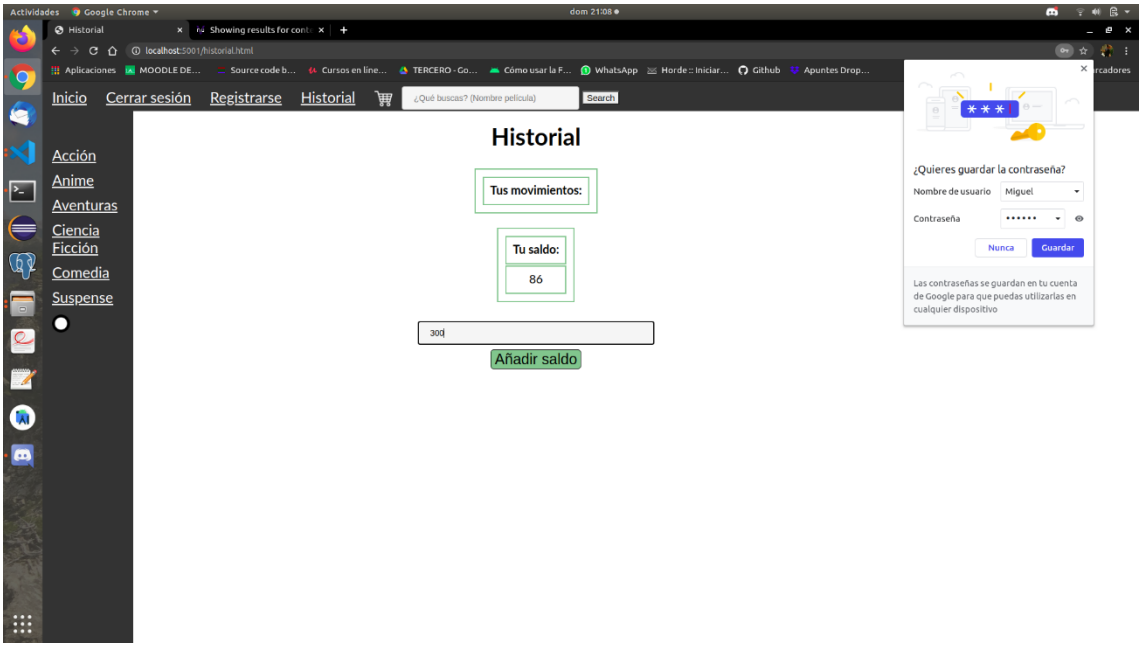
Podemos apreciar que el buscador nos muestra las películas que contienen la cadena que hemos introducido.

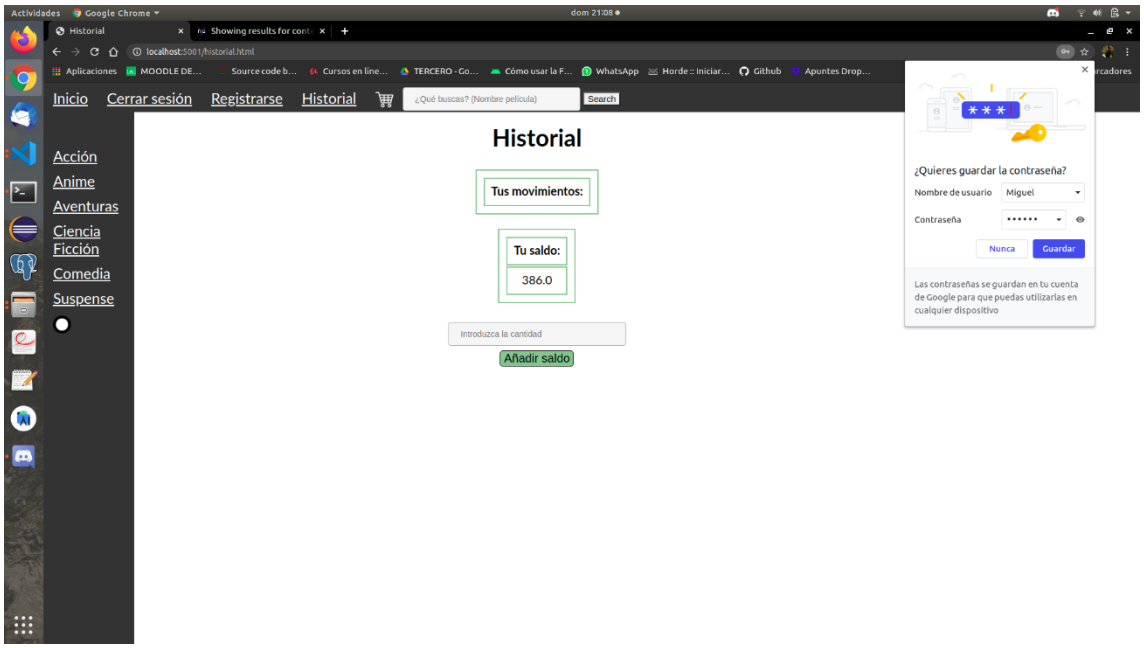


Antes de comprar la película podemos consultar nuestro saldo, para eso iremos a la página historial.

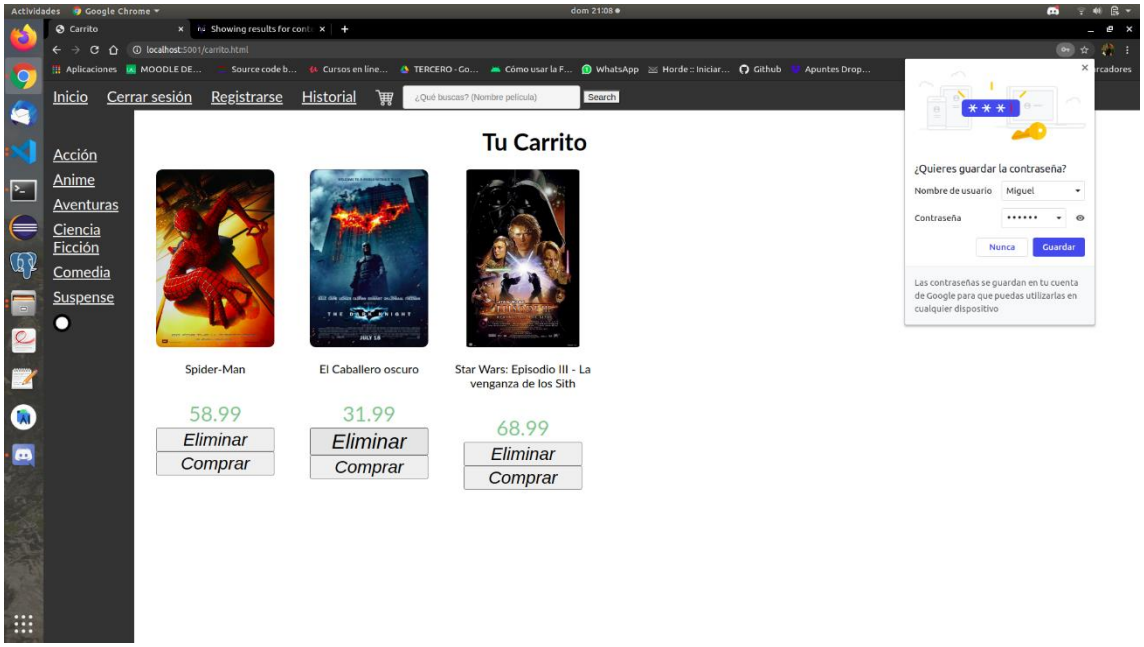


En historial también podemos añadir saldo a nuestra cuenta. Esto básicamente lo que hace es modificar el archivo datos.dat.

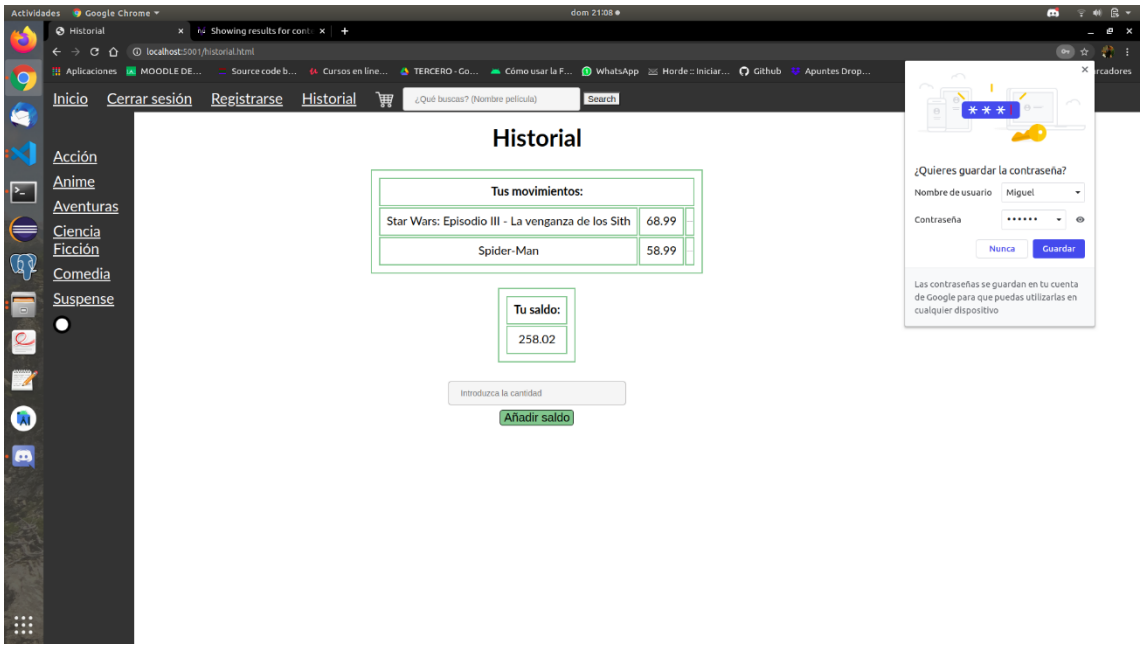


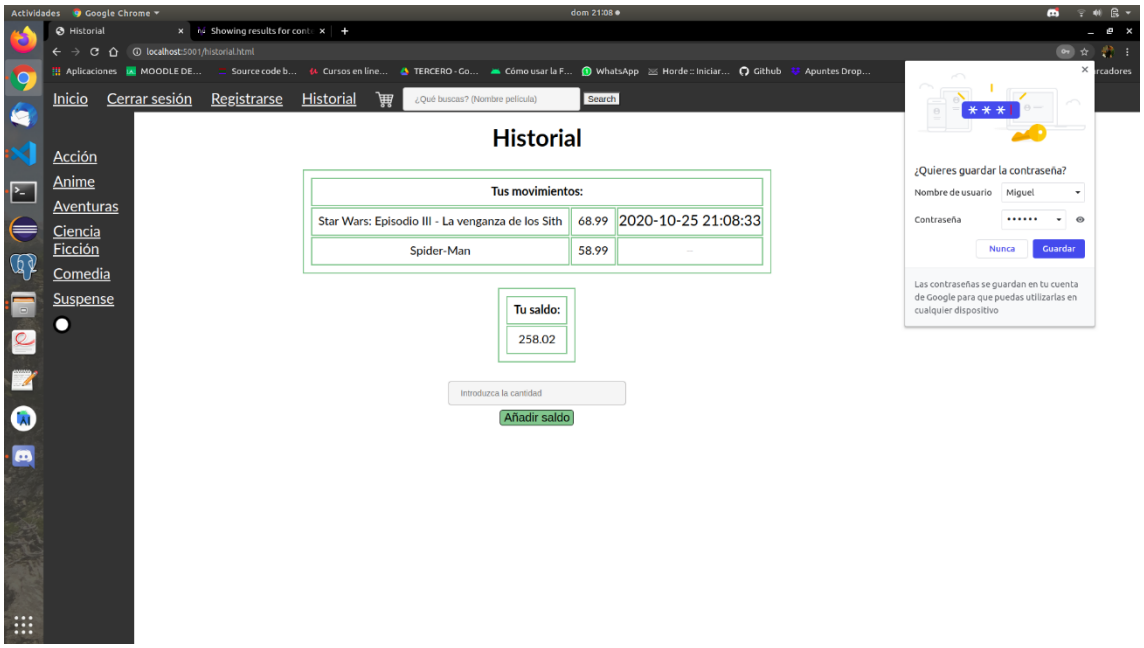


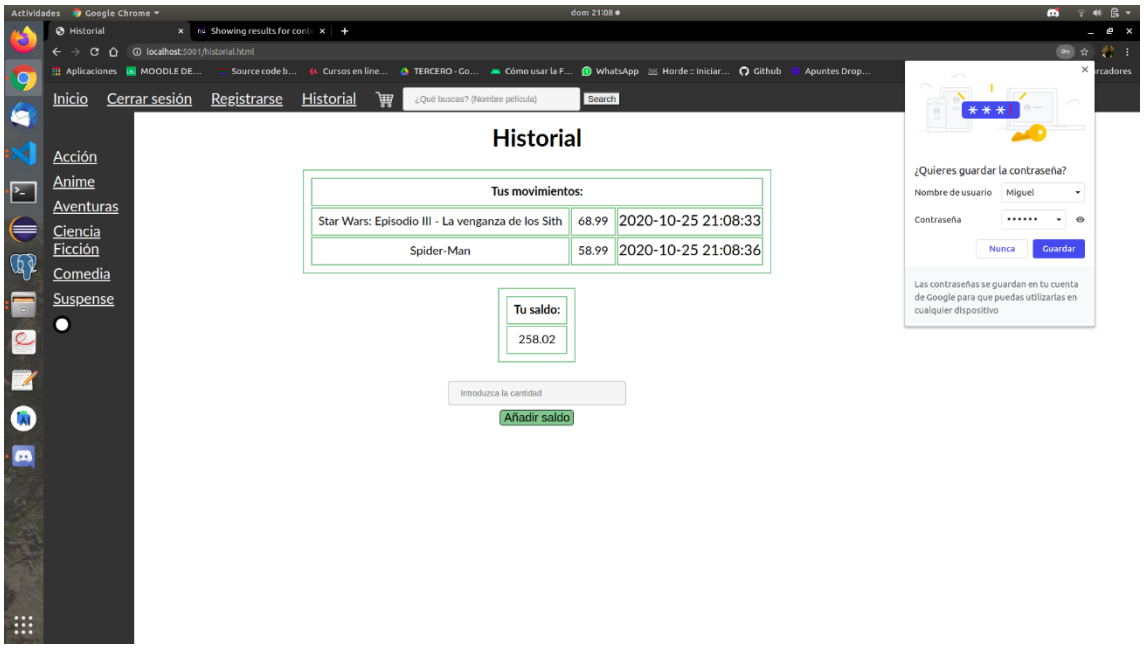
Ahora nos moveremos a carrito, desde ahí podemos comprar las películas ya añadidas.



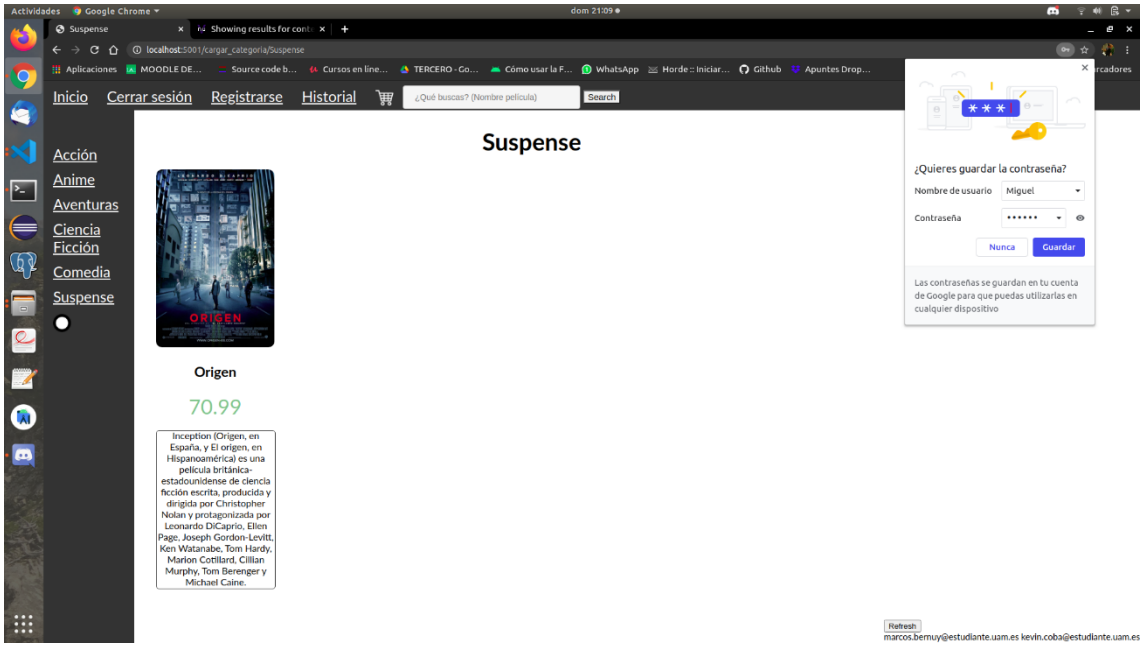
Para consultar las películas que hemos comprado iremos a la página historial, desde aquí podemos ver nuestras compras y podemos ver el detalle del momento en el que compramos dicha película si pulsamos en la tercera celda de la fila.


Inicio
Cerrar sesión
Registrarse
Historial
Search
Acción
Anime
Aventuras
Ciencia Ficción
Comedia
Suspense
Historial
Tus movimientos:
Star Wars: Episodio III - La venganza de los Sith
68.99
Spider-Man
58.99
Tu saldo:
258.02
Introduzca la cantidad
Añadir saldo
¿Quieres guardar la contraseña?
Nombre de usuario
Miguel
Contraseña
Nunca
Guardar
Las contraseñas se guardan en tu cuenta de Google para que puedas utilizarlas en cualquier dispositivo


Inicio
Cerrar sesión
Registrarse
Historial
Search
Acción
Anime
Aventuras
Ciencia Ficción
Comedia
Suspense
Historial
Tus movimientos:
Star Wars: Episodio III - La venganza de los Sith
68.99
2020-10-25 21:08:33
Spider-Man
58.99
Tu saldo:
258.02
Introduzca la cantidad
Añadir saldo
¿Quieres guardar la contraseña?
Nombre de usuario
Miguel
Contraseña
Nunca
Guardar
Las contraseñas se guardan en tu cuenta de Google para que puedas utilizarlas en cualquier dispositivo



Podemos usar también la barra lateral para buscar películas por categoría.



Guía de uso

Para poder ejecutar nuestra página desde un servidor flask, nosotros lo que hicimos fue lo siguiente. Lo primero crear un entorno virtual, nosotros decidimos hacerlo en la carpeta public_html, por lo tanto, desde ahí abrimos una terminal y escribimos el siguiente comando:

```
source si1pyenv/bin/actívate
```

Al ejecutar este comando entramos en el entorno virtual, y desde aquí ejecutamos el siguiente comando:

```
python3 -m app
```

ejecutar Con esto podremos navegar por nuestra web si introducimos en el navegador la siguiente dirección:

http://localhost:5001/