

Práctica 2

Programación Web Dinámica

Asignatura: Sistemas Informáticos 1

Profesor: Antonio Garmendia

Realizado por: Vanessa Salazar

Nicolás Trejo

Curso: 2018 / 2019

Tabla de contenido

[Ficheros que componen la entrega 3](#_Toc528754744)

[Instrucciones de uso 4](#_Toc528754745)

[Preparación del entorno 4](#_Toc528754746)

[Ejecución de la aplicación 4](#_Toc528754747)

[Funcionamiento de la app 6](#_Toc528754748)

[Navegación – frontend 6](#_Toc528754749)

[Lógica - backend 7](#_Toc528754750)

[Bibliografía 8](#_Toc528754751)

[Anexo 9](#_Toc528754752)

# Ficheros que componen la entrega

hello.wsgi

app

* \_\_init\_\_.py
* \_\_main\_\_.py
* routes.py
* catalogue
  + catalogue.json
* static
  + estilo.css
  + función.js
* templates
  + base.html
  + carrito.html
  + categorías.html
  + datousuario.html
  + historial.html
  + index.html
  + login.html
  + pago
  + película.html
  + sigin.html
* thesessions
  + Incluirá las sesiones de la aplicación conforme el uso de la misma
* usuarios
  + Contendrá ficheros con el nombre {{user}}
    - data.dat
    - historial.json

# Instrucciones de uso

## Preparación del entorno

Después de haber instalado los paquetes necesarios para el desarrollo del proyecto tales como son: apache2 , python versión 2, flask, virtualenv, etc. Se requiere crear un entorno virtual de Python con Flask, el mismo que nos permitirá probar las páginas desarrolladas, esto se realizar mediante los mandatos descritos a continuación, “si1pyenv” será el nombre que daremos al entorno virtual a emplear.

$ virtualenv -p python2 si1pyenv

$ source si1pyenv/bin/activate

$ pip install Flask SQLAlchemy Flask-SQLAlchemy SQLAlchemy-Utils \ psycopg2 'itsdangerous

Una vez creado el entorno virtual se creará un directorio denominado “public\_html” que deberá estar publicacio en la ruta http://localhost/~usuario/, ubicación donde se alojarán los ficheros del proyecto, para esto es necesario ejecutar los siguientes mandatos dentro de $HOME (home/usuario) :

$ mkdir ~/public\_html

$ sudo a2enmod userdir

$ sudo service apache2 restart

Después de creado el directorio, se deberá configurar el módulo “mod\_wsgi”, cuya función es ser un nexo entre el servidor web(Apache) y la aplicación web desarrollada(python), que a través de un fichero de extensión .wsgi que se ejecutará al inicio y permitirá acceder a la aplicación. Esto ser realiza con los siguientes pasos:

* Se ejecuta en el terminal

$ sudo a2enmod wsgi

* Posteriormente se abre y edita el archivo “userdir.conf”, ubicado en la ruta /etc/apache2/mods-enabled/userdir.conf, en donde se debe añadir ExecCGI a las opciones y wsgi-script .wsgi como handler.

Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec ExecCGI AddHandler wsgi-script .wsgi

## Ejecución de la aplicación

El acceso a los ficheros de la aplicación puede realizarse de dos formas:

1. Accediendo a través del browser, mediante la ruta:

http://localhost/~*usuario*/*hello.*wsgi

El archivo ”hello.wsgi” activa el entorno virtual e importa los archivos que se encuentran en la carpeta app, mediante la sentencia:

**from** app **import** app **as** application

1. Empleando el terminal, activando el entorno virtual y ejecutando la aplicación mediante los mandatos descritos a continuación, los mismos que al ser ejecutados activan el modo debug y permiten acceder a la aplicación desde el browser a través de la ruta http://127.0.0.1:5000/

cd~/public\_html

source si1pyenv/bin/activate

PYTHONPATH=. python -m app

# Funcionamiento de la app

## Navegación – frontend

Al iniciar el debug, se muestra el template “index.html, en el caso de esta aplicación , esta pagina se considera como pagina de inicio.

**Index:** muestra en su contenido el catálogo de películas del fichero .json, cada una con su respectivo poster y título.

Desde esta pagina es posible acceder a los templates de : detalle de películas, Sign in, Log in, Log out, Carrito e Historial

**Detalle de Peliculas:** Al dar click sobre cualquiera de las peliculas mostradas en el index, se accede al detalle de la misma , en donde se puede ver la portada, titulo, actores, sinopsis, formato y precio de la misma. Además desde esta página será posible añadir la pelicula al carrito de compras, para la posterior gestión de compra de ser deseado.

**Sign in :** Esta opcion nos permite crear un nuevo usuario en la aplicación , el mismo que debera ingresar un username, nombre, apellido, correo electronico y numero de tarjeta, la información ingresada deberá cumplir con ciertas restricciones para ser valida asi como:

*Username:* no debera tener espacios y debera tener menos de 10 caracteres, además de verificará que no exista otro usuario igual, de ser así se mostrara un mensaje que lo indique .

*Nombre:* deberán ser solo letras y no deberá tener mas de 12 caracteres

*Apellido:* deberán ser solo letras y no deberá tener mas de 12 caracteres

*Contraseña:* deberá tener 8 caracteres como minimo, esta se comparará con la confirmacion de contraseña para corroborar que ambas sean iguales.

*Correo:* debera tener formato correcto xxxx@mail.com

*Tarjeta:* deberá tener 10 caracteres.

**Log In:** Una vez creado el usuario, este podrá acceder al sistema con el numbre de usuario y contraseña, se verificarán en los datos ingresados en el registro corroborando que la contraseña ingresada sea igual a la contraseña que fue almacenada por la aplicación.

**Historial:** La opción de historial aparecerá una vez que el usuario haya ingresado al sistema con su nombre de usuario y contraseña, en esta página podrá acceder a la informacion de las peliculas que ha comprado anteriormente, además contendra una opción que permitirá visualizar los datos del usuario.

**Datos de Usuario:** Mediante esta pagina el usuario podrá visualizar los datos que ingreso al momento del registro además de poder conocer cual es el saldo restante que le queda para realizar compras.

**Carrito:** El carrito de compras permite almacenar temporalmente las peliculas que el usuario a añadido al mismo. Para gestionar la compra de los items que se encuentran en el carrito es necesario que el usuario ingrese al sistema, con un usuario y contraseña previamente creados.

**Pago**: Dentro de esta página el usuario podra terminar el proceso de compra, sumando los valores de las peliculas que se encuentran en el carrito y restandolas del saldo que este posee. Se mostrará un mensaje de compra exitosa o de saldo insuficiente dependiendo de cual sea el caso.

## Lógica - backend

Las paginas de la aplicación incluyen el template “base.html”, este es usado con el fin de hacer uniforme la presentacion del header, footer y menú lateral en todas las páginas de la aplicación. En este template se incluye las rutas de enlace de todos los estilos de css y funciones de js que manejara la aplicación.

Las categorias que se muestran el el menu lateral , se generan de forma automatica mediante una busqueda de los generos disponibles en el json de la aplicación y la publicación de los mismos, estos variaran en función del catalogo de JSON.

Adicionalmente en el menu lateral del template base, se encuentran las opciones de acceso a varias paginas del sistema, las mismas que estan disponibles de acuerdo al estado de la session que maneje en ese momento la aplicación. Si “user” no se encuentra “in session”, estaran disponibles las opciones de carrito, Sig in y Log in, y cuando “user” se encuentre “in session”, se harán visibles las opciones de historial y Log in y desaparecerán las dos primeras, además se mostrará un mensaje de bienvenida con el nombre del usuario que se encuentre ese momento “in session”.

**Index:** El index esta programado para realizar una busqueda en el catalogo de JSON todas laspeliculas disponibles y las muestra con su respectivo titulo y poster.

**Detalle de peliculas:** El detalle de peliculas a traves de una funcion identifica la pelicual que ha seleccionado el usuario y muestra los datos almacenados en el catalogo de JSON correspondientes a esa pelicula, ademas a través del boton añadir al carrito podemos agregar esta pelicula a la lista temporal que el carrito de compras almacena.

**Sign in :** Se podrá crear un nuevo registro siempre y cuando no haya ninguna sesion iniciada,es necesario ingresar al formulario los datos solicitados, los mismos que serán validados por la funcion validar.js.

Cuando los datos cumplan con el formato que se solicita y al hacer submit , se comprobará a traves de “signin\_activate”, que el Nombre de usuario ingresado no exista, esto se realizará buscando en la carpeta “usuarios”, un fichero con el nombre ingresado, de existir una carpeta con ese nombre se muestra un mensaje indicando que el usuario ya existe, caso contrario se crea una carpeta con el nombre del usuario ingresado.

La contraseña se encriptara con md5, para esto es necesario incluir en routes el uso de la librería mediante: from hashlib import md5,y llamando a la funcion “createuserlogin” que además de realizar la encriptación de contraseña, almacenará los datos ingresados en un archivo “datos.dat” , añadiendo una categoría adicional llamada saldo, a la que se le asignara un valor random entre 0 y 100.

Cuando esto se realice, el usuario hace log in automático a la aplicación con el nombre de usuario y contraseña ingresados, iniciando a “user” is “in session”.

**Log in:** Dentro de esta página el usuario debe ingresar el nombre de usuario y contraseña que ha registrado en el sistema, a través de “login\_activate”, se buscan dentro del archivo “datos.dat” la contraseña almacenada en el momento del registro y mediante la función ”check\_password” se comprueba si es la misma que ha ingresado el usuario, de no ser asi muestra un mensaje de error, y si es la misma contraseña se inicia la session “user” , con todos los datos del usuario para su uso posterior.

**Log out:** Esta opción se encuentra disponible únicamente cuando existe una sesión iniciada, a través de la sentencia “session.clear()” elimina toda la información de la sesión del “user” actual , y permite que otro usuario pueda iniciar una nueva sesión

**Historial:** El historial permite leer dentro de la ruta: “usuarios/{{user}}/historial.json” la informacion sobre las peliculas que el cliente ha comprado anteriormente , asi como permite escribir en este las nuevas peliculas que el usuario ha adquirido.

**Datos de usuario:** Con la sesion activa, la pagina datos de usuario muestra la informacion del usuario que fue ingresada en el momento del registro , extrayendola del la ruta “usuarios/{{user}}/datos.dat”

**Carrito:** Al añadir una película al carrito, se almacena en una lista temporal dentro de la sesión actual, en donde se incluyen las películas seleccionadas por el usuario, se extrae del catalogo de JSON información sobre esta, como es el titulo de la película y el precio. Para poder continuar con la compra es necesario que el usuario haya iniciado sesión, solo de esta forma podrá ingresar a la opción de pago.

**Pago:** El la opcion pago se realiza la suma de precios de las peliculas que se encuentran en el carrito , y este valor se resta de la variable saldo almacenada el el archivo “datos.dat”, si el valor de saldo es mayor al valor de la suma , se efectua la compra y se eliminan los items del carrito, de no ser asi se muestra un mensaje de fondos insuficientes.

# Bibliografía

**WEB:**

Miguel Grinberg (Octubre 2018). *The Flask Mega-Tutorial Part I: Hello World! - miguelgrinberg.com*. Recuperado de https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

Tutorials Point (Octubre 2018). *Flask Tutorial*. Recuperado de https://www.tutorialspoint.com/flask

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O’Reilly Media, Inc.

Ronacher, A. (2010, 2018). Flask (A Python Microframework). Retrieved October 31, 2018,

from http://flask.pocoo.org/

Maia, I. (2015). *Building Web Applications with Flask*. Packt Publishing Ltd.

# Anexo

Si hay cosas largar para agregar (muchas imágenes o código), que vaya en este Anexo

[Texto]