Memoria Práctica 2:

Desarrollo del backend en Flask de la aplicación presentada en la práctica 1.

Óscar Gómez Borzdynski y Lucía Colmenarejo Pérez

Introducción:

Durante esta práctica hemos desarrollado el backend de la aplicación realizada en la práctica 1: La casa del DVD. Para ello hemos utilizado Flask, un framework de python que crea un servidor que permite el uso de Jinja2 para templates. Haciendo que el código HTML se vea muy reducido.

El servidor utiliza un entorno virtual llamado si1pyenv con los siguientes paquetes instalados:

- Click==7.0
- Flask==1.0.2
- Flask-Session==0.3.1
- Flask-SQLAlchemy==2.3.2
- itsdangerous==0.24
- Jinja2==2.10
- MarkupSafe==1.0
- psycopg2==2.7.5
- six==1.11.0
- SQLAlchemy==1.2.12
- SQLAlchemy-Utils==0.33.5
- Werkzeug==0.14.1

Servidor:

Para que apache pueda utilizar este servidor, necesitamos un modulo para wsgi, permitiéndole ejecutar python. Por ello creamos un fichero index.wsgi que hará referencia a index.py, fichero donde se almacena toda la lógica de nuestro servidor.

```
#!/usr/bin/python
import os
import sys

# virtualenv
this_dir = os.path.dirname(os.path.abspath(__file__))
activate_this = this_dir + '/silpyenv/bin/activate_this.py'
execfile(activate_this, dict(__file__=activate_this))

# anhadir dir de este fichero a path
sys.path.insert(0, this_dir)
from index import app as application

```

Dicho index.py está definido de la siguiente manera:

Una sección de funciones auxiliares y variables globales
Destacan:

- La carga de películas y categorías como variables globales.
- Funciones auxilares de normalización de strings
- Gestión de los ficheros de usuario y comprobación de passwords
- Gestión del historial.

```
#Datos del catalogo (Data es un diccionario)
Data = open(os.path.dirname(__file__)+"/catalogo.json", "r").read()
Data = json.loads(Data)

#Conseguir todas las categorias, sin repetidos
aux = list(set([a["categoria"] for a in Data["peliculas"]]))
Categorias = []
for a in aux:
    auxdic = {"nombre": a, "imagen": None}
    for film in Data["peliculas"]:
        if (film["categoria"] == a):
            auxdic["imagen"] = film["poster"]
            break
    Categorias.append(auxdic)

#Ordenar las peliculas por año (las mas nuevas primero)
Novedades = sorted(Data["peliculas"], key=lambda x: -int(x["anno"]))

#Obtener las peliculas ordenadas por puntuacion de los criticos
def sort_rec(x):
    aux = 0.0
    for a in x["opiniones"]:
        aux += float(a["puntuacion"])
    return -aux/len(x["opiniones"])
Recomendadas = sorted(Data["peliculas"], key=lambda x: sort_rec(x))
```

Una sección con las funciones específicas de Flask, que comienzan con el wrapper @app.route() Cada una de ellas generará el HTML para mostrar la página correspondiente mediante el método render_template.

Nótese que algunas funciones realizan comprobaciones de acceso, es decir, evita que ciertas páginas se rendericen si el usuario que accede a las mismas no tiene los permisos correpondientes.

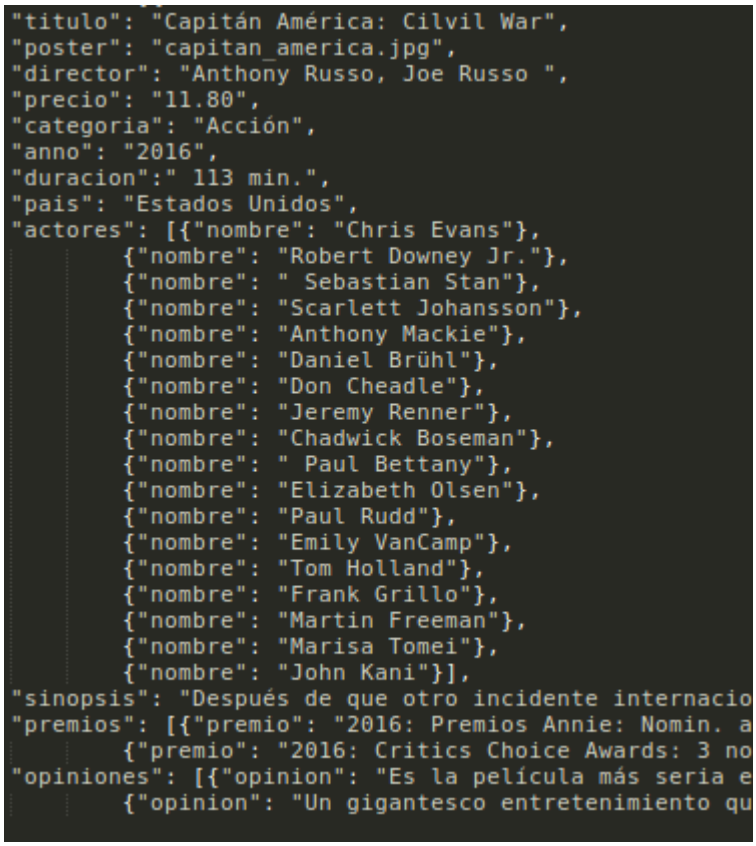
Uno de estos casos es la función que muestra los datos de usuario. Una persona que no está logueada no podrá acceder a ninguna información de usuario, aunque conozca el nombre del mismo. Un usuario logueado, sólo podrá acceder a sus datos.

```
@app.route("/users/<userc>/")
def user_info(userc):
    if "user" in session:
        user = session["user"]
        email = session["email"]
        saldo = session["saldo"]
        historial = sorted(get_historial(), reverse=True, key=lambda x: time.strptime(x["fecha"
        if userc != user:
            return index()
    else:
        return index()
    if "ncompra" in session:
        ncompra = session["ncompra"]
    else:
        ncompra = 0
    return render_template("user-info.html",\
     novedades_sidebar=Novedades[:4], populares_sidebar=Recomendadas[:4], ncompra=ncompra,\
     user=user, email=email, saldo=saldo, historial=historial)
```

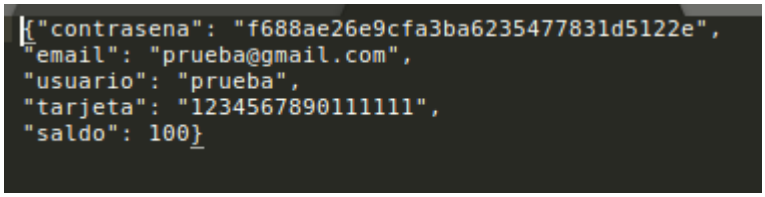
Para usar nuestro servidor. Es necesario disponer de apache2 instalado con el mod para wsgi. El proyecto deberá trasladarse a la carpeta correspondiente en el sistema donde se desee utilizar.

Datos:

Estos se guardarán en formato json, el catálogo de películas es catálogo.json e incluye los datos necesarios para representar cada película.



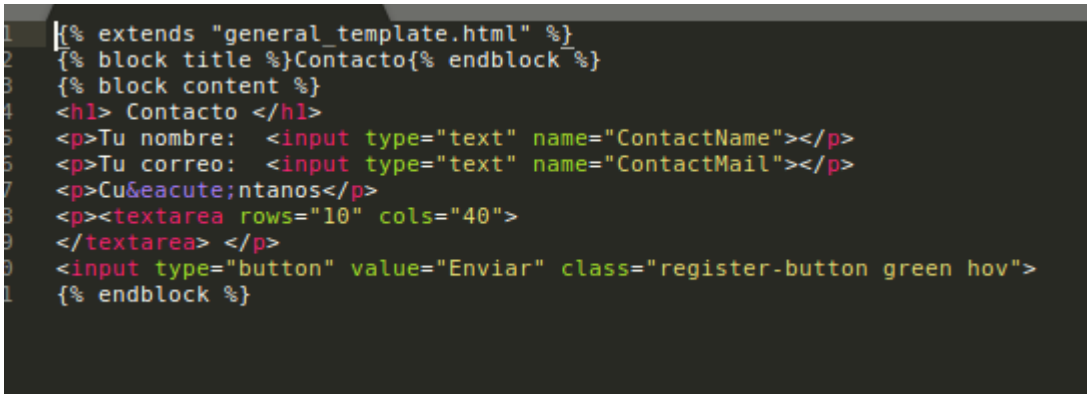
Los datos de usuario se almacenan en un fichero datos.dat (también en formato json) y su historial en historial.json.
Para el acceso a los datos utilizamos el módulo json de python y sus funciones json.loads() y json.dumps()



Templates:

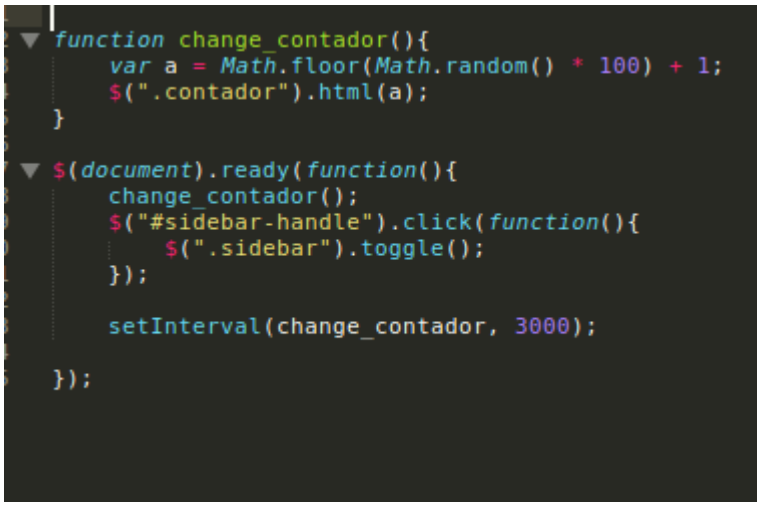
Los templates se almacenan en la carpeta templates. Utilizamos la sintaxis de jinja2, que permite ejecutar código con una sintaxis muy similar a python al renderizarlo.

Todos los templates heredan de un template llamado generla_template.html, donde se define la estructura genérica de la página.



Ficheros estáticos:

En cuanto a los ficheros estáticos, se almacenan en la carpeta static. Están divididos en imágenes, hojas de estilo y ficheros javacript. Dichos scripts están implementados con Jquery y Javascript clásico.



```
function change_contador(){
    var a = Math.floor(Math.random() * 100) + 1;
    $(".contador").html(a);
}

$(document).ready(function(){
    change_contador();
    $("#sidebar-handle").click(function(){
        $(".sidebar").toggle();
    });

    setInterval(change_contador, 3000);

});
```

Conclusiones:

En esta prácita hemos empezado a utilizar nuevas tecnologías como el framework Flask. Además hemos introducido lógica back-end y scripts de cliente.
Sin embargo hay una serie de aspectos que no nos han terminado de agradar. El principal es que hemos tenido que descartar la mayoría del trabajo realizado en la práctica anterior. Creemos que el esfuerzo realizado en la dicha práctica apenas se ha visto recompensado en el desarrollo de la práctica actual.

Referencias:

http://flask.pocoo.org/

https://www.tutorialspoint.com/flask/

https://www.python.org/

https://css-tricks.com/

https://jquery.com/

https://www.javascript.com/

https://www.w3schools.com/

https://es.stackoverflow.com/

https://www.google.ca/

https://www.desarrolloweb.com/

https://httpd.apache.org/