MUSICSHOP

Web Musical

PROGRAMACIÓN WEB

GRADO EN INGENIERÍA INFORMÁTICA.

Mario Ortega Aguayo

76439773D

1. Navegación por la web.

MusicShop es una aplicación web creada en Django en la cual podremos ver las últimas novedades musicales de varios estilos.

Al acceder a la web, vemos la sección principal, donde encontramos los discos principales, los más comentados, los últimos discos por género, discos más vendidos y hasta un apartado de noticias. En la parte superior encontraremos un pequeño formulario de login.

A través de la navegación por el menú, podremos acceder a dichos estilos, y encontraremos el contenido de los discos. Al acceder a un disco, si el usuario está registrado en la base de datos, podemos realizar comentarios sobre cada uno de ellos y este comentario se verá reflejado en la parte inferior de este apartado. La sesión de usuario se mantendrá abierta hasta que el propio usuario pulse el botón Logout.

Para registrarnos en la web, una vez iniciada, podremos hacerlo pulsando en REGISTRATE, donde insertaremos nuestros datos personales, y tras esto tendremos acceso automáticamente al contenido privilegiado, en este caso podremos comentar discos. Este formulario tendrá una pequeña validación de campos, por lo cual tendremos que rellenarlos correctamente según indica el formato en el interior del campo.

El usuario admin (con contraseña admin), aparte de realizar comentarios en modo moderador, podrá registrar discos de cualquier estilo en la web (insertando como género: rock, metal, punk o electronica). Para ello, una vez logueado, pulsaremos en el botón Administrar, en la parte superior de la web.

Vamos a profundizar a continuación en como se ha desarrollado la web.

2. Django.

Django es el framework utilizado para el desarrollo de este sitio web. Django no solo dispone de todas las ventajas de un framework, como funcionalidades ya creadas a las cuales accederemos a través de su librería. Brinda muchas más ventajas, como el diseño propio de formularios, la utilización de Python, un lenguaje actual que da muchas facilidades para su desarrollo), la estructuración en Modelo-Vista-Controlador o su potente sistema de plantillas.

Todos estos aspectos han sido utilizados y aprovechados para el desarrollo de esta web. Vamos a explicar el contenido de cada uno de los ficheros propios de este framework.

- El archivo views.py:

Es el archivo que contiene el código Python que da las funcionalidades a cada uno de los elementos con los que interactúa el usuario, como por ejemplo login, logout o comentarios. También incluye funciones para el administrador, por ejemplo la inserción de discos.

- El archivo urls.py

Es el archivo que se encarga de gestionar el direccionamiento de cada uno de los enlaces de la web. A través de su formato específico, asignaremos una funcionalidad, en este caso con el nombre de la función que le corresponde en el views.py, a cada uno de los enlaces por los que vamos a navegar por la web. El formato para asociar la url con la función es el siguiente:

```
urlpatterns = [
    url(r'^login/',views.login,name='login'),
]
```

Donde especificamos que cuando direccionemos a /login va a llamar a la función login del fichero views, con nombre de url login. Este nombre de url lo llamaremos desde la plantilla html con este formato: {% url ‘función_views’ %}.

3. <u>El sistema de plantillas.</u>

En primer lugar, tenemos que crear un base.html, el cual va a tener el contenido común a toda la web. En mi caso, he tomado como común la cabecera de la web, que contiene el titulo, logo y formulario de login, este último variando el diseño en función del tipo de usuario que esté logueado en ese momento.
También incluye la publicidad, ya que aparece en todas las secciones de la web y el pie de página.
Por otra parte, incluiremos las llamadas a los contenidos estáticos, como archivos css o javascript, en mi caso, contenidos en la carpeta static. Para este hecho, tenemos que incluir en el inicio la etiqueta

{% load staticfiles %}. A cada uno de estos ficheros, en el header los llamaremos de la siguiente forma:

<link href="{% static 'css/style.css' %}" rel="stylesheet">

Donde indicamos que en static/css va a tener un fichero css llamado style. Lo haremos igual para todos los necesarios.

Al final de las inserciones correspondientes a la cabecera, entendiendo que vamos a añadir un contenido distinto en cada sección, tendremos que bloquearlo, por tanto usaremos la etiqueta {% block content%} {% endblock %}.

El restante contenido, por ejemplo el footer, ira a continuación de dicho bloqueo.

En las plantillas restantes tendremos simplemente que extender del base poniendo la siguiente etiqueta {% extends "base.html"%}, bloquear el contenido y cargar los datos estáticos y finalmente terminar el contenido, con las etiquetas que anteriormente explicamos.

4. Base de datos.

La base de datos utilizada para la web ha sido MongoDB, ya que es una base de datos potente, de código abierto y NOSQL, por tanto, perfecta para la web, ya que no va a ser necesario el relacionamiento de tablas, pueden funcionar perfectamente sin relaciones.

Una vez instalado en nuestro sistema MongoDB y pymongo, podremos acceder a la base de datos con los siguientes comandos básicos.

> mongo -> Con este comando iniciaremos nuestra base de datos en la terminal.
> show dbs -> veremos nuestras bases de datos creadas, en mi caso, desde el views.py.
> use base_de_datos -> accederemos a la base de datos seleccionada.
> show collections -> veremos las tablas creadas en la base de datos.
> db.coleccion.find() -> veremos los datos insertados en la tabla.

Como he comentado anteriormente, el views.py va a ser el encargado de crear la base de datos, por tanto, tendremos que añadir los parámetros correspondientes. Vamos a poner un ejemplo:

```
from pymongo import MongoClient
import pymongo
```

Estas cabeceras se añaden para importar el cliente de MongoDB y la librería pymongo, asegurándonos de que están instaladas en el sistema.

```
client = MongoClient()
client = MongoClient('localhost', 27017)
db = client.pw
```

Declaramos las variables que van a contener la base de datos.
La variable client la dirección de la misma y el puerto y la llamada a la propia base de datos.
Db va a contener la base de datos propiamente, la cual se va a llamar pw.

A continuación, vemos las variables que van a contener cada una de las colecciones o tablas:

```
usuarios=db['usuarios']
disco=db['disco']
comentario=db['comentarios']
```

Como vemos, la base de datos se utiliza de manera muy simple y es perfecta para realizar una web sencilla y sin contenido relacional.