Documentación Programación Web Tienda de musica Django

Alejandro José Reyes Portellano

Este proyecto esta basado en Django versión 1.9.5 con base de datos en mongodb.

Los requisitos mínimos para hacer que la practica funcione son la instalación de:

1. → Django (pip install django)
2. → Mongodb (sudo apt-get install mongodb)
3. → Pymongo (pip install pymongo)

En este proyecto he realizado una aplicación web de una tienda de musica llamada FireStorm donde podemos hacer uso de la pagina, en la que contiene los requisitos de:

1. → Registro de usuarios
2. → Login/logout
3. → Vista de perfil
4. → Modificar perfil
5. → Subida de discos de musica
6. → Reproducción de contenidos mp3.

He utilizado Django porque es un framework de desarrollo web, rápido, basado en python un lenguaje de programación amigable y mongodb porque es una base de datos (no sql) basada en colecciones (diccionarios) en las que soporta de manera bastante buena la carga y reproducción de archivos de audio y vídeo, es decir, una base de datos orientada a documentos.

Django:

La configuración de Django para el proyecto ha sido bastante sencilla se puede montar en un entorno virtual como virtualenv o hacerlo directamente en local desde tu dispositivo. Una vez que instalamos django debemos de crear el proyecto y activarlo.

Una vez este activado, debemos hacer:

```
python manage.py makemigrations
python manage.py migrate
```

Estos comandos van a definir todas las operaciones que Django va a realizar sobre la base de datos (sqlite) creada por defecto, pero en nuestro caso no nos afecta ya que utilizamos mongo como base de datos, todo en un archivo python generado automaticamente en la carpeta migrations.

La activación se produce en el archivo settings.py donde en una de las secciones de código llamada “ INSTALLED_APPS ” debemos de añadir nuestra aplicación y configurar también el directorio static que por defecto es “ STATIC_URL = '/static/' ”.

La jerarquía de archivos dentro de nuestra aplicación estan nuestros archivos python y las distintas carpetas:

Templates: donde se encuentran los html.
Static: donde encontramos, css, img, songs y js.
Migrations.

Debemos de añadirle la dirección de las carpetas, por lo menos en esta versión de django, en anteriores se hace automáticamente.

En el proyecto django nos crea varios archivos de extensión python, los archivos claves utilizados son:

Views.py

Donde van todas las funciones que ponen en funcionamiento nuestra aplicación.

Urls.py:

Este archivo contiene todas las urls de la pagina donde vamos a redirigir asignándole una función del views que va a marcar su funcionalidad.

```
url(r'^$',views.index,name='index'),
```

Este es un ejemplo de url donde el primer campo nos indica la dirección del navegador que se va a mostrar, el segundo campo la función del views que se va a aplicar y el name es el nombre de la función.

Forms.py:

Creación de modelos de formulario para la aplicación web. Donde vamos a crear los formularios pasándole un estilo y al que vamos a llamar dentro del html con estas etiquetas.

Ejemplo: {{ form_log.Nombre }}

La variable form_log la definimos dentro una o varias funciones del views como form_log=LoginForm() y en el html, llamamos a los distintos campos con form_log.Nombre, Nombre como ejemplo, pero podemos acceder a cualquiera de los campos que estén definidos en la función LoginForm del forms.py

Models.py:

En este archivo partiendo del forms.py le pasamos distintos valores a las distintas clases que tengamos en el forms.py para validar los formularios.

Una vez explicado el contenido, para ejecutar la aplicación web, nos movemos dentro de la terminal a nuestra carpeta de proyecto y ejecutamos:

```
python manage.py runserver
```

y por defecto se ejecuta en la dirección:

127.0.0.1:8000.

MongoDb:

Esta es la base de datos que hemos utilizado en nuestra aplicación. Su uso es muy sencillo, a parte de tener instalada el paquete pymongo, en nuestro archivo views.py le indicamos:

En esta linea ejecutamos mongo:

```
client = MongoClient()
```

Añadimos la dirección y el puerto para ver el registro de mongo desde nuestro navegador web:

```
client = MongoClient('localhost', 27017)
```

Creamos la base de datos:

```
db = client.tiendamusica
```

Creamos las distintas colecciones donde vamos a almacenar los datos:

```
users=db['users']
discos=db['discos']
comentarios=db['comentarios']
```

El uso básico para acceder a mongo desde la terminal y sus comandos básicos son:

- `show dbs`: muestran las bases de datos existentes.
- use 'nombre de la base de datos': seleccionar base de datos.
- `show collections`:muestran las distintas colecciones de una base de datos.
- `db.coleccion.find()`:muestra los datos que están almacenados.

FireStorm(aplicacion web):

Una vez tenemos claro el funcionamiento de django y mongo, procedemos con nuestra aplicación web.

Nuestros templates están realizados de forma dinámica, todos parten del base.html.Esto quiere decir que nuestras plantillas index.html,registro.html.. etc, heredan del base.html. Esto se realizada utilizando las etiquetas que django nos proporciona:

```
{% block content %}
     Contenido que aparecerá en los demas html.
{% endblock %}
```

Con esto tenemos siempre fijo en los html:

- **1** → Article: donde va la publicidad.
- **2** → Header: donde tenemos el formulario login y el titulo de la pagina.
- **3** → Footer: donde tenemos la información de contacto.

Una vez explicado esto, cada usuario con su inicio de login debe de aparecer en el header. Esto se ha realizado de la siguiente forma:

```
{% if 'username' in request.session %}
     {{request.session.username}}
{% else %}
     Formulario de login
{% endif %}
```

Asi podemos realizar el inicio de sesión e identificarlo en la pagina. Como debemos de tener un usuario admin que gestione la pagina, hay una sección settings, donde el admin se encargara de subir a la pagina los nuevos discos esto lo hemos representado en el html de la siguiente forma.

```
{% if 'username' in request.session and request.session.username
== 'admin' %}
          settings
{%  endif %}
```

Así, este contenido solo puede visualizarlo el usuario admin.

Procedemos a las secciones y diferentes discos.
Esta parte la he desarrollado de la siguiente manera, tenemos 4 secciones Rock,Punk,Metal y Acustic.

Cuando registras un disco, a parte de tener todos los campos previstos, le he añadido uno que se llama genero, que dependiendo del que sea, al pinchar en una sección partiendo del mismo html, te va a cargar una sección y otra de forma dinámica y con los discos pertenecientes a ese genero.

Entrando un poco en código. Dentro del section.html en el menu de selección dentro del href incluimos:

```
<a href="{% url 'section' %}?g=Rock">
```

Lo que vemos en negrita es lo que va a seleccionar el genero y va a cargar los elementos de ese genero y para eso tenemos en el views.py una funcion que realiza eso:

```
def mostrardisc(request):
    disco=request.GET.get('g',None)
    …
```

Si sigues la función en el views.py comprobamos que, si dentro de disco hay algún genero, lo busca en la base de datos y si hay lo muestre, así vamos cargando contenido en función del genero.

Además de tener una función que te busca discos por genero utilizando la función `find({genero:disco},{elementos del disco})` lo almacenamos en una lista y devolvemos el valor.

Y para los discos y comentarios se utiliza la misma filosofía.

Los discos tanto en el index, como en la sección se muestran en el html con la siguiente etiqueta:

```
{{ 'x'.0.'Titulo' }}
```

El primer campo es la variable donde le estamos pasando la lista de los discos en su función correspondiente en el views.py.
El segundo campo el número del disco en orden de almacenamiento en la base de datos.
El tercer campo, es el elemento que queremos llamar dentro de la colección.

Conclusiones:

Django combinado con mongodb es una buena herramienta de trabajo para desarrollo web, de entorno amigable y uso fácil en las que se pueden desarrollar ideas con baja dificultad.