Django

# Instalacion de python

Para poder aprovechar el diseño de esta plantilla, use la galería de estilos de la pestaña Inicio. Puede dar formato a los encabezados mediante los estilos de encabezado o resaltar texto importante con otros estilos, como Énfasis o Cita destacada. Estos estilos ya vienen preparados para ofrecer el mejor aspecto juntos y para ayudarle a comunicar sus ideas.

Continúe para probarlo.

# Instalacion de Django

**.**

**.**

**.**

# Crear Proyecto

Para crear un proyecto nuevo ejecutamos el comando:

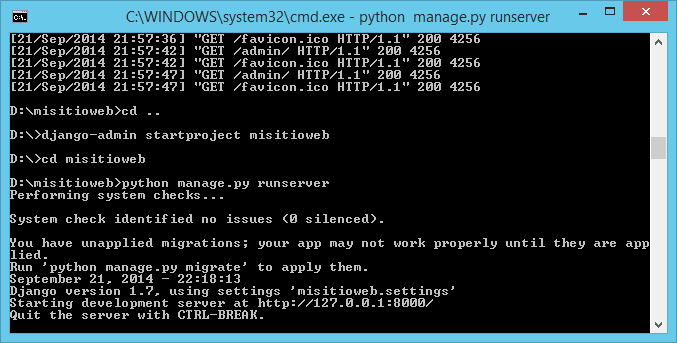
**django-admin startproject misitioweb**

ingresamos al nuevo directorio del proyecto creado:

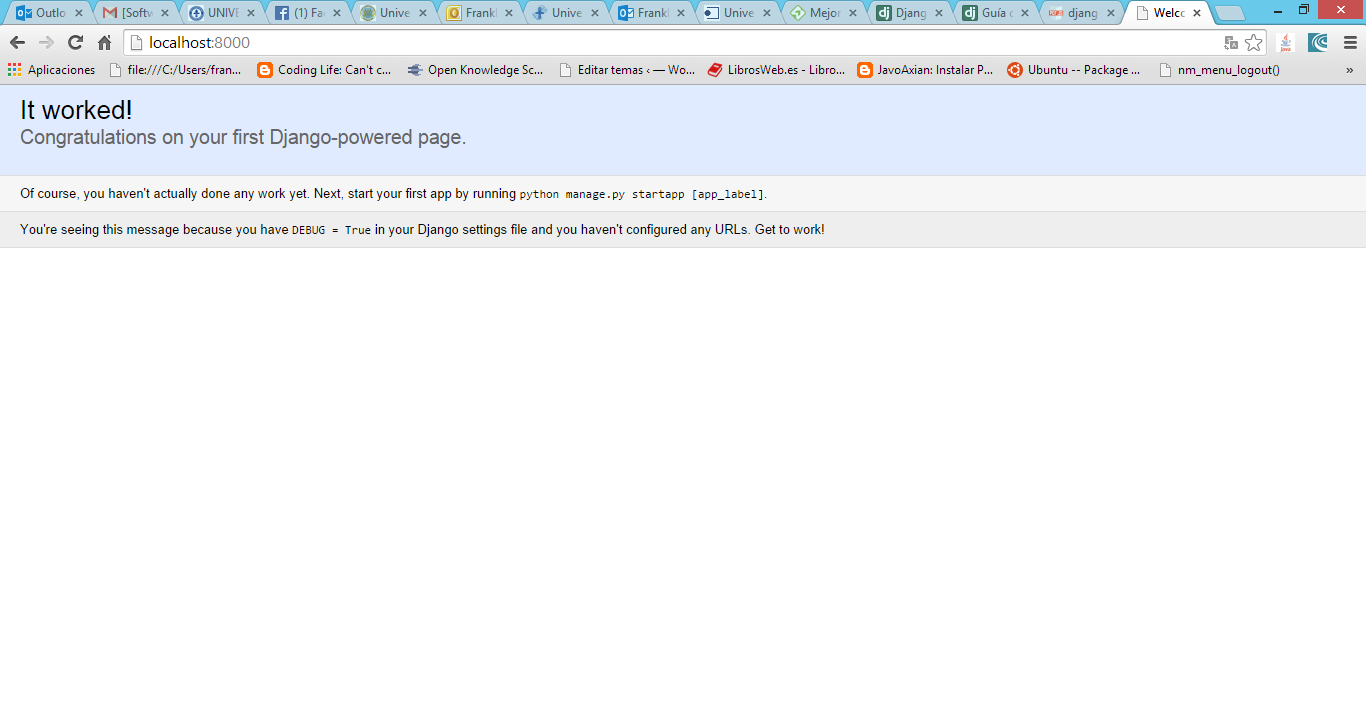
**cd misitioweb**

desde alli podemos iniciar el servidor de aplicaciones que viene con django:

**python manage.py runserver**



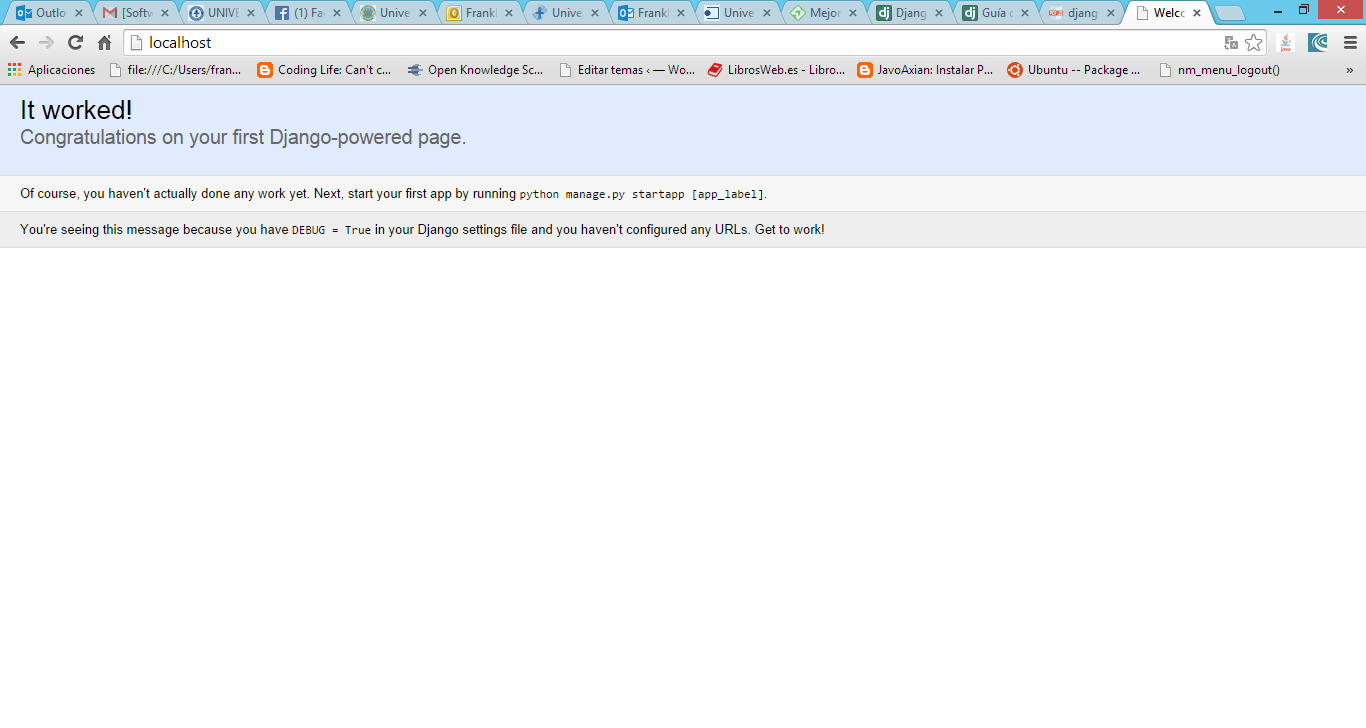
con lo cual podemos dar inicio al servidor por el puerto por defecto que para el caso es el 8000, hacemos la prueba en el navegador web de nuestra preferencia:



Es posible cambiar el puerto al numero de nuestra predileccion siempre y cuando este se encuentre libre:

**python manage.py runserver 80**

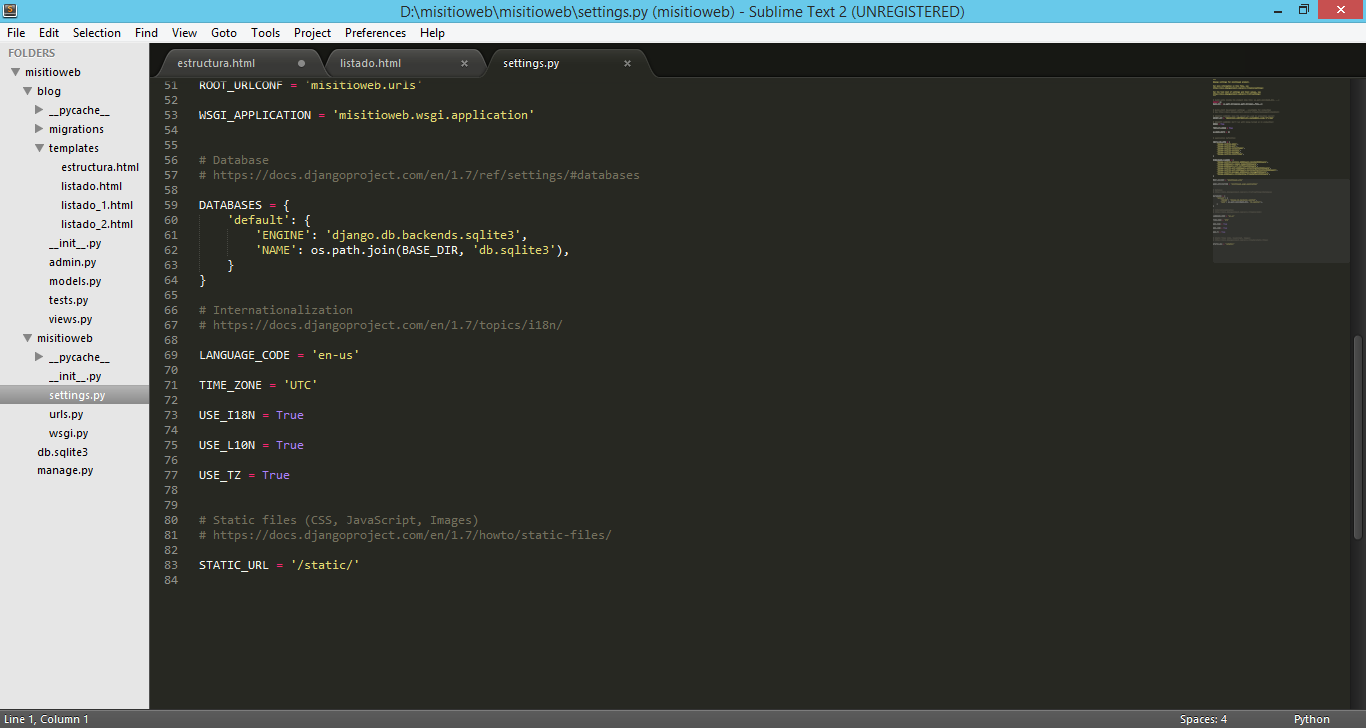
Con lo cual se da inicio al servidor en este caso por el puerto 80.



# SELECCIONAR MOTOR DE BD

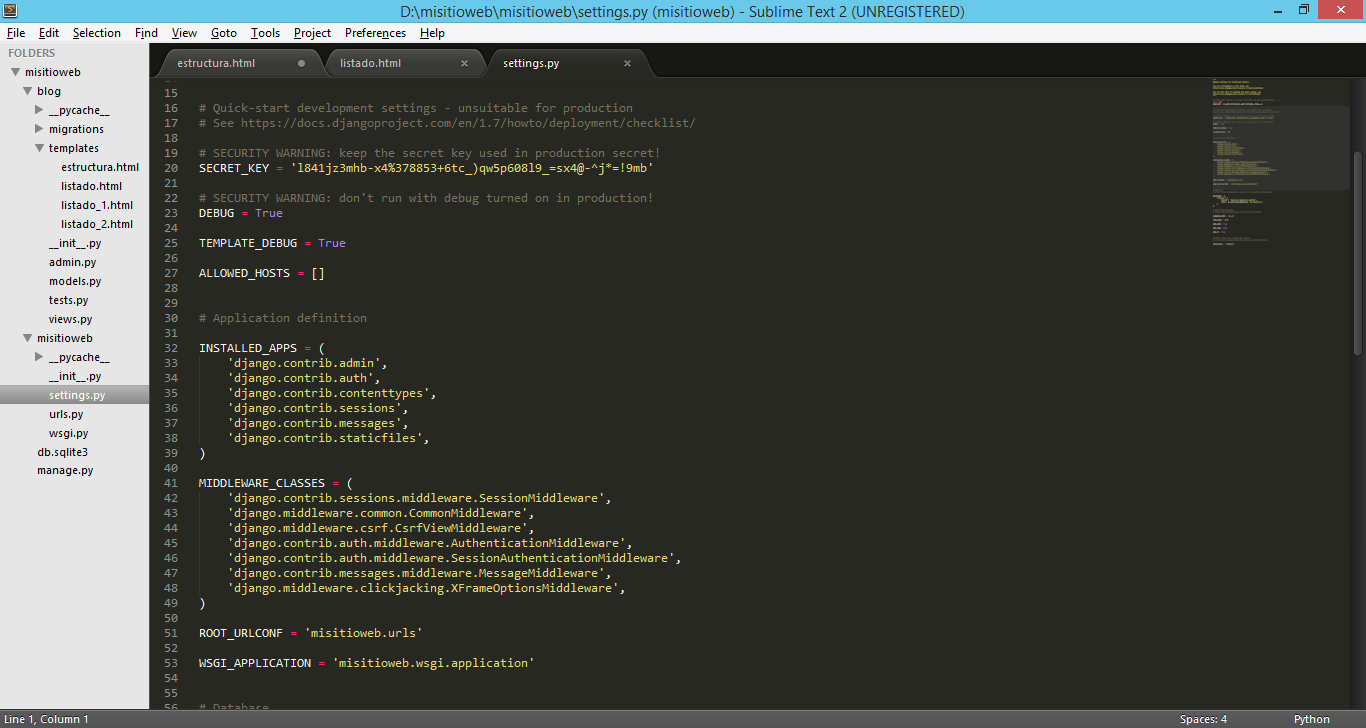
En Django para configurar la conexión a BD hacemos uso del archivo settings.py, este archivo se generó de forma automatica en la creacion del proyecto; aquí podemos configurar el motor de BD (sqlite3, postgresql, mysql, oracle, etc), el nombre de nuestra base de datos (para sqlite se refiere al nombre del archivo donde se almacenará la información) y de ser necesario tambien el usuario USER, el servidor HOST, la contraseña PASSWORD y el puerto PORT, donde se encuentre configurada nuestra base de datos.

Buscamos en settings.py la seccion donde se conigura las conexiones a BD y hacemos:



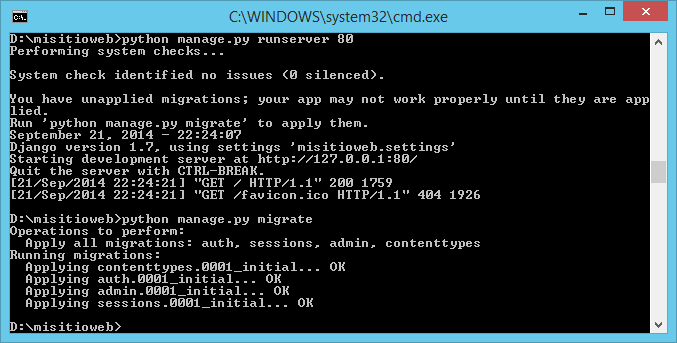
Donde db.sqlite3 se refiere al archivo donde se almacenará la información.

Algunas aplicación vienen pre-configuradas en el archivo settings.py, las cuales van a necesitar crear algunas tablas en la base de datos anteriormente configurada,



para realizar esta accion ejecutamos el comando:

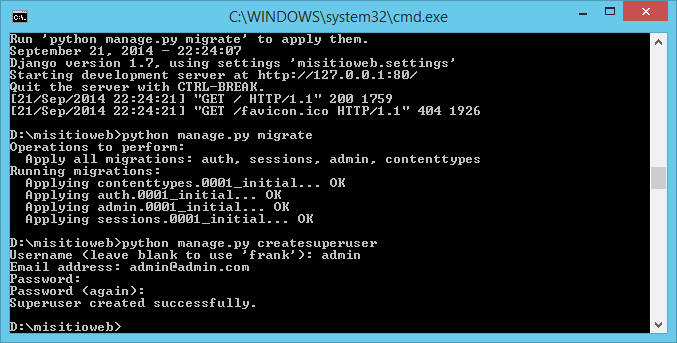
**Python manage.py migrate**



# Activar la consola de administración

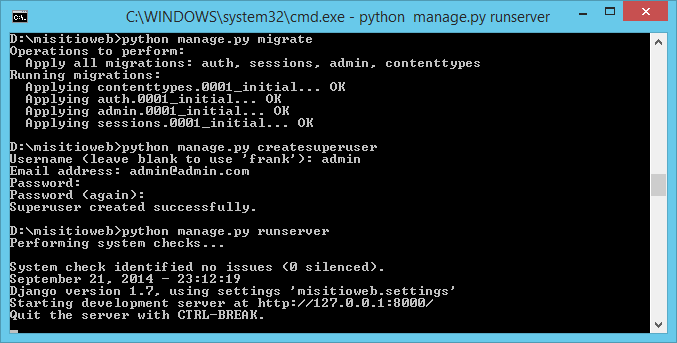
Primero tenemos que crear el superusuario que sera el administrador de nuestra consola:

**python manage.py createsuperuser**



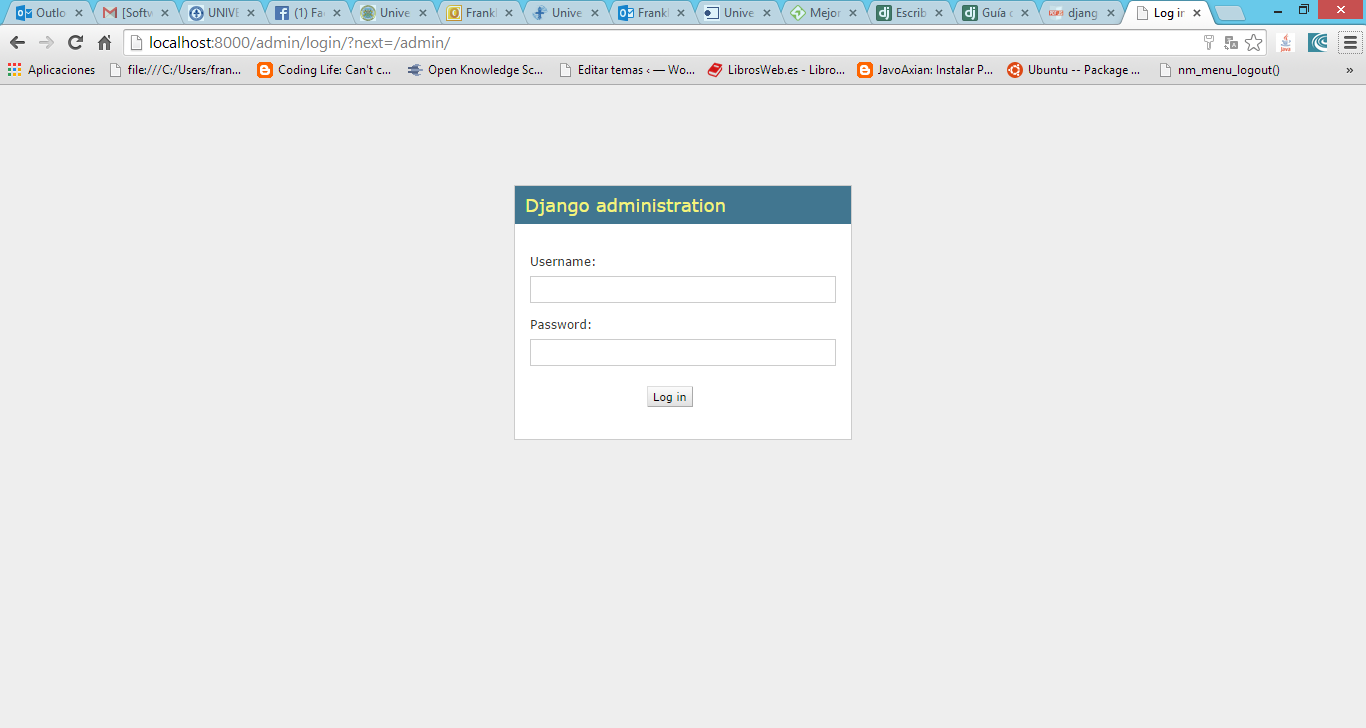
Iniciamos el servidor de desarrollo, un servidor solo para pruebas con muchas restricciones pero adecuado para realizar pruebas y ser usado en etapas de desarrollo mas no en etapa de producción.

**python manage.py runserver**

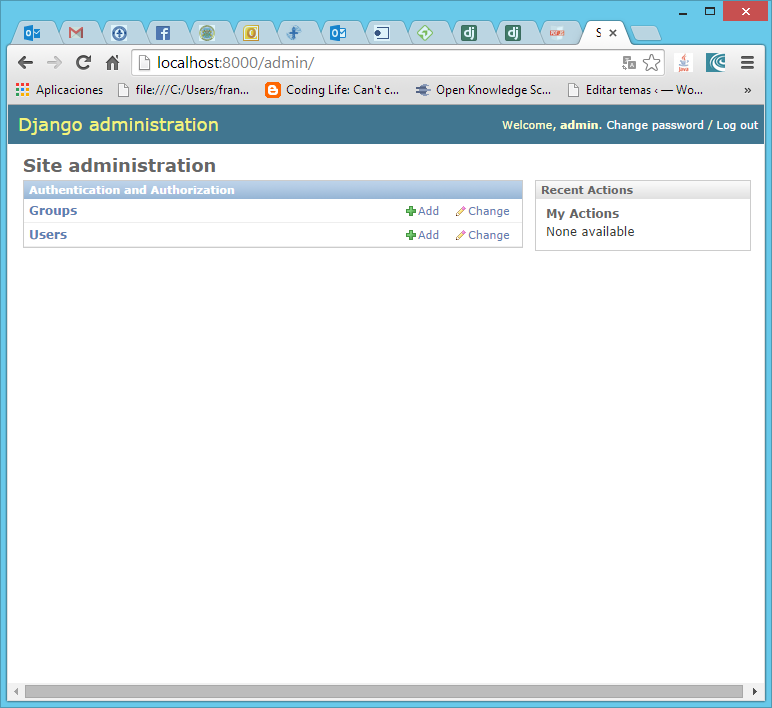


Ahora ingresamos a un navegador web y digitamos:

[**http://127.0.0.1:8000/admin/**](http://127.0.0.1:8000/admin/)



Ingresamos al sitio de administracion con la informacion proporcionada al momento de crear el superusuario.



# Crear una nueva aplicación

Creamos una aplicación dentro de nuestro proyecto llamada blog:

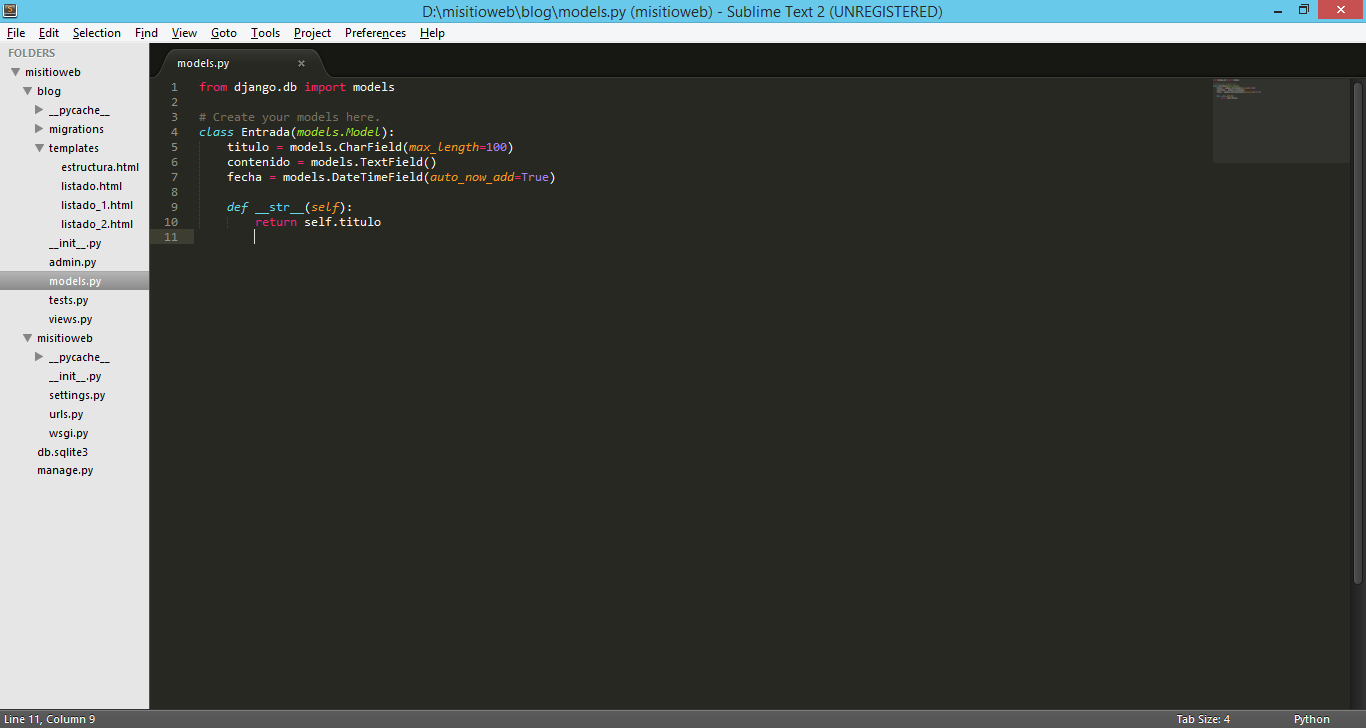
**python manage.py startapp blog**

De esta forma creamos una estructura de directorios y archivos para albergar nuestra aplicación blog.

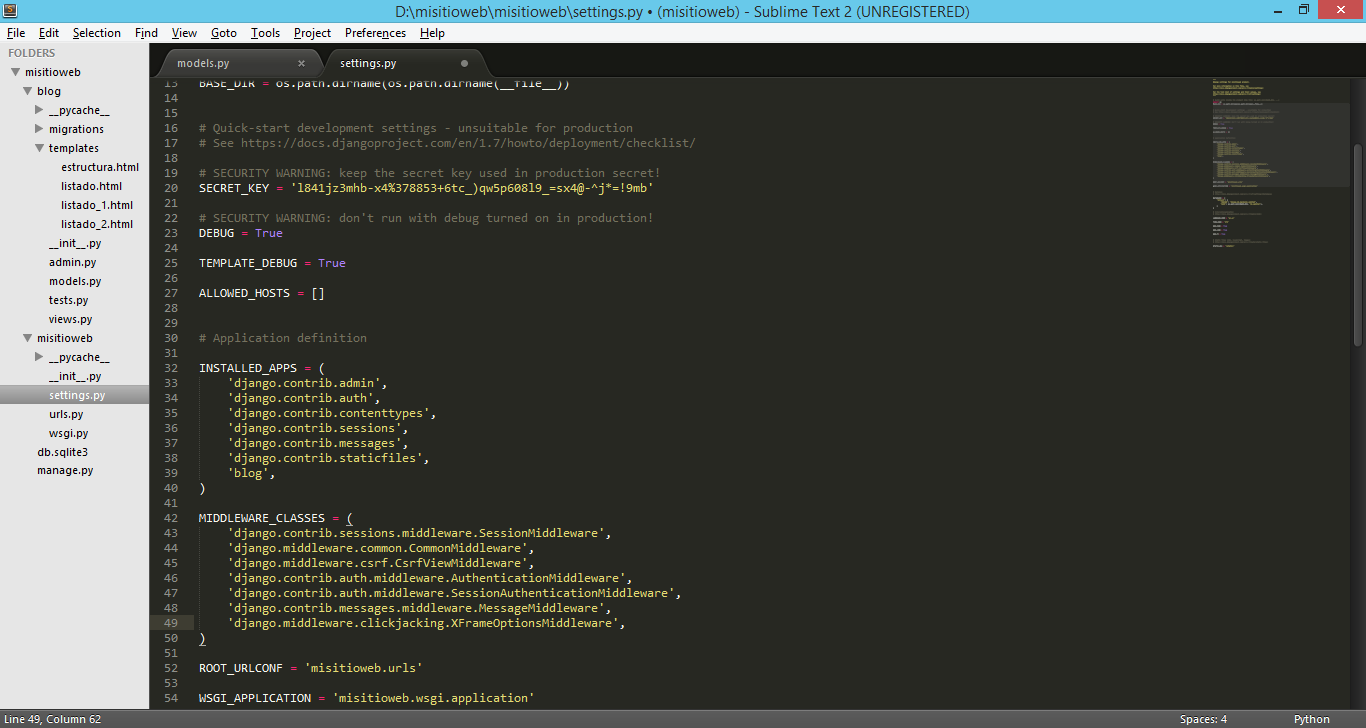
# manejo de modelos

En nuestra aplicación blog iniciaremos creando el modelo Entrada, el cual esta compuesto por un titulo, el contenido de la entrada y la fecha en la cual se realiza.

Para realizar este proceso es necesario editar el archivo models.py dentro de la aplicacion blog que se encuentra en misitioweb/blog

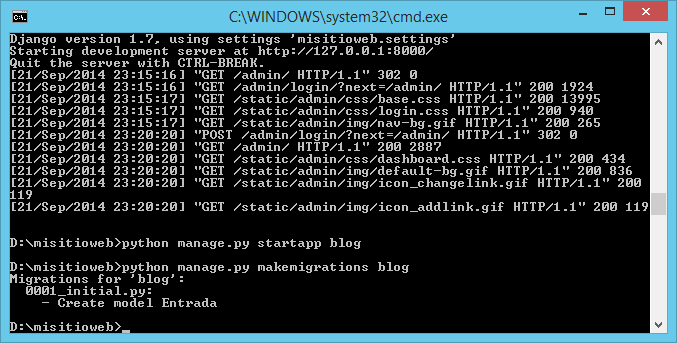


Se registra la aplicación blog en settings.py:



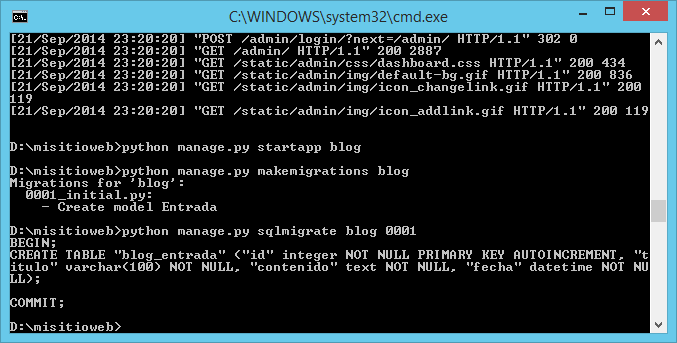
Ejecutamos un nuevo comando para decir a django que se han reaizado cambios a los modelos, en el caso particular estamos hablando de cambios en la generacion de un nuevo modelo llamado Entrada.

**python manage.py makemigrations blog**



Con el comando:

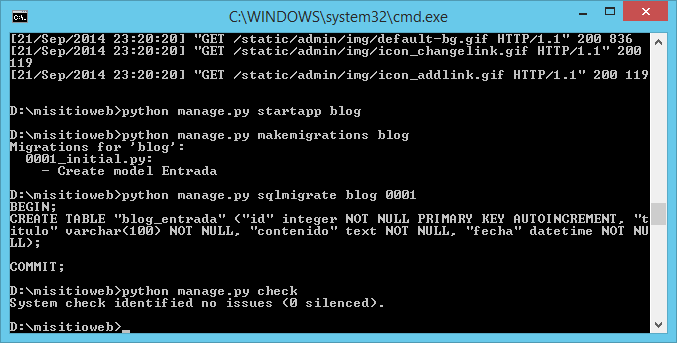
**python manage.py sqlmigrate blog 0001**



Podemos observar la sentencia sql a ejecutarse, con este comando no hemos hacho ningun cambio aun en la BD, solamente podemos observar como django interpreta el o los modelos nuevos a crear, esta sintaxis se crea de manera diferente deacuerdo al motor de BD que se esté usando.

Podemos de igual manera ejecutar el comando:

**python manage.py check**

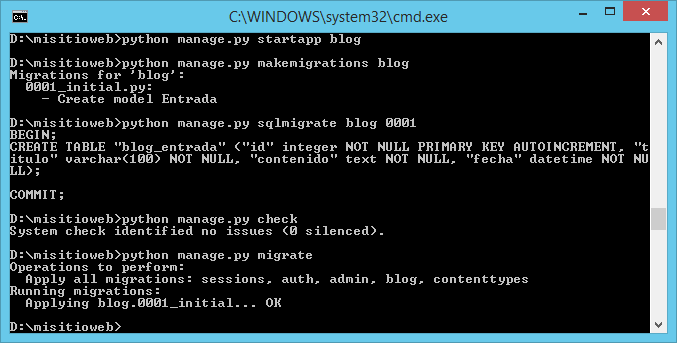


Para observar que no hay ningun problema en nuestro proyecto.

Ejecutamos:

**python manage.py migrate**

Comando que ejecuta todas las migraciones que aún no se hayan hecho.



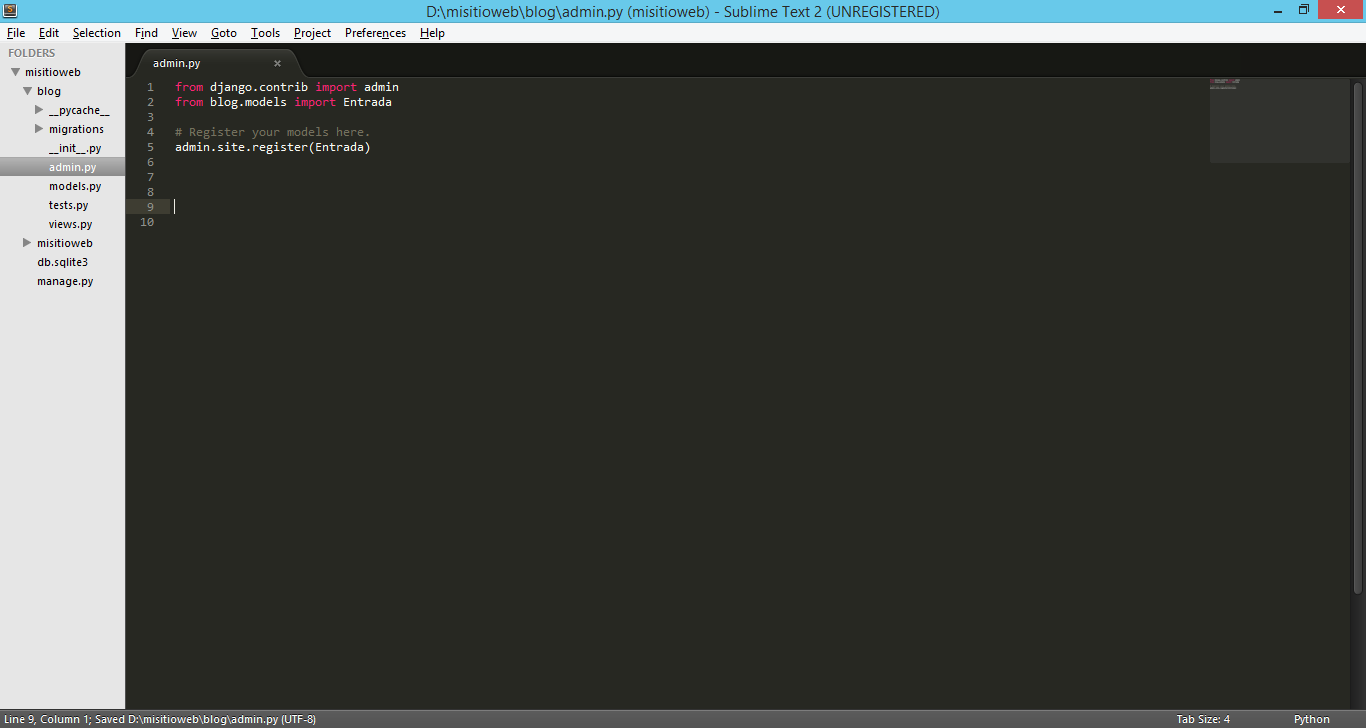
# Habilitar nuevo modelo en el administrador

Para realizar la administracion del nuevo modelo se debe modificar el archivo admin.py que se encuentra dentro del directorio de la aplicación es decir misitioweb/blog/admi.py

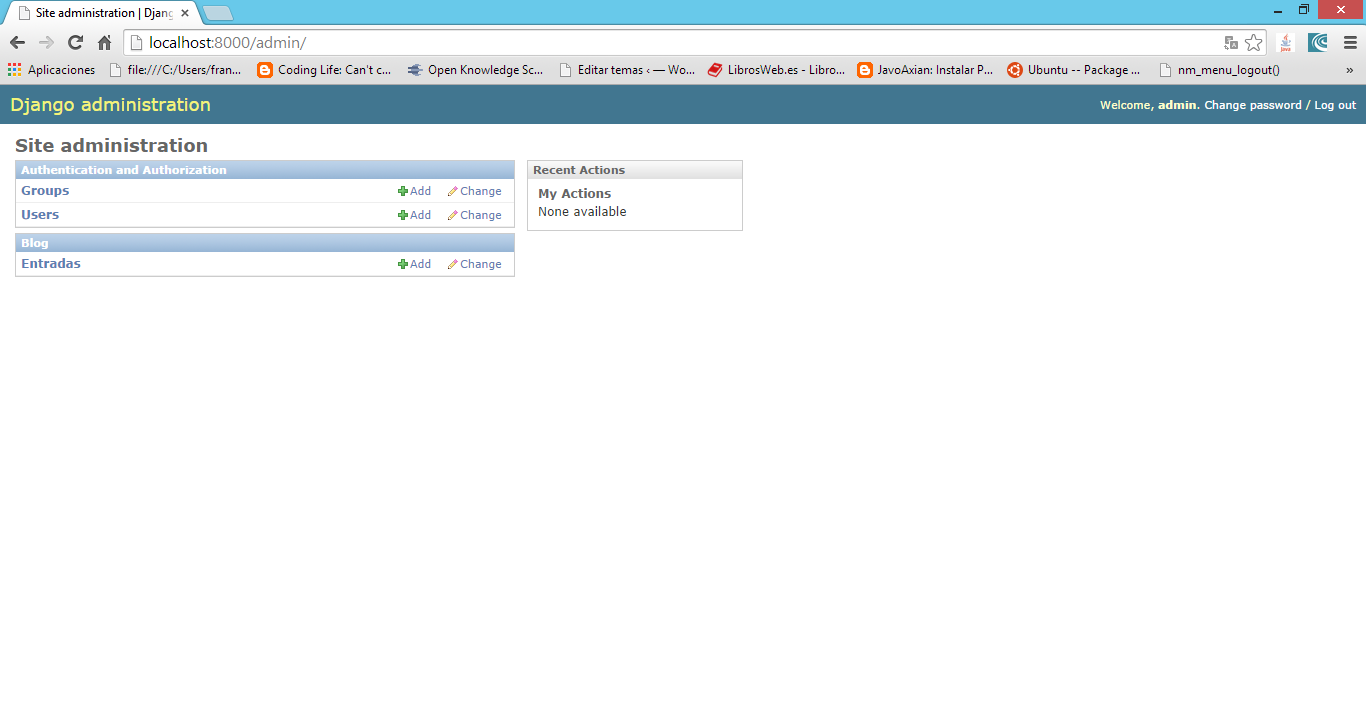
Adicionamos las lineas:

from blog.models import Entrada

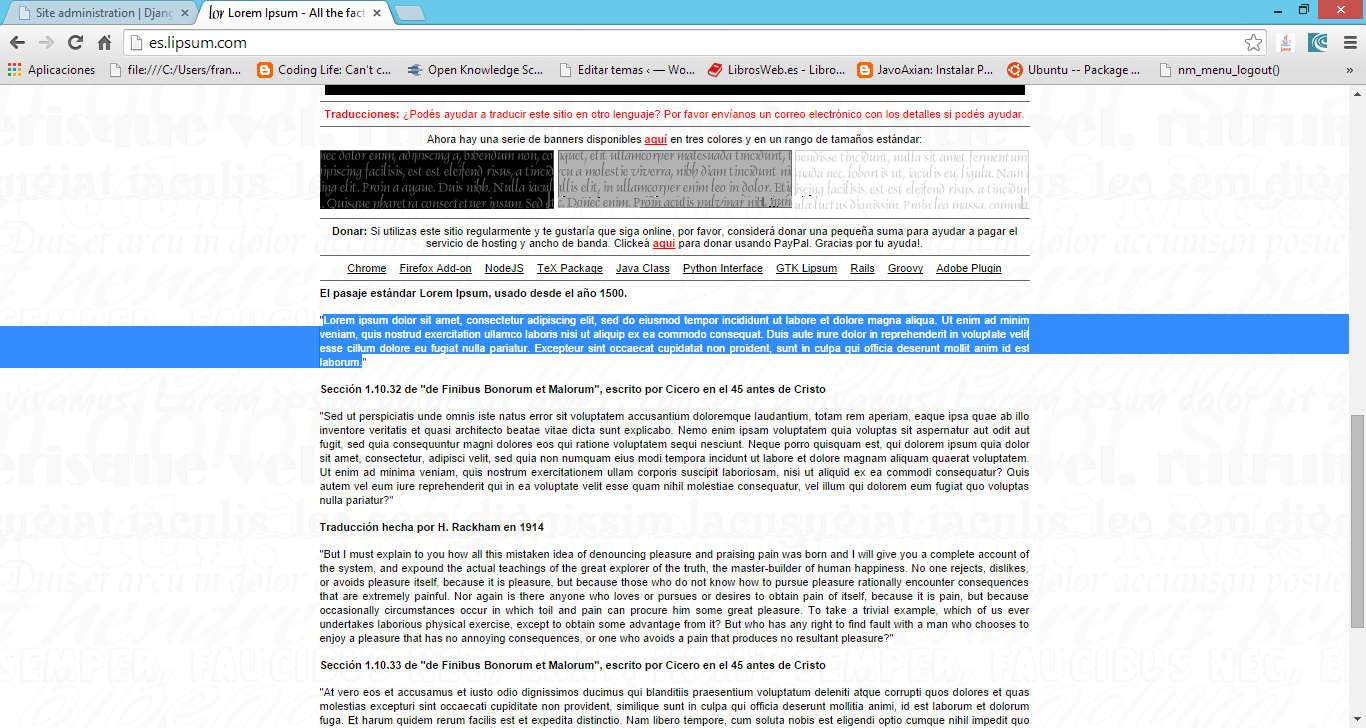
admin.site.register(Entrada)



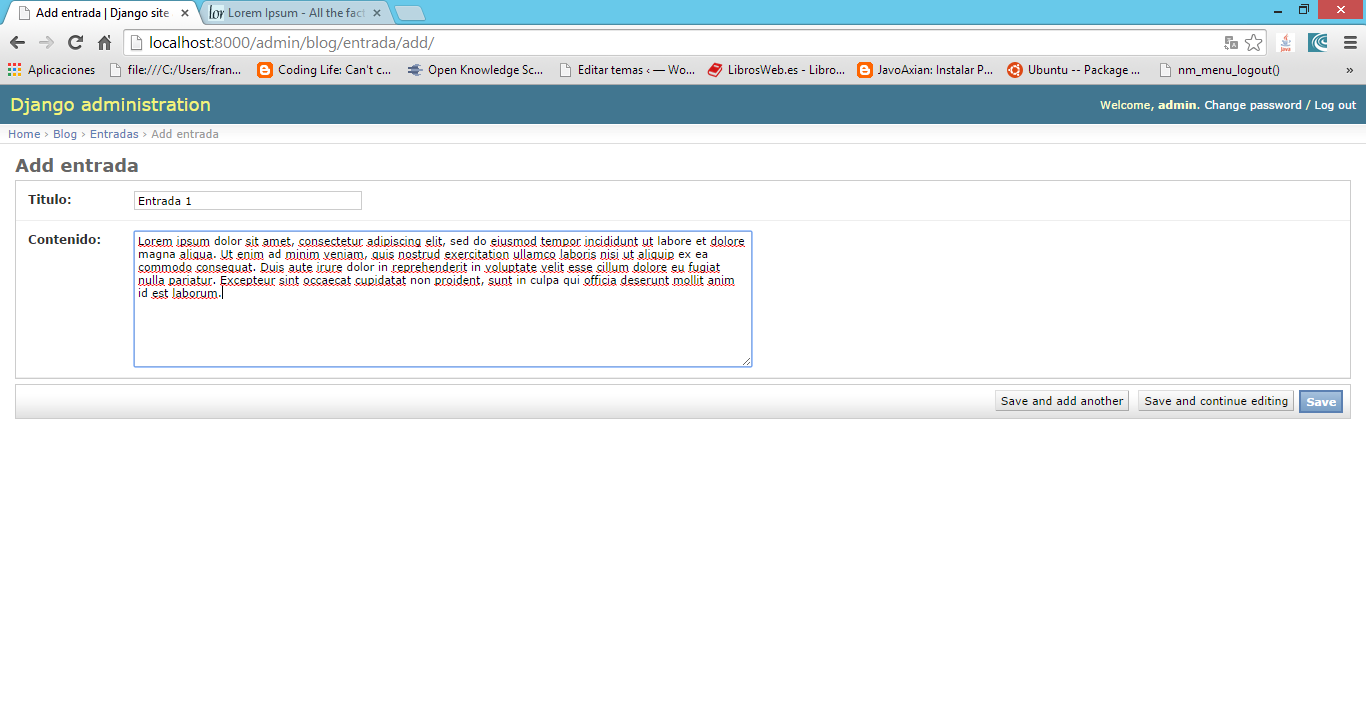
Ingresamos al administrador de Django para insertar contenido al nuevo modelo:



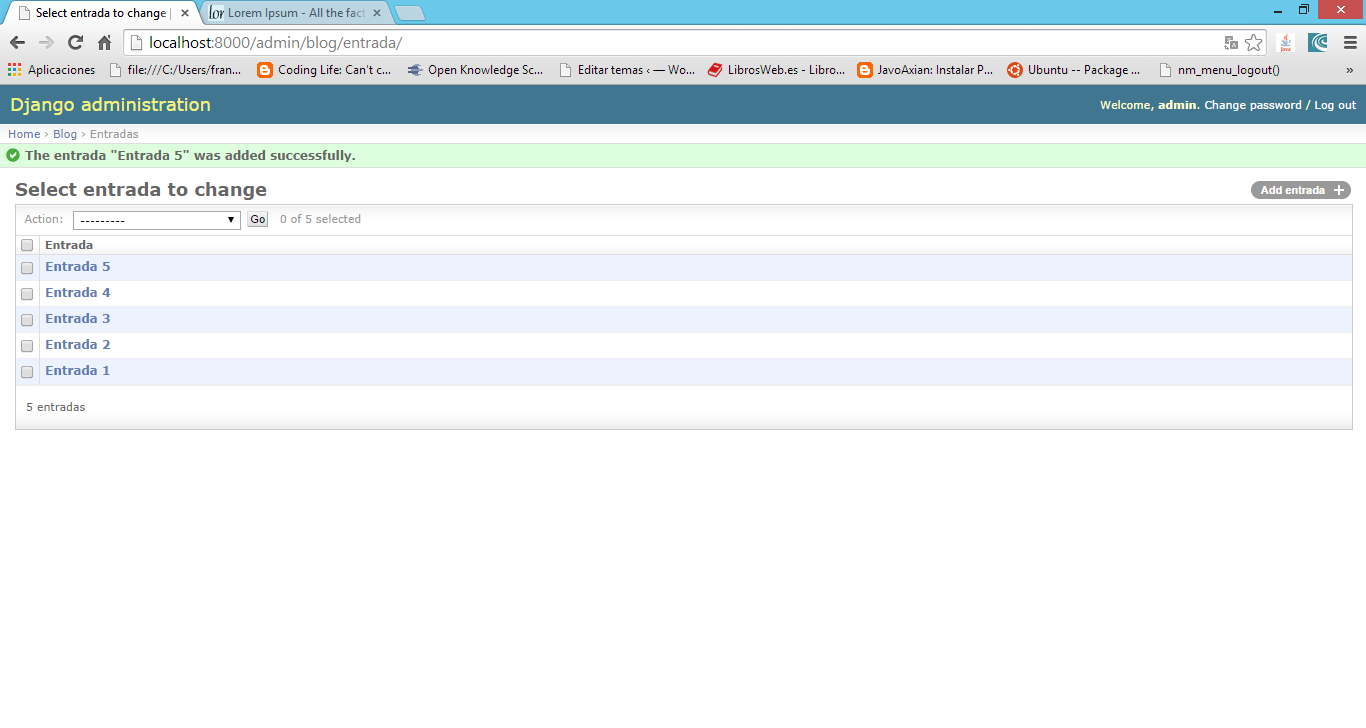
Para ingresar algunos datos de prueba en este modelo hacemos la busqueda en google de la palabra “lorem ipsum” y de los resultados de la busqueda ingresamos al link <http://es.lipsum.com/>, desde alli seleccionamos uno de los parrafos con informacion para ser usado como contenido de nuestras entradas, asi:



Realizamos el registro de 5 nuevas entradas:



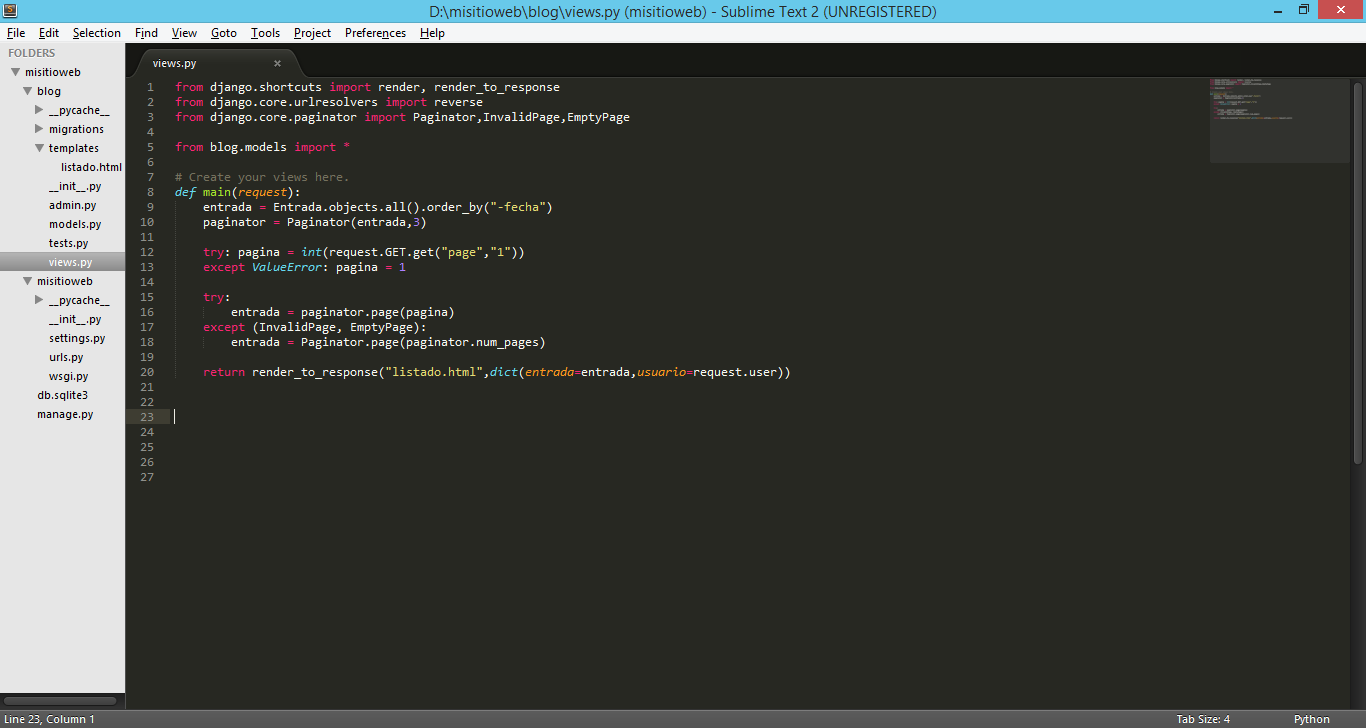
Al final debemos observar un listado como el siguiente:



# Crear una vista para mostrar contenido

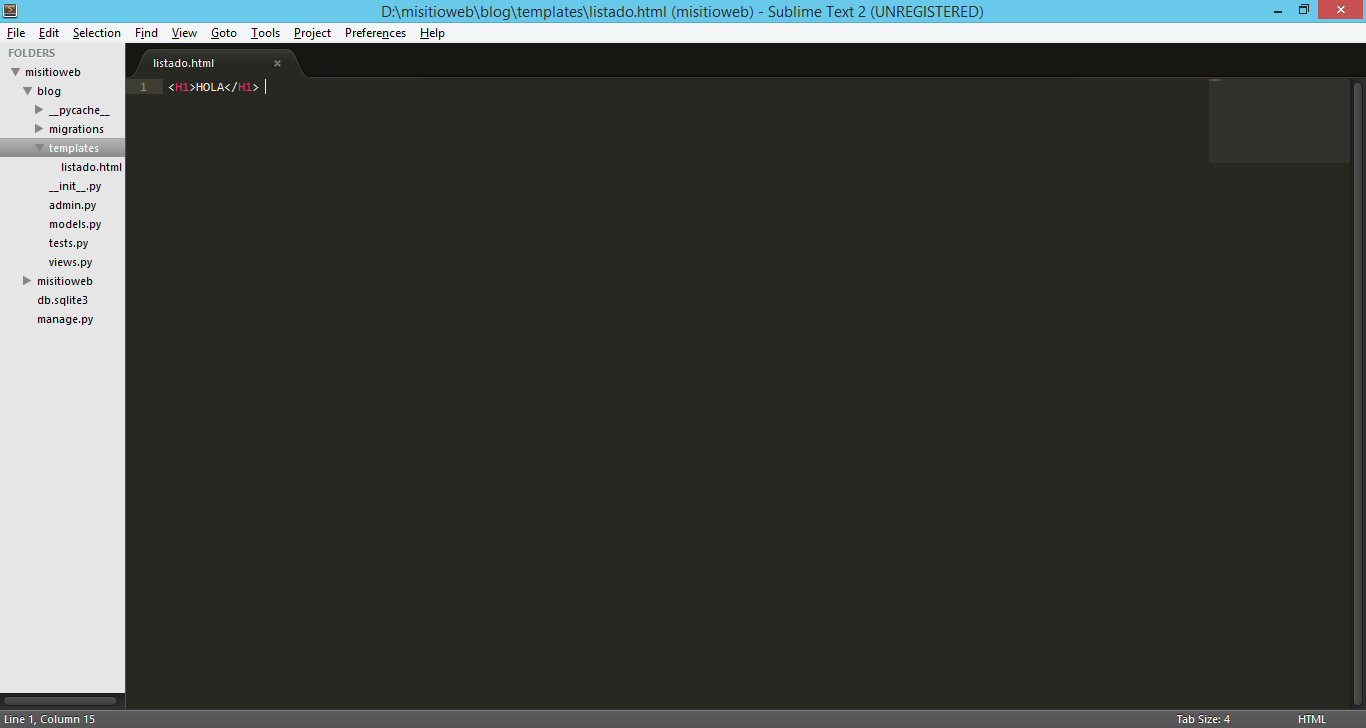
En el archivo views.py se registra la nueva vista que servira para mostrar el contenido de nuestras entradas.

Creamos la nueva vista llamada main, en ella hacemos la consulta de todos los registros de la tabla Entrada:

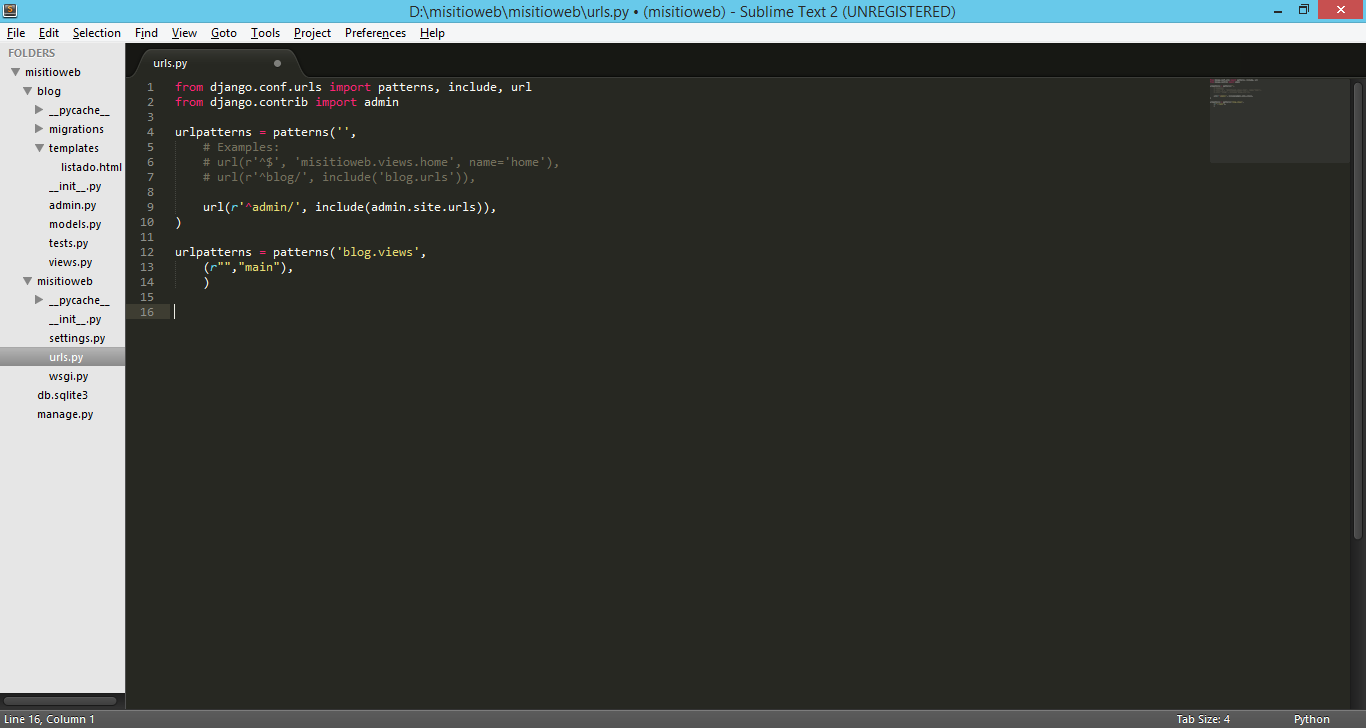


Crear el directorio “templates” dentro del directorio de la aplicación, **misitioweb/blog/templates,** dentro de este directorio creamos un archiuvo llamado “**listado.html**” el cual simplemente va a tener como contenido:

**<H1>HOLA</H1>**

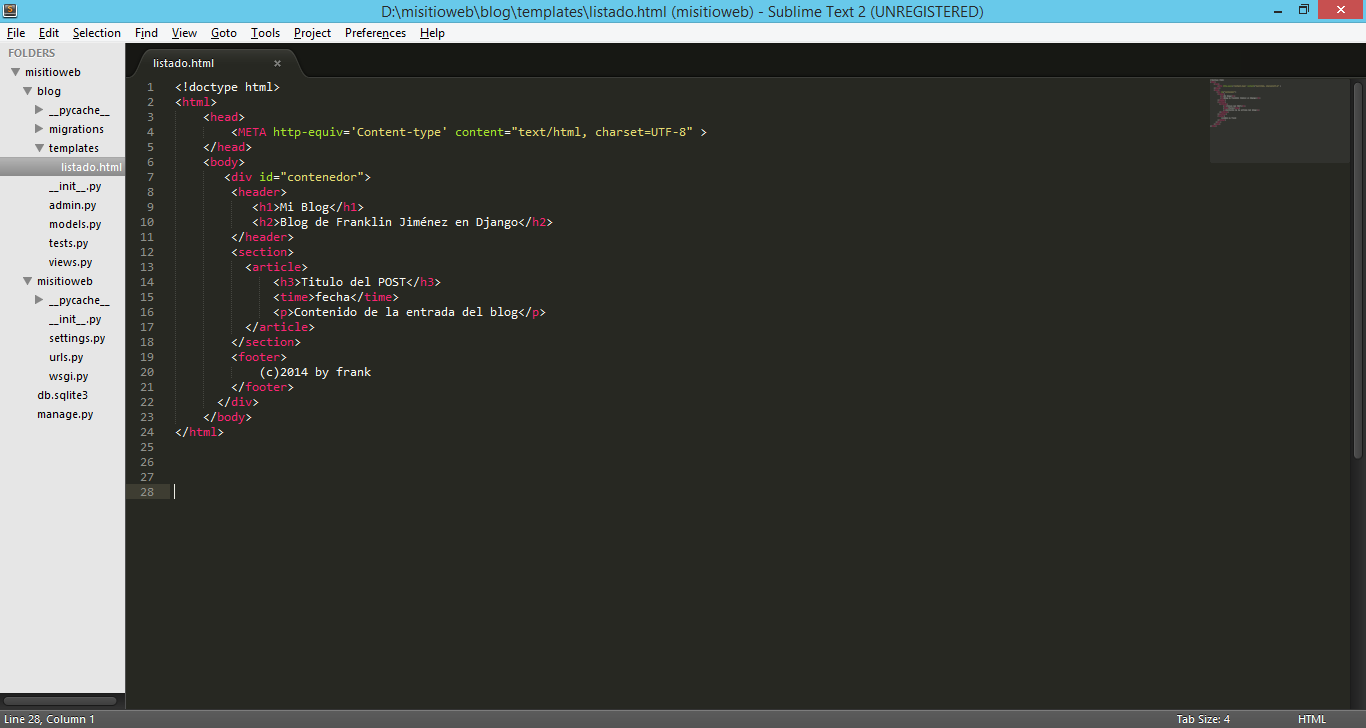


Ahora se debe realizar cambios al archivo **misitoweb/misitioweb/urls.py**

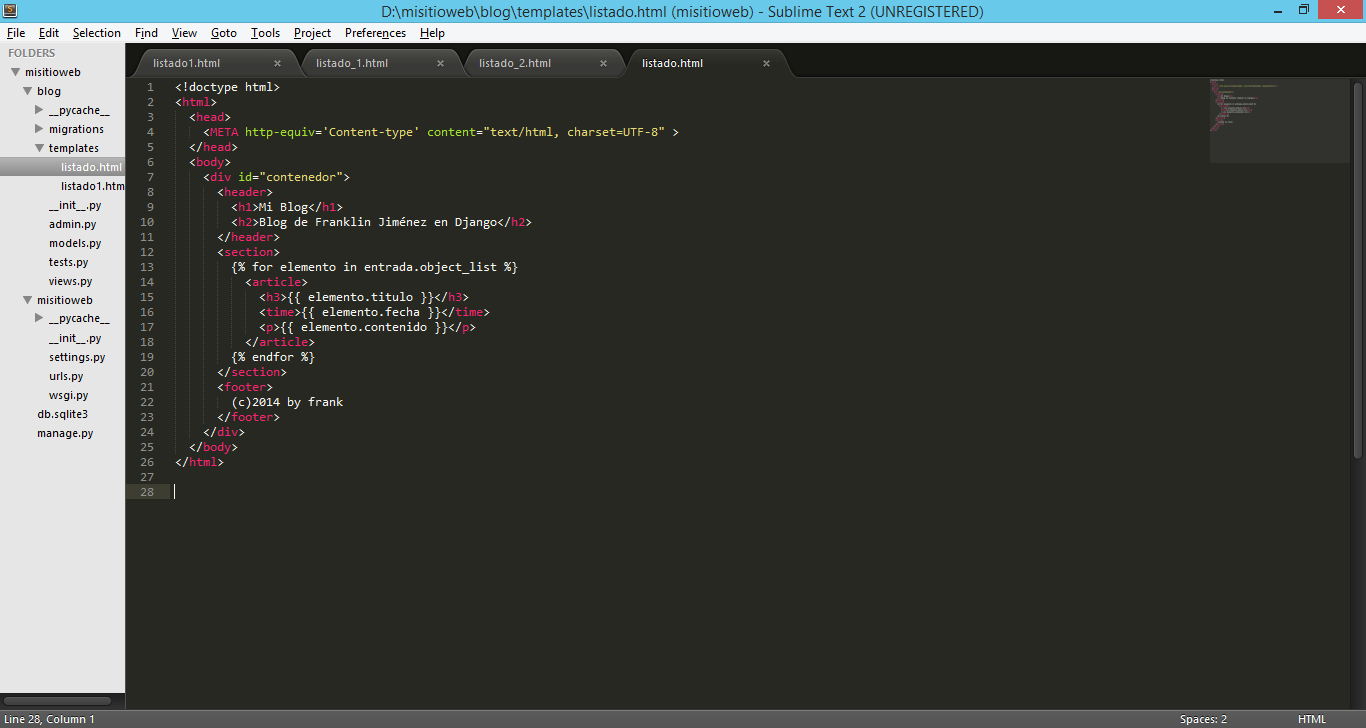


# Programar la plantilla listado.html

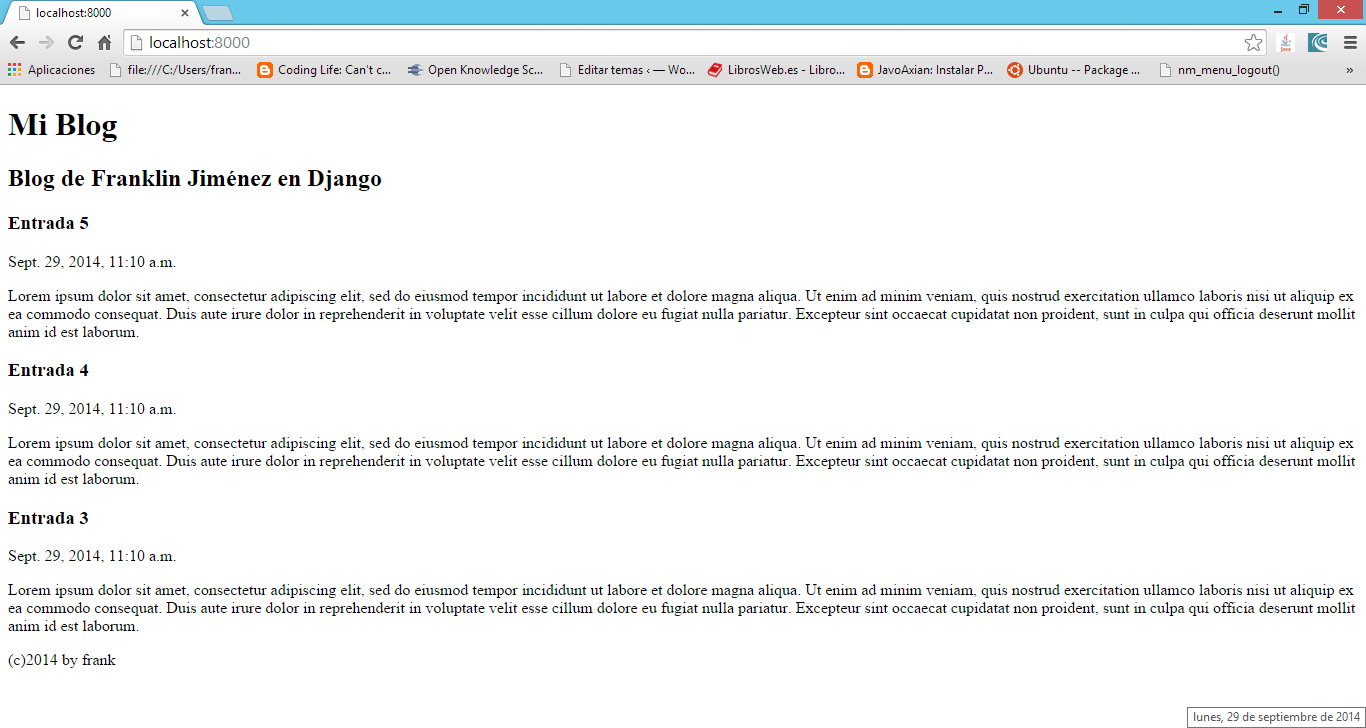
Crear la plantilla listado.html con un contenido estatico:



Para que el contenido se muestre de manera dinamica se hace los cambios respestivos en listado.html:

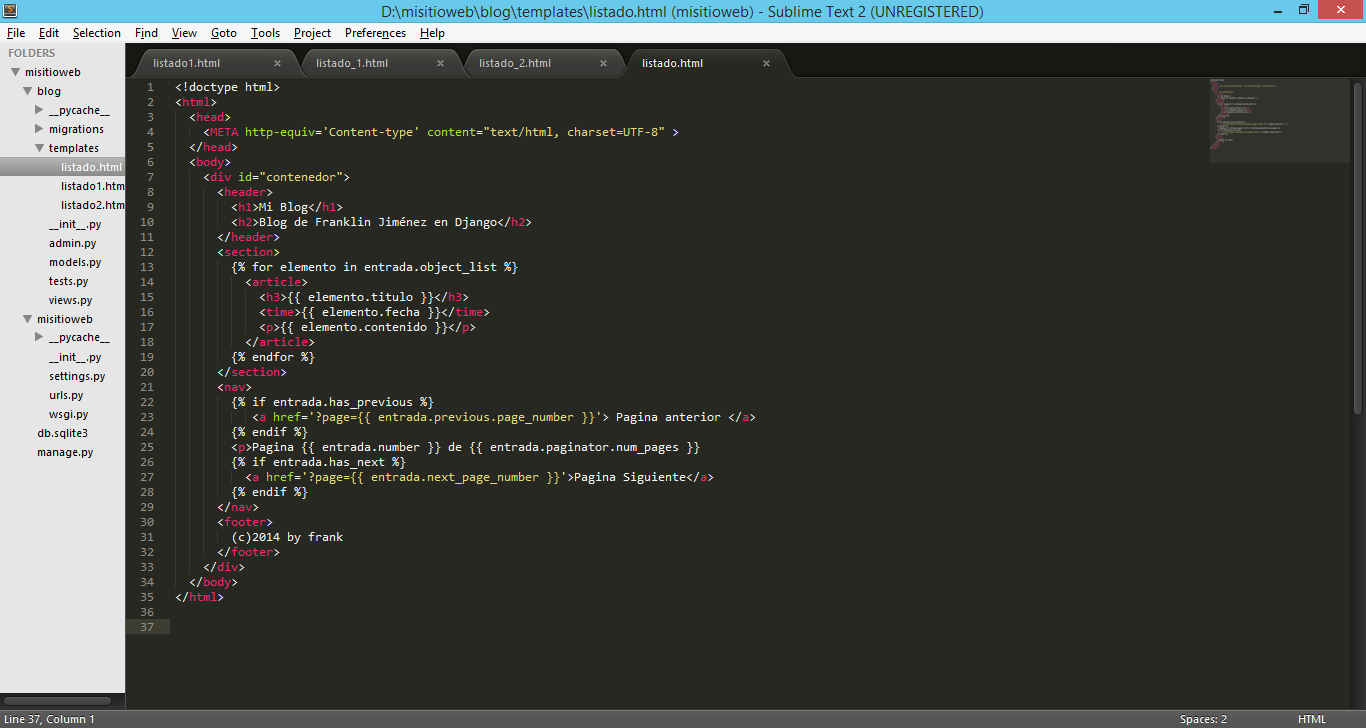


De esta forma podemos observar el contenido de las tres ultimas entradas, solo se muestran tres debido a la paginacion asignada en la vista main.



# Navegación entre páginas

Para la navegacion entre las paginas de los resultados hacemos uso de la etiqueta html5 <nav>, haciendo uso de hipervinculos con las opciones para cambio de pagina:



# Extensiones de plantillas

Creamos en templates el archivo estructura.html en donde cortamos y pegamos todo el contenido HTML y en listado.html solo dejamos el codigo dinamico.

**estructura.html**

<!doctype html>

<html>

<head>

<META http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<div id="contenedor">

<header>

<a href="http://localhost" title="Mostrar página principal"><h1>Mi Blog</h1></a>

<h2>Blog de Franklin Jiménez en Django</h2>

</header>

<section>

<div id="articulos">

{% block contenido %}{% endblock %}

</div>

<aside>

<h4>Histórico</h4>

{% for month in months %}

<a href="{ url 'blog.views.month' month.0 month.1 }"><p>{{ month.2 }}</p></a>

{% endfor %}

<h4>Sobre Mi</h4>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>

</aside>

</section>

<div style="clear:both;"></div>

<footer>

(c) by frank

</footer>

</div>

</body>

</html>

**listado.html**

{% extends "estructura.html" %}

{% block contenido %}

{% for entrada in entrada.object\_list %}

<article>

<h3>{{ entrada.titulo }}</h3>

<time>{{ entrada.fecha }}</time>

<p>{{ entrada.contenido }}</p>

</article>

{% endfor %}

<nav>

{% if entrada.has\_previous %}

<a href='?page={{ entrada.previous\_page\_number }}'>Pagina Anterior</a>

{% endif %}

<P>Pagina {{ entrada.number }} de {{ entrada.paginator.num\_pages }}</P>

{% if entrada.has\_next %}

<a href='?page={{ entrada.next\_page\_number }}'>Pagina Siguiente</a>

{% endif %}

</nav>

{% endblock %}