Contenidos a Trabajar

1. ¿Qué es Django?
2. Estructura de Directorios de Django
3. Primer Proyecto Django
4. URL dispatcher

Agencia de Aprendizaje a lo largo de la vida

¿Qué es Django?

Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago.

Django fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible. Se toma la seguridad en serio y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes.

Django incluye docenas de extras que puede usar para manejar tareas comunes de desarrollo web. Django se encarga de la autenticación del usuario, la administración de contenido, los mapas del sitio, las fuentes RSS y muchas más tareas, desde el primer momento.

Django se toma la seguridad muy en serio y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes, como la inyección de SQL, las secuencias de comandos entre sitios, la falsificación de solicitudes entre sitios y el secuestro de clics. Su sistema de autenticación de usuarios proporciona una forma segura de administrar las cuentas y contraseñas de los usuarios.

Empresas, organizaciones y gobiernos han utilizado Django para crear todo tipo de cosas, desde sistemas de gestión de contenido hasta redes sociales y plataformas informáticas científicas.

Django fue desarrollado inicialmente entre 2003 y 2005 por un equipo que era responsable de crear y mantener sitios web de periódicos. Después de crear varios sitios, el equipo empezó a tener en cuenta y reutilizar muchos códigos y patrones de diseño comunes. Este código común se convirtió en un framework web genérico, que fue de código abierto, conocido como proyecto "Django" en julio de 2005.

Django ha continuado creciendo y mejorando desde su primer hito, el lanzamiento de la versión (1.0) en septiembre de 2008, hasta el reciente lanzamiento de la versión 1.11 (2017). Cada lanzamiento ha añadido nuevas funcionalidades y solucionado errores, que van desde soporte de nuevos tipos de bases de datos, motores de plantillas, caching, hasta la adición de funciones

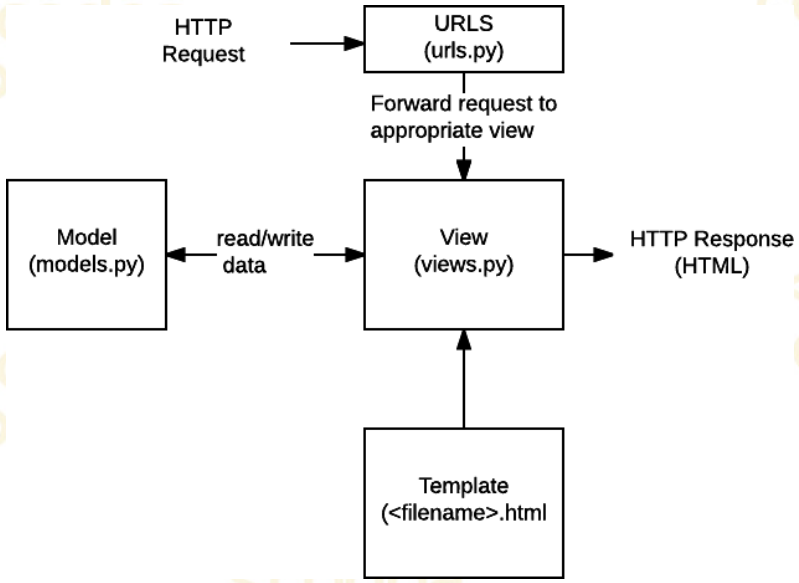
genéricas y clases de visualización (que reducen la cantidad de código que los desarrolladores tiene que escribir para numerosas tareas de programación). Hoy endía se encuentra en su versión 4.0.

Los sitios de alto nivel que usan Django incluyen: Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest y Open Stack

https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

Patrón MVT

Django se basa en la arquitectura MVT (Model-View-Template) . MVT es un



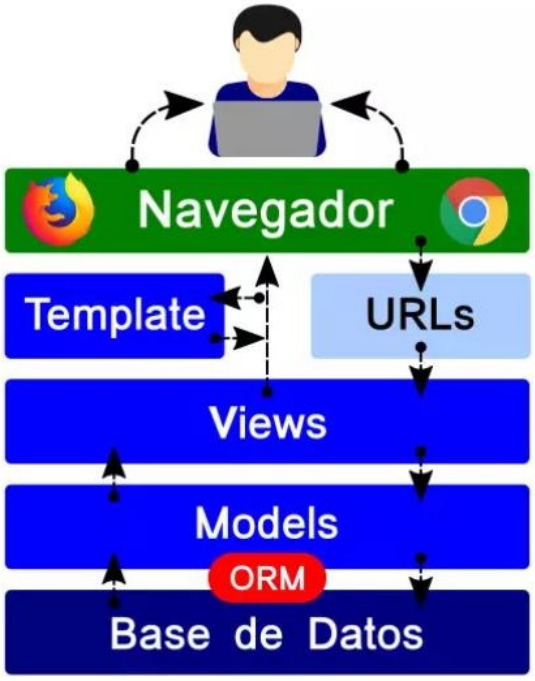
patrón de diseño de software paradesarrollar una aplicación web

- *M* significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos. En esta capa es donde se utiliza el ORM de Django para comunicarse con la base de datos y realizar un manejo transparente para el desarrollador.
- *V* significa "View" (Vista), la capa de la lógica de negocios. Esta capa

contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelo y las plantillas.

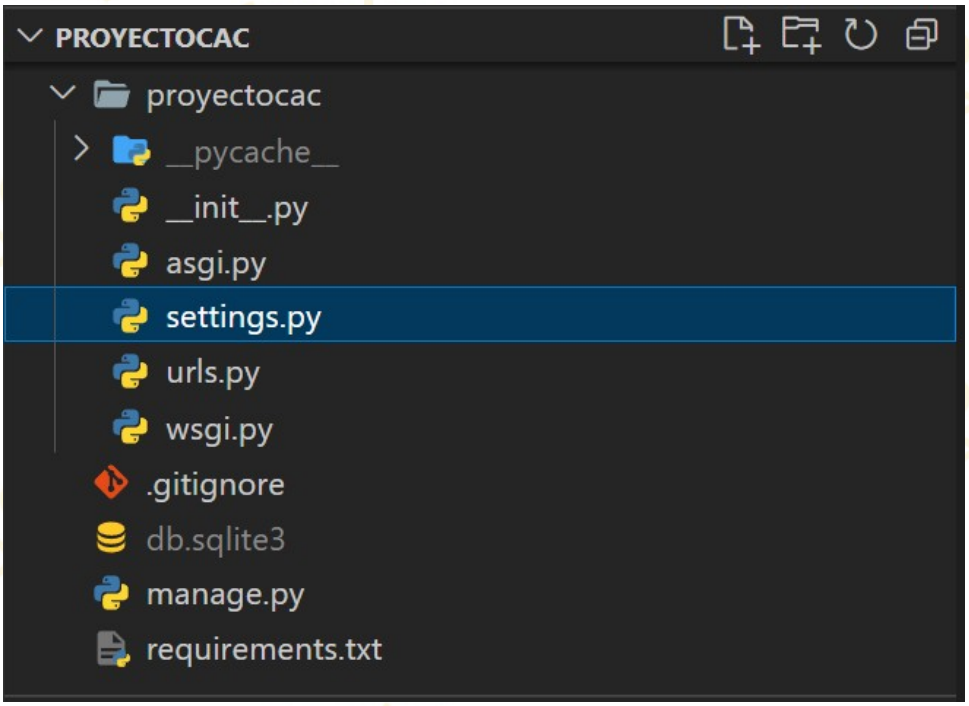
- *T* significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web o otro tipode documento.

Otra representación del mismo modelo puede ser la siguiente:



Estructura de directorios Django

Aquí podremos ver la estructura general de directorios de cualquier proyecto Django. Que más adelante veremos en detalle. Para mostrar dicha estructura, tomamos un proyecto denominado “proyectocac” para elejemplo.



manage.py: Una utilidad de la línea de comandos que le permite interactuar con este proyecto Django dediferentes formas.

__init__.py: Un archivo vacío que le indica a Python que este directorio debería serconsiderado como un paquete Python.

settings.py: Ajustes/configuración para este proyecto Django. Django settings le indicará todo sobre cómofunciona la configuración.

urls.py: Las declaraciones URL para este proyecto Django; una «tabla de contenidos» de su sitio basado enDjango.

wsgi.py: Un punto de entrada para que los servidores web compatibles con WSGI puedan servir su proyecto.

asgi.py: Un punto de entrada para que los servidores web compatibles con ASGI puedan servir su proyecto.

Primer proyecto Django

Ya teniendo un entorno virtual con el Framework instalado, como hemos visto en semanas anteriores, lo primero que debemos hacer es activar el entorno virtual en el cual deseamos trabajar con nuestro proyecto Django. Para esto en la consola deberemos ejecutar:

path \mientornovirutual \Scripts \activate

Una vez activado el entorno virtual, para verificar que el paquete de Django se instaló correctamente podemosejecutar en la consola los siguientes comandos:

python

>>> import django

>>> print(django.get_version())

En caso de que el framework no se encuentre instalado, debemos hacerlo con pip (revisar módulos anteriores donde se explica la instalación)

Una vez instalada las librerías de Django, podremos desde nuestra consola consola/terminal, crear un nuevo proyecto ejecutamos el siguiente comando

django-admin startproject "nombreproyecto"

Django provee un servidor web local para desarrollo, en el cual podremos probar lo que vamos desarrollando de nuestro proyecto de manera local, sin necesidad de realizar un despliegue en un servidor real (esto aplica a cualquier de los sistemas operativos disponibles). Para eso, desde la consola nos posicionamos en el directorio donde se encuentra nuestro proyecto, y ejecutamos el manage.py (utilidad que usaremos para interactuar con Django desde la línea de comando) seguido del comando runserver, por ejemplo:

cd .\"nombreproyecto" python manage.py runserver

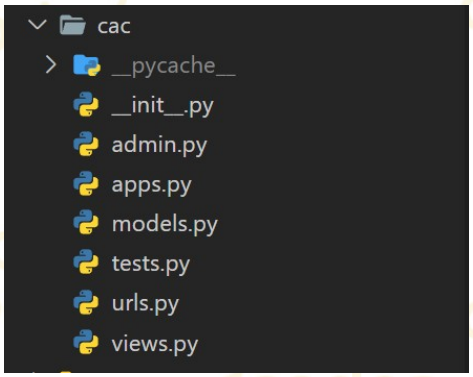
Este comando inicializará un servidor web local de desarrollo al cual podremos acceder por medio de lasiguiente url

http://127.0.0.1:8000/

Un proyecto de Django puede estar compuesto por una o más aplicaciones (que tendrán una funcionalidad específica, y en su conjunto brindarán todo lo necesario para que nuestro proyecto cumpla con los requerimientos funcionales). Para crear una aplicación vamos a tener que ejecutar el siguiente comando:

python manage.py startapp “cac”

Esta acción nos creará un nuevo directorio el cual tendrá una arquitectura muy similar a la que Django creo alcrear nuestro proyecto



Por último, tendremos que registrar nuestra aplicación creada dentro del archivo settings.py, para elloagregamos el nombre de nuestra aplicación en la lista de INSTALLED_APPS

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cac'
]
```

URL dispatcher

Un esquema de URL limpio y elegante es un detalle importante en una aplicación web de alta calidad. Django fomenta un diseño de URL hermoso y no añade ningún elemento innecesario en las URLs, como **.php** o **.asp**. Para ello Django cuenta con un archivo de configuración de rutas urls.py en el cual vamos a poder detallar todas las URL que queremos se encuentren disponibles en nuestro proyecto e indicarle también con que métodos de la Vista estarán asociados. Estos archivos URL sirven además para desacoplar el manejo de URLs del código de Python de negocio.

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),

]
```

Cuando un usuario solicita una página de nuestro sitio impulsado por Django, este es el algoritmo que sigue el URLDispatcher para determinar qué código de Python (vista) ejecutar:

1. Django carga ese módulo de Python y busca la variable urlpatterns. Esta debería ser una lista de Python, en el formato devuelto por la función **django.conf.urls.patterns()** .
2. Django recorre cada patrón de URL, en orden, y se detiene en el primero que coincida con la URL solicitada (si ninguno de ellos coincide Django llama a una vista del caso especial 404).
3. Una vez que una de las expresiones regulares coincide, Django importa y llama a la vista dada, que es una función de Python simple (o una vista basada en clases). A la vista se le pasa una **HttpRequest** como su primer argumento y cualquier valor capturado en la expresión regular como argumentos restantes.
4. Si ninguna expresión regular coincide, o si se genera una excepción durante cualquier punto de este proceso, Django invoca una vista de manejo de errores adecuada.