# ¿Qué es Django?

Django es un framework para aplicaciones web gratuito y open source, escrito en Python. Es un WEB framework - un conjunto de componentes que ayudan a desarrollar sitios web más fácil y rápidamente.

Cuando estamos construyendo un sitio web, frecuentemente necesitas un conjunto de componentes similares: una manera de manejar la autenticación de usuarios (registrarse, iniciar sesión, cerrar sesión), un panel de administración para tu sitio web, formularios, una forma de subir archivos, etc. Hace tiempo varias personas notaron que los desarrolladores web enfrentan problemas similares cuando construyen un sitio nuevo, por eso juntaron cabezas y crearon frameworks (Django es uno de ellos) que te ofrecen componentes listos para usarse.

Los frameworks existen para evitar tener que reinventar la rueda y ayudar a aliviar la carga cuando se construye un sitio.

# ¿Por qué es necesario un framework?

Para entender para que es Django, necesitamos mirar más de cerca a los servidores.

Imagine un buzón (puerto) el cual es monitoreado por cartas entrantes (peticiones). Esto es realizado por un servidor web. El servidor web lee la carta, y envía una respuesta con una página web. Pero cuando quieres enviar algo, tienes que tener algún contenido. Y Django es algo que te ayuda a crear el contenido.

# ¿Qué sucede cuando alguien solicita una página web de tu servidor?

Cuando llega una petición a un servidor web, ésta es pasada a Django, el cual intenta averiguar lo que realmente es solicitado. Toma primero una dirección de página web y trata de averiguar qué hacer. Esta parte es realizada por el urlresolver de Django (ten en cuenta que la dirección de un sitio web es llamada URL - Uniform Resource Locator - así que el nombre urlresolver tiene sentido). Este no es muy inteligente - toma una lista de patrones y trata de encontrar la URL. Django comprueba los patrones de arriba hacia abajo y si algo coincide entonces Django le pasa la solicitud a la función asociada (que se llama vista).

Imagine a un cartero llevando una carta, él está caminando por la calle y comprueba cada número de casa con el que está en la carta. Si coincide, el cartero deja la carta allí. Así es como funciona el urlresolver.

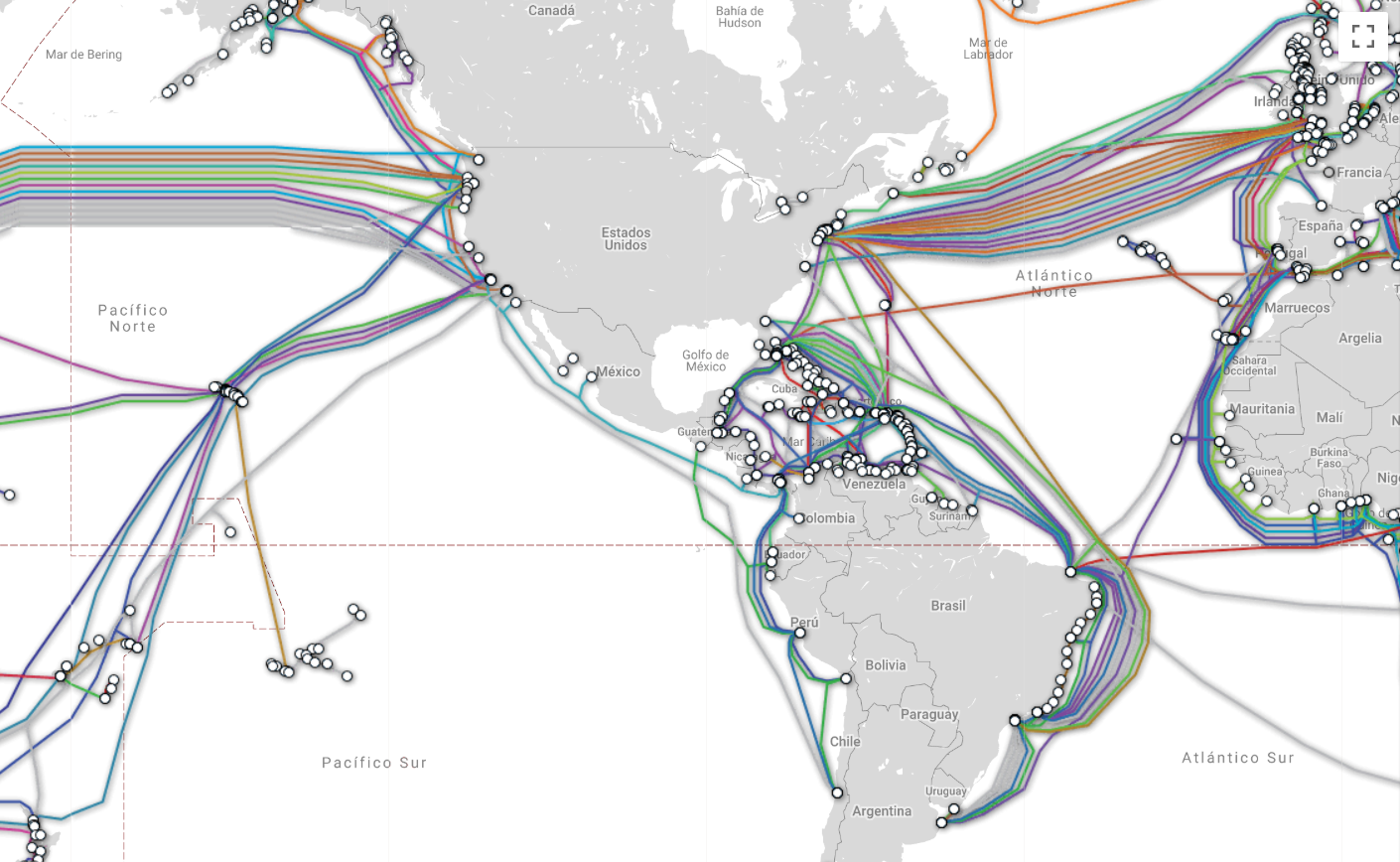
En la función de vista se hacen todas las cosas interesantes: podemos mirar a una base de datos para buscar alguna información. ¿Tal vez el usuario pidió cambiar algo en los datos? Como una carta diciendo "Por favor cambiar la descripción de mi trabajo." La vista puede comprobar si el remitente tiene permitido hacer eso, entonces actualizar la descripción del trabajo y devolver un mensaje: "¡hecho!". Entonces la vista genera una respuesta y Django puede enviarla al navegador del usuario.

# ¿Cómo funciona internet?

Lo primero que tenemos que entender es que un sitio web es un montón de archivos guardados en un disco duro. Al igual que tus películas, música o fotos. Sin embargo, los sitios web poseen una peculiaridad: ellos incluyen un código de computadoras llamado HTML.

Si no estás familiarizado con la programación, puede ser difícil de captar HTML al principio, pero tus navegadores web (como Chrome, Safari, Firefox, etc.) lo aman. Los navegadores web están diseñados para entender este código, seguir sus instrucciones y mostrar todos esos archivos de los cuales tu sitio web está hecho de la manera exacta como se quiere que se muestren. Como cualquier otro archivo, tenemos que guardar los archivos HTML en algún lugar de un disco duro. Para Internet, usamos unas computadoras especiales y poderosas llamadas servidores. Ellas no tienen una pantalla, mouse o teclado, debido a que su propósito es almacenar datos y servirlos. Por esa razón son llamados servidores -- porque ellos sirven los datos.

Entonces físicamente internet se vería más o menos (¡Cientos de miles de máquinas! Muchos, muchos kilómetros de cables alrededor del mundo) así:

  
Fuente: <http://submarinecablemap.com/>

Imagine que cuando escribe una URL (https://docs.djangoproject.com/es/2.0/), está enviando una carta que dice: "Queridos Djangoproject, me gustaría ver su sitio web docs.djangoproject.com/es/2.0/. Por favor, envíenmelo!".

Su carta va hacia la oficina de correo más cercana. Luego va a otra un poco más cercana de su destinatario, luego a otra y a otra hasta que es entregada en su destino. La única cosa diferente es que si tu envías cartas (paquetes de datos) con frecuencia al mismo lugar, cada carta puede pasar por oficinas de correos (routers) totalmente diferentes, dependiendo de cómo se distribuyen en cada oficina; es tan simple como eso. Enviar mensajes y esperar alguna respuesta. Por supuesto, en vez de papel usas bytes de datos.

# Antes de empezar a hablar exclusivamente de Django:

La línea de comandos o interfaz de línea de comandos  
Es una aplicación basada en texto para ver, manejar y manipular archivos en tu computadora (como por ejemplo el Explorador de Windows o Finder en Mac, pero sin la interfaz gráfica). Otros nombres para la línea de comandos son: cmd, CLI, símbolo del sistema, consola o terminal.

Cada sistema operativo tiene un conjunto diferente de comandos para la línea de comandos, aquí dejare una tabla de los comandos básicos a utilizar:

|  |  |  |  |
| --- | --- | --- | --- |
| Comando (Windows) | Comando (Mac OS / Linux) | Descripción | Ejemplo |
| exit | exit | Cierra la ventana | exit |
| cd | cd | Cambia el directorio | cd test |
| dir | ls | Lista directorios/archivos | dir |
| copy | cp | Copia de archivos | copy c:\test\test.txt c:\windows\test.txt |
| move | mv | Mueve archivos | move c:\test\test.txt c:\windows\test.txt |
| mkdir | mkdir | Crea un nuevo directorio | mkdir testdirectory |
| del | rm | Elimina archivos/directorios | del c:\test\test.txt |

Una referencia complete de los comandos, para todos los sistemas operativos, es: ss64.com

## Python

Es un lenguaje de programación muy popular que puede utilizarse para la creación de sitios web, juegos, software académico, gráficos y mucho, mucho más. Python se originó en la década de 1980 y su objetivo principal es ser legible por los seres humanos, por eso parece mucho más simple que otros lenguajes de programación, se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Para instalar Python en su computador: <https://www.python.org/downloads/>

### Variables:

La sintaxis para declarar que es: variable\_name = variable\_value si bien puedes nombrar las variables como desee (excepto algunas palabras clave reservadas), debería usar un esquema de nombres que tenga sentido intuitivo. El nombre de la variable debe proporcionar alguna indicación sobre cuáles son los valores que se le asignan. Por ejemplo, muchos programadores nuevos usan nombres de variables simples como X y Y intente no hacer esto.

### Tipos de datos incorporados:

Python tiene una serie de tipos de datos incorporados, como números (enteros, flotantes, números complejos), cadenas, listas, tuplas y diccionarios.

**Numbers**

Los números pueden ser enteros, flotantes, booleanos o complejos. Los primeros tres son los más importantes:

* Los Integers son números enteros - 1, 2, 22, 476,-99999
* Floats tienen puntos decimales - 1.0, 2.22, 22.098, 476.1,-99999.9
* Booleanos representan ya sea True o False (1 o 0). Representan datos que solo pueden ser una cosa u otra.

**Manipulando números**

Operadores:

>>> 2 + 3 # Adición

5

>>> num1 = 10

>>> num2 = 9.99

>>> num3 = num1 + num2

>>> num3

19.990000000000002

>>> 8 - 5 # Resta

3

>>> 2 \* 6 # Multiplicación

12

>>> 12 / 3 # división

4.0

>>> 7 % 3 # Módulo (devuelve el resto de la división)

1

>>> 3 \*\* 2 # Elevar

9

Operadores lógicos:

>>> 2 < 5

True

>>> 4 > 10

False

>>> 3 > = 3

True

>>>

>>> 5 == 6

False

>>> 6 ! = 9

True

Funciones:

Python le proporciona una serie de funciones integradas para manipular enteros. Estos están siempre disponibles. Tenga en cuenta que estas funciones se pueden usar en cualquier tipo de datos. También hay varios módulos disponibles en la Biblioteca estándar de Python, como math. Para usar las funciones asociadas a estos módulos, primero deberá importar el módulo.

float() - dado un número, esta función devuelve un float.

>>> float( 9 )

9.0

>>> float( -99999 )

-99999.0

int()- dado un numero de tipo float, esta función retorna un entero. Esta función NO redondea la entrada hacia arriba al entero más cercano - simplemente devuelve el numero antes del punto decimal. Entonces la entrada de 10.6 devuelve 10, NO 11. De manera similar, 3.25 devuelve 3.

>>> int( 10.6 )

10

>>> int( 3.25 )

3

Métodos de tipo de datos

Además de las funciones, hay varios métodos de tipo de datos asociados con cada tipo de número.

float.is\_integer() - prueba si un float puede ser de tipo Int.

>>> (10.0).is\_integer()

True

>>> (10.1).is\_integer()

False

**Strings**

Las cadenas de caracteres (Strings) son líneas de texto que se declaran con comillas simples o dobles:

>>> simple\_string = "hey!"

>>> simple\_string

'hey!'

>>> "hello world!"

'hello world!'

>>> escaped = 'can\'t'

>>> escaped

"can't"

>>> not\_escaped = "can't"

>>> not\_escaped

"can't"

**Manipulación de cadenas**

Operadores

Como en los números, puede concatenar cadenas (string concatenation):

>>> "happy" + " " + "birthday"

'happy birthday'

>>> "my name is " + "john"

'my name is john'

Funciones

len() - dada una cadena, esta función devuelve la longitud de la misma.

len(name\_string)

15

slice() - dado un valor de inicio y finalización, puede acceder a un conjunto de caracteres o a uno solo.

>>> print("Hello"[2])

l

>>> print("Hello"[3])

l

>>> print("Hello"[0])

H

>>> print("Hello"[0:2])

He

Métodos de tipos de datos

Solo se enunciaran algunos, para más información: <https://docs.python.org/3/library/stdtypes.html#string-methods>

string.capitalize() - devuelve la cadena con la primera letra en mayúscula.

>>> lower\_case\_string = "hernan"

>>> lower\_case\_string.capitalize()

'Hernan'

>>> ("empire").capitalize()

'Empire'

string.format() - Puede formatear fácilmente valores en cadenas.

>>> name = "Hernan Alvarez"

>>> greeting = "My name is {}".format(name)

>>> greeting

'My name is Hernan Alvarez'

>>>

string.strip() - devuelve la cadena con el espacio en blanco inicial y final eliminado.

>>> are\_you\_happy = " Yes "

>>> are\_you\_happy.strip()

'Yes'

**Listas**

Las listas (Lists), que se llaman matrices (Arrays) en casi todos los demás lenguajes de programación, agrupan varios tipos de datos en conjunto.

* create\_a\_list = []
* numbers\_list = [1, 2, 3]
* strings\_list = ["spam", "eggs", "cheese"]
* mixed\_list = ["Hello", [1, 2, 3], False]

Como puede ver arriba, las listas pueden contener cualquier tipo de datos, incluidas otras listas o nada en absoluto. Puede acceder a partes de listas como cadenas con índices. La sintaxis es la misma:

>>> numbers\_list[0]

1

>>> numbers\_list[0:1]

[1]

>>> numbers\_list[0:2]

[1, 2]

>>> numbers\_list = [ 1 , 2 , 3 , 200 ]

>>> new\_list = numbers\_list[0:3]

>>> new\_list

[1, 2, 3]

Fíjese que puede crear una nueva lista simplemente accediendo a los elementos, los índices negativos cuentan desde el último elemento:

>>> strings\_list = ["batman", "superman", "iron man"]

>>> strings\_list[-1]

'iron man'

Si anida una lista dentro de otra, puede acceder a la lista interna usando múltiples índices:

>>> mixed\_list = ["Hello World", [4, 5, 6], False]

>>> mixed\_list[1][2]

6

**Manipulando listas**

Operadores

Las listas se pueden concatenar:

>>> fruits = ["apples", "grapes", "oranges"]

>>> veggies = ["corn", "kale", "mushrooms"]

>>> grocery\_list = fruits + veggies

>>> grocery\_list

['apples', 'grapes', 'oranges', 'corn', 'kale', 'mushrooms']

Métodos de tipo de datos

Para mas información: <https://docs.python.org/3.7/tutorial/datastructures.html#more-on-lists>

list.append() - utilizado para agregar un artículo al final de una lista

>>> fruits = ["apples", "grapes", "oranges"]

>>> fruits.append("blueberries")

>>> fruits

['apples', 'grapes', 'oranges', 'blueberries']

list.sort() - ordenar esta lista

>>> fruits.sort()

>>> fruits

['apples', 'blueberries', 'grapes', 'oranges']

list.pop() - dado un valor de índice, eliminar un elemento de la lista

>>> numbers\_list = [1, 2, 3, 200]

>>> numbers\_list.pop(2)

3

>>> numbers\_list

[1, 2, 200]

**Tuples**

Las tuplas son similares a las listas, pero son inmutables, lo que significa que no se pueden cambiar después de la creación.

my\_tuple = ("Michael", "Hernan", 31, "Software Developer")

Al igual que una lista, también puede usar índices para acceder a diferentes elementos:

>>> my\_tuple = ("Michael", "Hernan", 31, "Software Developer")

>>> my\_tuple[1]

'Hernan'

**Manipulando tuplas**

Operadores

Sí, puedes agregar dos tuplas:

>>> first\_tuple = (1, 2)

>>> second\_tuple = (3, 4)

>>> third\_tuple = first\_tuple + second\_tuple

>>> third\_tuple

(1, 2, 3, 4)

Funciones

list() - utilizado para convertir una tupla en una lista

>>> first\_tuple = (1, 2)

>>> list(first\_tuple)

[1, 2]

**Diccionarios**

Los diccionarios son un tipo de matriz asociativa (un objeto) implementada usando tablas hash que contienen pares clave / valor. Ellos están desordenados.

my\_dict = {"Key 1": "Value 1", "name": "michael herman", "pi": 3.14}

Puede devolver el valor de una clave con la siguiente sintaxis:

>>> my\_dict = {"Key 1": "Value 1", "name": "michael herman", "pi": 3.14}

>>> my\_dict["name"]

'michael herman'

>>> my\_dict["Key 1"]

'Value 1'

Esto solo es un pequeño abre bocas de lo que es python, para más información puede remitirse a: <https://docs.python.org/3.7/>

## Entornos de trabajo virtualizados con Virtualenv

Aunque existen alternativas como Conda (un tutorial bastante bueno para poner en practica Django en un entorno virtual de Conda es: <https://www.instalardjango.com/>) para esta guía se utilizó Virutalenv.

Virtualenv es una herramienta para crear entornos de Python aislados, los problemas básicos que se aborda con este método de trabajo es de dependencias, versiones y permisos indirectos. Imagine que tiene una aplicación que necesita la versión 1 de LibFoo, pero otra aplicación requiere la versión 2. ¿Cómo puede usar ambas aplicaciones? Si instala todo en /usr/lib/python2.7/site-packages (o la ubicación estándar de su plataforma), es fácil terminar en una situación en la que involuntariamente actualice una aplicación que no debe actualizarse. Además, ¿qué sucede si no puede instalar paquetes en el site-packages del directorio global ? Por ejemplo, en un host compartido.

### Instalación y funcionamiento

Para instalar Virutalenv debemos tener previamente instalado el administrador de paquetes PIP y tener en modo administrador el símbolo del sistema:

\Documents> pip install virtualenv

Para crear y activar el entorno virtual:

\Documents> virtualenv vm

\Documents> cd vm

\Documents\vm> cd Scripts

\Documents\vm\Scripts> activate

(vm) \Documents\vm\Scripts>

# El patrón de diseño Model-Template-View

El patrón de diseño que conocemos normalmente es el MVC el cual ha existido como un concepto durante mucho tiempo, pero ha visto un crecimiento exponencial desde la llegada de Internet porque es la mejor forma de diseñar aplicaciones cliente-servidor. El patrón de diseño MVC es realmente simple de entender:

* El modelo (**M**) es un modelo o representación de sus datos. No son los datos reales, sino una interfaz para los datos. El modelo le permite extraer datos de su base de datos sin conocer las complejidades de la base de datos subyacente. El modelo generalmente también proporciona una capa de abstracción con su base de datos, de modo que puede usar el mismo modelo con múltiples bases de datos.
* La vista (**V**) es lo que ves. Es la capa de presentación para tu modelo. En su computadora, la vista es lo que ve en el navegador para una aplicación web o la interfaz de usuario para una aplicación de escritorio. La vista también proporciona una interfaz para recopilar la información del usuario.
* El controlador (**C**) controla el flujo de información entre el modelo y la vista. Utiliza la lógica programada para decidir qué información se extrae de la base de datos a través del modelo y qué información se pasa a la vista. También obtiene información del usuario a través de la vista e implementa la lógica empresarial: ya sea al cambiar la vista, o al modificar datos a través del modelo, o ambos.

Django sigue de cerca el patrón MVC, sin embargo, utiliza su propia lógica en la implementación. Debido a que la "C" es manejada por el propio framework y la mayor parte de la emoción en Django ocurre en los modelos, plantillas y vistas, Django a menudo se conoce como un marco de MTV. El patrón de desarrollo MTV se puede ver:

* **M** significa "Modelo", la capa de acceso a datos. Esta capa contiene todo sobre los datos: cómo acceder a él, cómo validarlo, qué comportamientos tiene y las relaciones entre los datos.
* **T** significa "Template", la capa de presentación. Esta capa contiene decisiones relacionadas con la presentación: cómo se debe mostrar algo en una página web u otro tipo de documento.
* **V** significa "View", la capa de lógica de negocios. Esta capa contiene la lógica que accede al modelo y difiere a la(s) plantilla(s) apropiada(s). Se puede hacer símil con que es el puente entre los modelos y las plantillas.

# Instalar Django e iniciar un proyecto

Ya que tenemos Python y estamos ejecutando un entorno virtual, instalar Django es muy fácil, simplemente escriba el comando:

(vm) \Documents> pip install django

Esto le indicará PIP que instale Django en su entorno virtual.

Un proyecto es una colección de configuraciones para una instancia de Django. Al crear un proyecto se genera una serie de archivos y ficheros, desde la línea de comandos, cambie a un directorio donde le gustaría almacenar su código, luego, ejecute el siguiente comando:

(vm) \Documents> django-admin startproject mysite

Esto creará un directorio mysite en su directorio actual, el cual tiene la forma de:

mysite/

manage.py

mysite/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

* El directorio raíz externo mysite/ solo es un contenedor de su proyecto. Su nombre no es relevante para Django; puede cambiarle el nombre a lo que quiera.
* manage.py: Una utilidad de la línea de comandos que le permite interactuar con este proyecto Django de diferentes formas.
* En interior del directorio mysite/ es el propio paquete de Python para su proyecto. Su nombre es el nombre del paquete de Python que usted tendrá que utilizar para importar todo dentro de este (por ejemplo, mysite.urls).
* mysite/\_\_init\_\_.py: Un archivo vacío que le dice a Python que este directorio debe considerarse un paquete de Python.
* mysite/settings.py: archivo de configuración para este proyecto Django.
* mysite/urls.py: Las declaraciones URL para este proyecto Django; una “tabla de contenidos” de su sitio basado en Django.
* mysite/wsgi.py: Un punto de entrada para que los servidores web compatibles con WSGI puedan servir su proyecto.

Para comprobar que el proyecto Django funciona correctamente, desde la terminal, entramos al directorio y escribimos:

(vm) \Documents\mysite> python manage.py runserver

Verá la siguiente salida en la línea de comandos:

Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.

Run 'python manage.py migrate' to apply them.

Sep 20, 2018 - 15:50:53

Django version 2.0, using settings 'mysite.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CONTROL-C.

Usted ha iniciado el servidor en desarrollo de Django, un servidor web ligero escrito puramente en Python. Está incluido con Django para que pueda desarrollar cosas rápidamente sin tener que lidiar con la configuración de un servidor en producción, como Apache, hasta que esté listo para la producción. Ahora es un buen momento para tener en cuenta que: no debe utilizar este servidor en algo parecido a un entorno de producción.