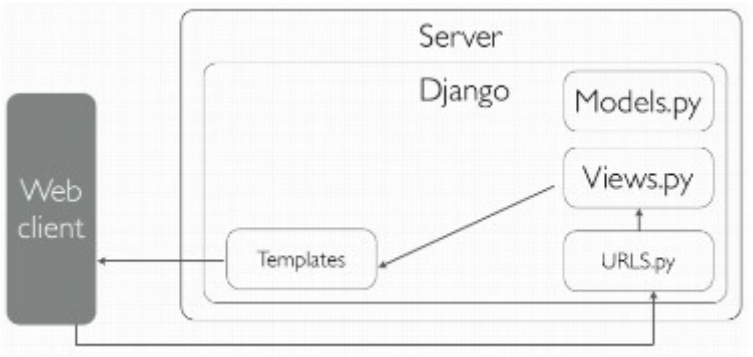
# Django

### *Que es:*

Django es un *framework* web de tercera generación, de código abierto, **escrito en Python**, lo que hace que sea mucho más ágil para programar que otros *frameworks* escritos en PHP o JSP.

## Flujo de trabajo de Django (arquitectura)




*** Web Client:** es el navegador (FireFox - Chrome - Opera) → se conecta al *web server* → el *server* se Conecta a Django → controladores : vistas → modelo. Cuando ese proceso termina pintas la vista que renderiza a  html, para luego ser interpretado por los navegadores (HTML - Javascript).

*** Server:** puede ser Apache, por ejemplo.

---

*** Django:** El flujo de trabajo del framework es:

**URLS.py** →  Escucha al *Web client* y retorna la vista (en *Views.py*) asociada a ella. En caso de no haber vista asociada a la URL, retorna la vista por defecto.

**views.py** →  Son funciones que reciben una solicitud y responden con un contenido de una página, una imagen o un error en el servidor. Estas funciones las puedes definir en tu proyecto de Django, en el archivo views.py, y contendrán toda la lógica necesaria para dar respuesta a una solicitud.

**Models.py** Un modelo es la representación de los datos de nuestra aplicación. Contiene los campos básicos y el comportamiento de los datos que serán almacenados. Por lo general, cada modelo se convierte en una tabla de la base de datos.

**Templates** Reciben información de la vista para mostrar el resultado de una forma mucho más organizada. El sistema de templates recibe los datos que mandas desde la vista y luego, para

insertar estos datos, sólo debes ponerlo entre llaves de la siguiente forma: {{ nombre }}.

# ¿Cómo iniciar tu proyecto?

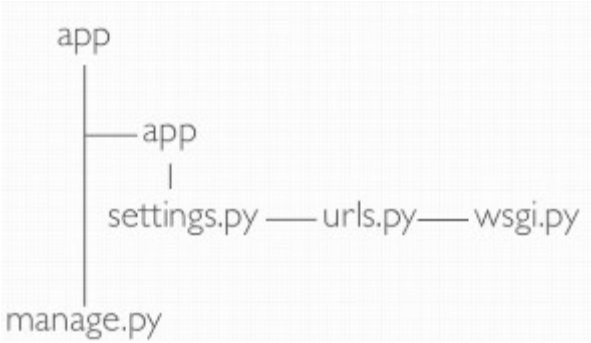
Para arrancar con tu proyecto ejecutas en terminal o cmd:

```
$ django-admin.py startproject _NombreDeTuProyecto_
```

Si tienes problemas en Windows puedes usar:

```
$ C:\[directorio de python]\Scripts\django-admin.py startproject
_NombreDeTuProyecto_
```

Estos comandos te generan una carpeta en el lugar desde donde ejecutes la línea con la siguiente estructura:




A diferencia de otros *frameworks* Django es súper minimalista porque encapsula todo lo que necesita en estos pocos archivos **.py**.

# Configuraciones generales

*** manage.py** es tu brazo derecho, es el que te permite ejecutar líneas de comando. Si ejecutas **python manage.py** en la consola podrás descubrir los múltiples comandos que tiene. Dentro de ellos está el cambio de contraseña del súper usuario, administrar la base de datos, crear aplicaciones, entre otros.

*** wsgi.py** es la configuración que se utiliza para conectarnos al sistema en producción (Apache - nginx).

* **Para ejecutar el proyecto RUNSERVER + el archivo manage.py**:

```
$ python manage.py runserver
```

Así levantas el servidor en la ruta http://127.0.0.1:8000/ o http://localhost:8000.

Aparecerá un mensaje diciendo: **It Works!**. Después desde la consola puedes supervisar todo lo que pasa con tu servidor.

# Configuración de Django: settings.py

Defines cómo va a funcionar tu proyecto (configuración del servidor, aplicaciones, lenguaje, etc.) en el **archivo settings.py**. Este viene por defecto con una configuración básica (lo mínimo que necesitas y hasta más) para que funcione tu *app* Django, pero le puedes agregar lo que quieras para que tu aplicación se ajuste a lo que necesitas.

*** ADMINS**: administradores del servidor a los que les mandan correos si tienes algún problema.

*** DATABASES**: configuración a la conexión con el motor de base de datos.

*** ALLOWED_HOSTS:** cuáles *hosts* tienen permitido acceder al sitio o tener solicitudes.

*** LANGUAGE_CODE**: el lenguaje en el cual está la interficie de Django. Django está traducido a muchos lenguajes, en esta variable lo puedes cambiar.

*** USE_L10N**: sirve para manejar el formato calendario (*dates*).

*** MEDIA_ROOT**: directorio de almacenamiento toda la información que genera el usuario.

*** STATIC_ROOT**: directorio de almacenamiento de los archivos estáticos que pueden estar enlazados a los *templates* como archivos de formato CSS, Javascript, imágenes, etc.

*** SECRET_KEY**: es la llave que utilizas para codificar (*hashear*) las contraseñas o sesiones (Recordad que por este motivo, no se suele subir a Github este archivo de la versión final del server).

*** INSTALLED_APPS**: aquí activas las aplicaciones de tu proyecto, tanto las que has estado creando como los paquetes externos que necesitas.

**Nota:** existen múltiples paquetes que pueden resolver el mismo problema podéis elegir libremente cual usar… o crear una de nueva.

En https://pypi.python.org/pypi y https://www.djangopackages.com/ tienes un directorio de paquetes que puedes usar en tus proyectos.

# Aplicaciones en Django

Las aplicaciones son carpetas con código (módulos) que alimentan tu proyecto. Estas aplicaciones pueden ser empacadas y distribuidas para que las uses en otros proyectos.

* Para crear una aplicación en el proyecto debes ir al directorio del proyecto y ejecutar:

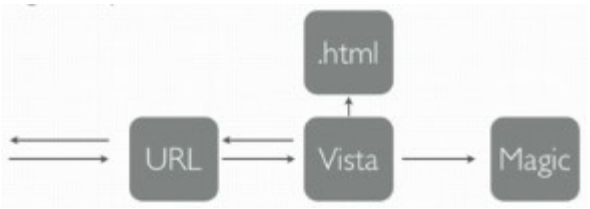
```
$ python manage.py startapp _NombreDeLaApp_
```

Esto te crea una nueva carpeta dentro de tu proyecto con tres archivos **.py** que son los que manejan las funcionalidades de la *app*.

* Al crear tu *app* debes activarla en **INSTALLED_APPS** de **settings.py**:

```
INSTALLED_APPS = (
    '_NombreDeLaApp_',
)
```

# URLs - Vistas




Las *urls* son las que reciben las peticiones del *web client* y se resuelven con expresiones regulares. Estas *urls* apuntan a vistas en las que se define el comportamiento de la aplicación.

**¿Cómo puedes crear una url? → en el archivo urls.py dentro de urlpatterns**.

**Ejemplo**:

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

**r'^$'** es la raíz del sitio que llama a **app.views.home**, que es la vista que define el comportamiento de esa ruta (como enviar un template .html). Luego en **name** pones el nombre de la url, **en este caso home**.

Las *urls* pueden recibir parámetros en forma de expresiones regulares

**Ejemplo**:

```
    url(r'^post/(\d+)$', 'app.views.post', name='post')
```

En este caso la *url* recibe un dígito → \d+

Las *views* siempre reciben un *HttpRequest* y devuelven un objeto *HttpResponse*. **Para usar HttpResponse en views.py debes importar HttpResponse**:

```
>>> from django.http import HttpResponse
```

Ejemplo:

```
>>> def home(request):
…   return HttpResponse("Hola Mundo este es el Home")
```

*request* es el *HttpRequest* y *home* devuelve como *HttpResponse* "**Hola Mundo este es el home**".

Las vistas pueden recibir múltiples parámetros, estos llegan en forma de un *string* (así se manden como un dígito).

**Ejemplo**:

```
>>> def post(request,id_post):
…     if int(id_post) > 10 :
…         return HttpResponse("Este es mayor que 10 : %s" % id_post)
…     else:
…         return HttpResponse("Este es menor que 10 : %s" % id_post)
```

Dado que las vistas son funciones, pueden contener lo que quieras: en este caso estructuras de control *if - else* para analizar el parámetro *id_post* que mandaste en la url.