# Django

venv\Scripts\activate

django-admin startproject myproject

python manage.py runserver

django-admin startapp app\_name

cd ..

cd file

## Fundamentos:

**MODELS:** a representation of your applications database layout.

All models are subclass of the **django.db.models.Model** class. Each class will be transformed into database tables. Each field is represented by instances of **django.db.models.Field** subclasses (built-in Django core) and will be translated into database columns.

The fields CharField, DateTimeField, etc., are all subclasses of **django.db.models.Field** and they come included in the Django core – ready to be used.

Every model comes with a special attribute, the MODEL MANAGE. You can access it via the Python attribute objects

board = Board**.**objects**.**create(name**=**'Python', description**=**'General discussion about Python.')

Board**.**objects**.**all() : esto retorna una lista QuerySet

django\_board **=** Board**.**objects**.**get(id**=**1) : retorna el objeto q cumple

Fields:

Charfield: we should always define max\_length

DateTimeField: auto\_now\_add setea el tiempo del momento

ForeignKey: crea relaciones entre modelos a nivel de base de datos. Espera un parámetro posicional con la referencia del modelo relacionado.Tiene el parámetro obligatorio, on\_delete, que hay que definir que ocurre cuando el objeto se elimina; se usa harto models.CASCADE. Related\_name parámetro es usado para crear un relación inversa donde las instancias de la otra clase podrán tener una lista de las instancias quienes le pertenecen. La relación inversa se hace automáticamente, pero es bueno poner nombre. Si ponemo un ‘+’ en related\_name, le estamo diciendo a django q no necesitamos este tipo de relación.

Si no epecificamos primary keys/ids, django los hara automáticamente.

| **Operation** | **Code sample** |
| --- | --- |
| Create an object without saving | board = Board() |
| Save an object (create or update) | board.save() |
| Create and save an object in the database | Board.objects.create(name='...', description='...') |
| List all objects | Board.objects.all() |
| Get a single object, identified by a field | Board.objects.get(id=1) |

**Migrating:** decirle a django que cree la base de datos.

python manage.py makemigrations

python manage.py migrate

**Experimenting**

python manage.py shell

exit()

**VIEWS:** recive un httpRequest y debe entregar un HttpResponse. Se puede utilizar render para interactuar con el html.

**TEMPLATE:** se encargan de renderizar. Del html.

Hay que crear una carpeta llamada templates/

{% for ... in ... %} and {{ variable }}. They are part of the Django Template Language. The {{variable}} renders the name of the board in the html template, generating a dynamic html document.

Open the **settings.py** inside the **myproject** directory and search for the TEMPLATES variable and set the DIRS key to os.path.join(BASE\_DIR, 'templates'):

TEMPLATES **=** [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [

os**.**path**.**join(BASE\_DIR, 'templates')

],

**TESTING:**