Installl django = pip3 install django ==3.0.3   
<https://docs.djangoproject.com/en/3.2/howto/windows/>

Créate a Project:  
django-admin startproject projectname

To run the project:  
python manage.py runserver

Django apps = components in Django projects (features)  
A project may have 1 or many apps

Graphical user interface, text, application

Description automatically generated

**To create the app**

Python manage.py startapp appname

**To add the app to the project**go to the main folder that was first created when starting everything about the project, look for SETTINGS.PY > INSTALLED\_APPS > agregar al final el nombre del app ‘nombredeapp’, example: 'adoptions',

**Pieces of an app**

|  |  |
| --- | --- |
| FILE OR FOLDER |  |
| Apps.py | Settings specifics for the app |
| Models.py | Data layer, to construct data schema and queries |
| Admin.py | Administrative interface related to the app to edit it and manage it |
| Urls.py | URL routing, specific to this app |
| Views.py | Logic and control flow handling request |
| Tests.py | Unit tests |
| Migrations/ | Files to migrate the database |

**STRUCTURE IS DONE AT THIS POINT!!!**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**MVC ARCHITECTURE** (underlying architecture of the framework, conceptual structure)

Model-view-controller

A link = a request [www.example/example1](http://www.example/example1) the request of the example1 page

|  |  |
| --- | --- |
| Name | About |
| URL patterns | Decide what URL patterns to decide which view to pass the request to for handling |
| Views | Logic, control flow portion, takes HTTP request and returns HTTP response |
| Models | Queries from database  Create the data layer  Define database structure  Allow us to query the database  MODELS AS SPREADSHEETS:   * Model = a table in a spreadsheet * Field = a column in the spr. * Record = a row in the spr. |
| Templates | Views, HTML will look like |

Diagram

Description automatically generated

Diagram

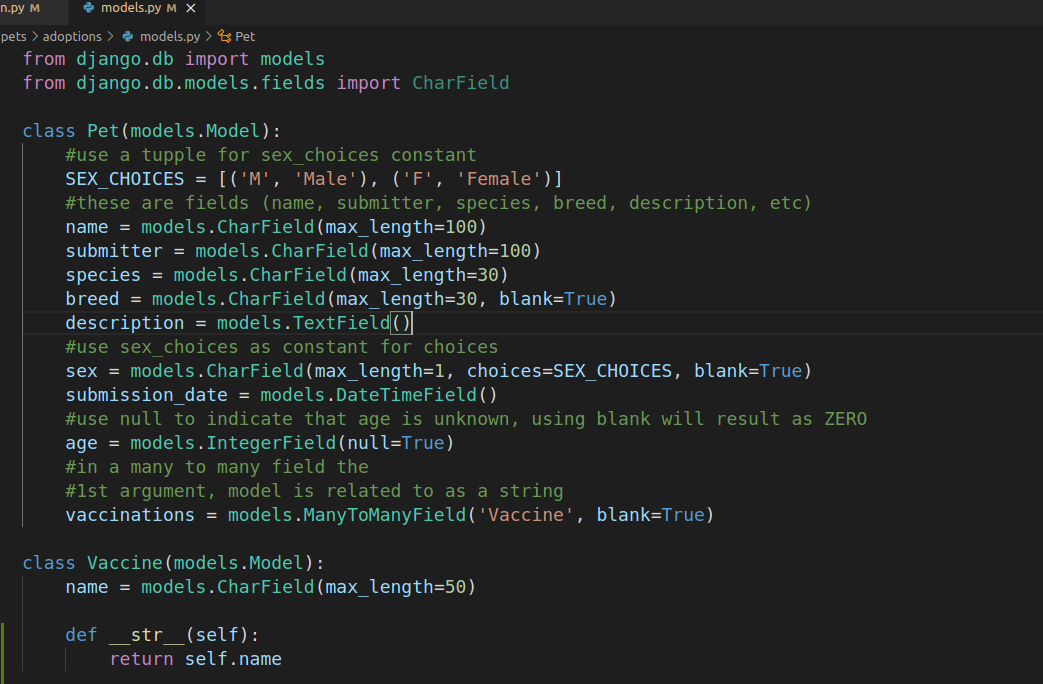
Description automatically generated

Field Types: Textual Data (email, text, characters)  
Field Types: Numeric Data (integers, decimals)  
Field Types: MIscellaneous Data (boolean, dateTIme)  
Field Types: Relational Data (foreignkey, manytomanyfield)  
FIELDS contain attributes)   
 blank(attribute)=true ==not required)  
 null(attribute) (no data) |   
 choices(atribute) (set of choices)

<https://docs.djangoproject.com/en/3.2/ref/models/fields/>

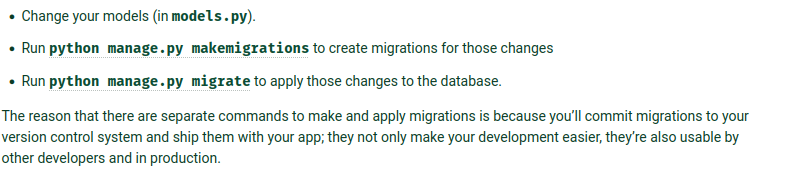
**CREAR EL MODELO DE LOS DATOS QUE LLEVARÁ**

model = python class that can be saved into a database



MIGRATIONS – Generate scripts to change the DB structure  
Migrates the models to the DB  
These are needed when

Adding a model  
Adding a field  
Removing a field  
Changing a field



INITIAL MIGRATION – 1st migration created for a new Django app will create tables for the models that are defined

(to make the initial migration)

**python manage.py makemigrations** – will created the models, if it is the first time it will be the Initial migration

-there are default apps when making the initial migration, they com with models and migrations

-after MAKINGS MIGRATIONS you need to MIGRATE

-it is good to make migrations any time a change is made to a model / or add a new model

**python manage.py migrate-** this will apply the migrations

**python manage.py showmigrations-** this will show the migrations that has been made  
[ ] 0001 square brackets with empty space indicates that those migrations have not yet been applied  
[X] 0001 square brackets with X indicates that those migrations have been applied

To inspect the SQLite databases to see structure and content <https://sqlitebrowser.org/>   
The default DB is db.sqlite3 in the main folder

Load the data with a script---

**python3 manage.py data\_to\_be\_loaded.py**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DJANGO ADMIN**

Project folder > app folder > admin.py

1. Import the model
2. Make an admin interface for out pet model (create the class) (pet admin)
3. Register the class with the admin to tell it which model its associated with (@adming.register) (el modelo/modelos en admin.py)

Text

Description automatically generated

1. Create super user – **python manage.py createsuperuser**  
   4.1 add username, email isn’t required, add password
2. Run server (**python manage.py runserver 7000**)
3. Localhost:7000/admin and login in

(los detalles de los objetos se verán solo como objetos en lugar de listar el nombre correcto)

El list\_display attribute will define the fields that will display in the listing screen

A screenshot of a computer

Description automatically generated

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**URL PATTERNS**

Diagram

Description automatically generated

Also called URL confs…

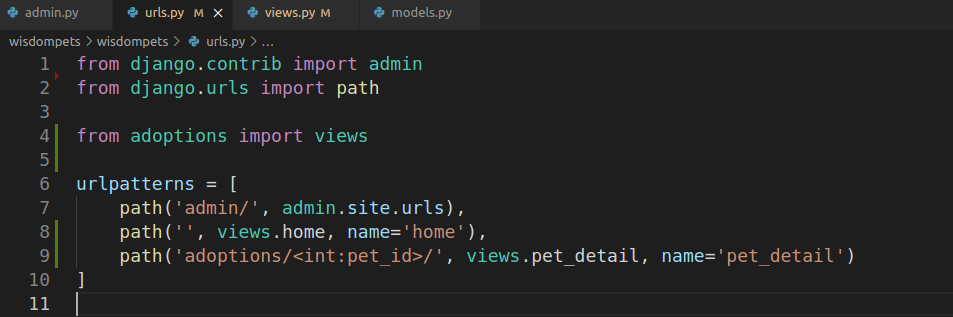
add the URLS in urls.py at the main folder

path(‘hola’, views.hola, name=’hola)

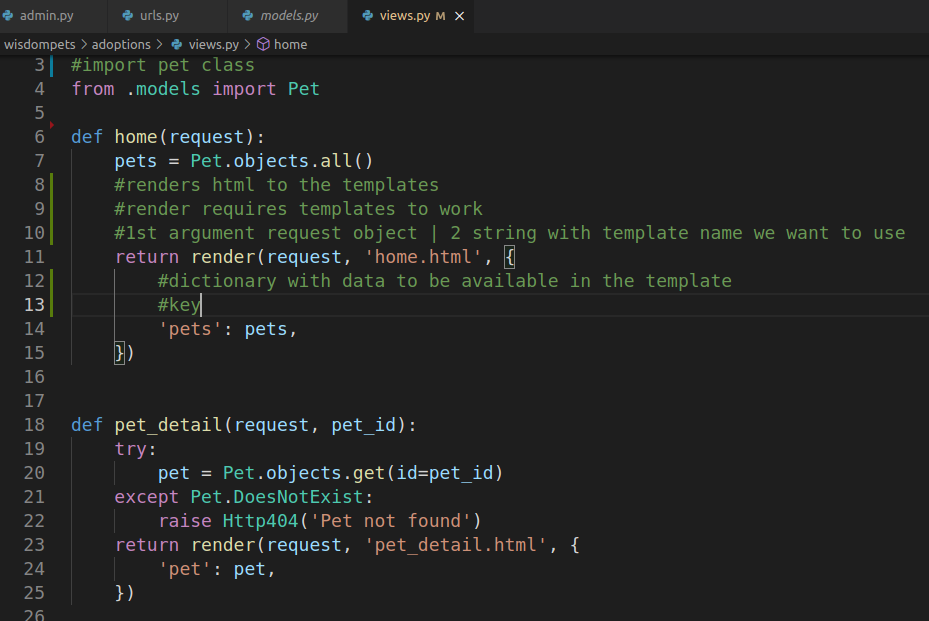
esto de arriba, si alguien escribe <http://misitio/hola> … el usuario será llevado al view que se llame hola, por eso en el ejemplo de abajo, al dejar el primer parámetro como una empty string lo lleval al view del home (ejemplo de la imagen)

si alguien deja empty string irá al folder jobs, archivo views, función home

path('', jobs.views.home, name='home'),



add the views at the app we want folder views.py  
the Pet comes from models.py



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DJANGO TEMPLATES**

Template has 3 pieces:

1. variables value **{{variable}} | <h3> {{pet.name}}</h3>**

2. template tag **{% tag %}** (for loops, ifs, structural elements)

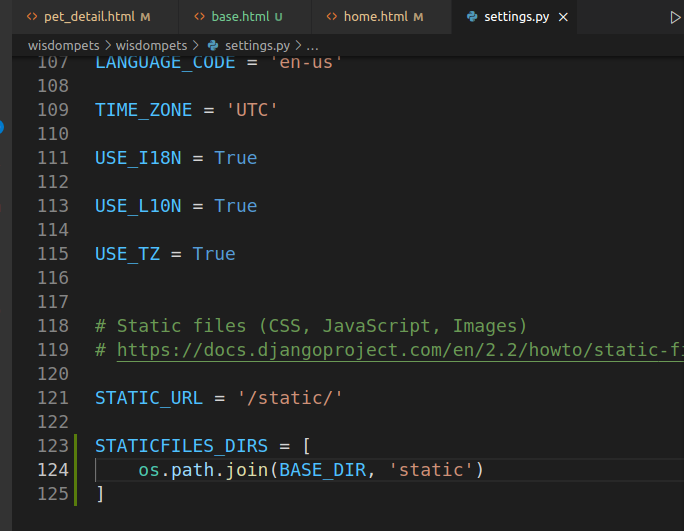
{% for pet in pets %}  
 <li>{{pet.name}}</li>  
{% endfor %}

3. **{{template | filter}}** pipe character to use a template filter (take string as input and return a string as output) **<h3> {{pet.name|capfirst}}</h3>**

To add **css and JS,** in Django they are called “STATIC” files, a STATIC files folder should be inside the project folder

add the path to the settings.py file that’s inside the app folder

add STATICFILES\_DIRS = [ os.path.join(BASE\_DIR, ‘static’)]

to make possible to load the assets, **add {%load static%}** at first line of html file

create the STATIC folder in the app folder

STATIC\_URL = '/static/'

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')

you can also use   
STATIC\_ROOT = [

os.path.join(BASE\_DIR, 'static')

]

**USING POSTGRESQL:**

se puede conectar al postgresql, hay que instalarlo

crear DB <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04-es>

psql → linea de comandos postgresql  
\q → salir de la linea de comandos

sudo -u postgres psql – to run postgres command line

sudo -u postgres createuser --interactive → crear un nuevo usuario

\password postgres → for the postgres of the project and password

user for portfolio project = postgres

password for portfolio project = django1234  
\conninfo (to know what port you are using)

sudo -u postgres createdb dbname → create database

CREATE DATABASE portfoliodb; //last word is the name for the DB

app folder name > settings.py > search for DATABASES line

**DEPLOY TO HEROKU**

sudo snap install heroku –classic

heroku login

virtualenv venv → for the virtual enviroment to install the following

source venv/bin/activate → activate the virtual environment to install

pip install django pillow psycopg2-binary gunicorn → after installation create the next file below

pip freeze > requirements.txt

while in the virtual environment type → deactivate (to quit the venv)

CREATE HEROKU APP (a git repository is needed too)

git add -A

git commit – m “message”

heroku create → to create app

git push heroku main

edit settings.py DEBUG = False | ALLOWED\_HOST = [‘OUR HOST NAME’]