

Despliegue del modelo en django

1. Ubíquese en la carpeta del proyecto (PI_MLProject), active el ambiente virtual

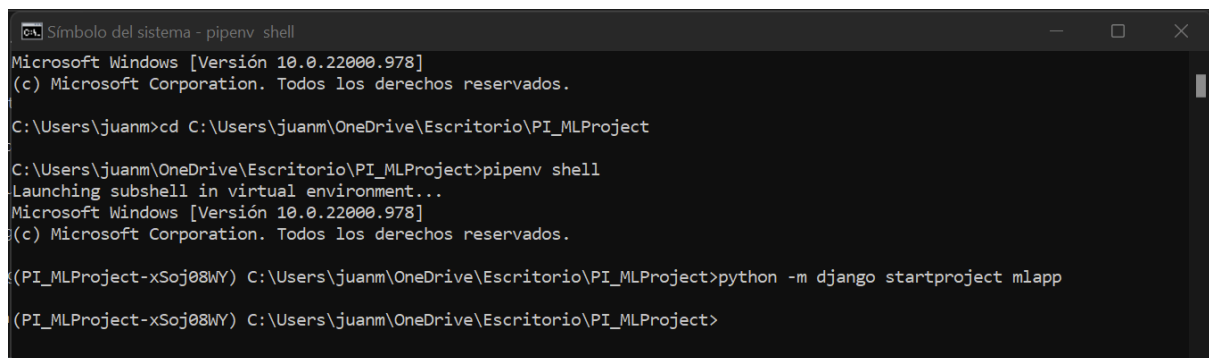
```
cd C:\Users\juanm\OneDrive\Escritorio\PI_MLProject
```

```
pipenv shell
```

2. Ejecute el siguiente comando:

```
python -m django startproject project_name
```

donde **project_name** debe ser el nombre del proyecto de django, en este caso **mlapp**



```
Símbolo del sistema - pipenv shell
Microsoft Windows [Versión 10.0.22000.978]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\juanm>cd C:\Users\juanm\OneDrive\Escritorio\PI_MLProject

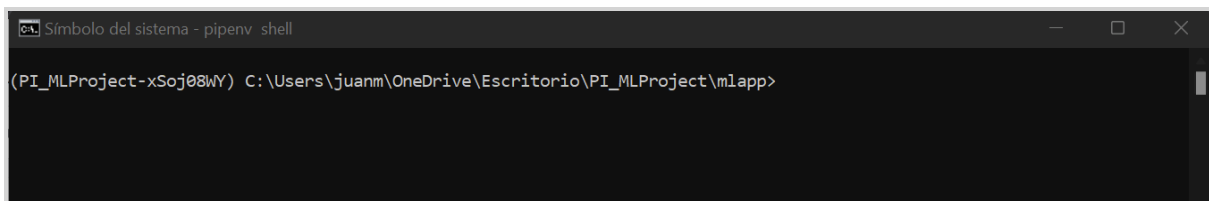
C:\Users\juanm\OneDrive\Escritorio\PI_MLProject>pipenv shell
Launching subshell in virtual environment...
Microsoft Windows [Versión 10.0.22000.978]
(c) Microsoft Corporation. Todos los derechos reservados.

(PI_MLProject-xSoj08wY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject>python -m django startproject mlapp

(PI_MLProject-xSoj08wY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject>
```

3. Acceda a la carpeta del proyecto que acaba de crear

```
cd mlapp
```



```
Símbolo del sistema - pipenv shell

(PI_MLProject-xSoj08wY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlapp>
```

4. Ejecute el servidor local web de django

```
python manage.py runserver
```

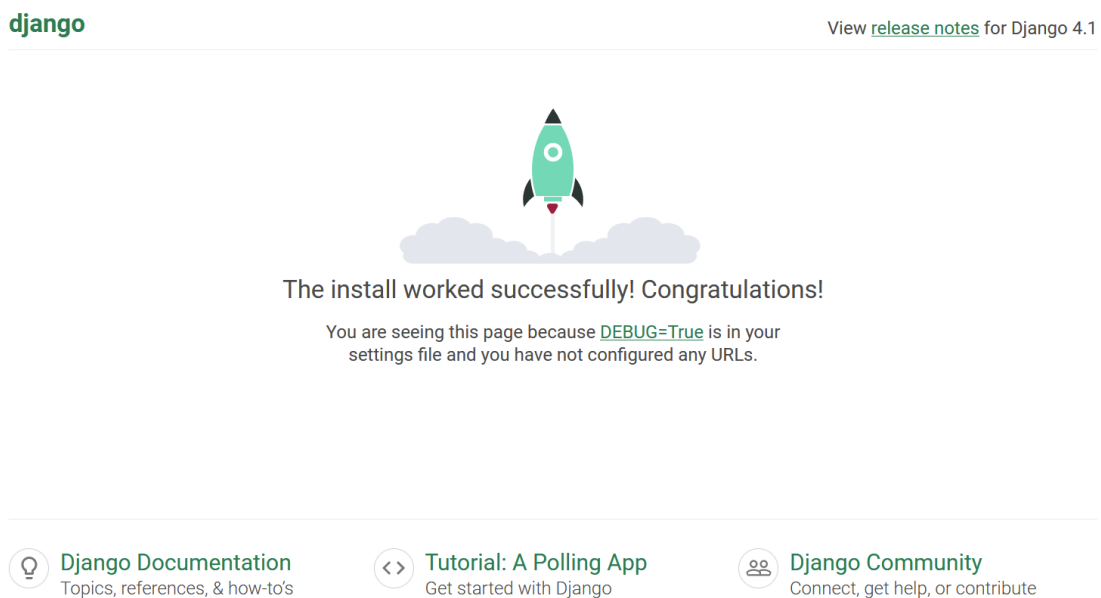
```
Símbolo del sistema - pipenv shell - python manage.py runserver

(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlapp>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

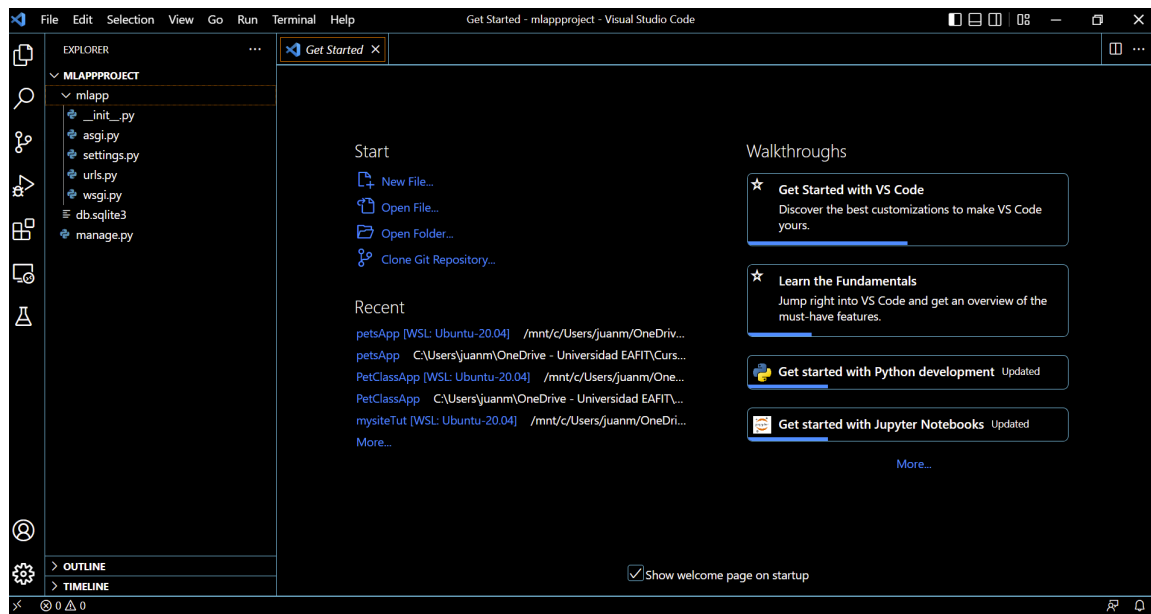
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 19, 2022 - 08:59:58
Django version 4.1.1, using settings 'mlapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

5. En el navegador acceda a la dirección <http://127.0.0.1:8000/>. Deberá observar un mensaje que indica que todo está funcionando correctamente.



6. Detenga el servidor desde la terminal presionando las teclas **CTRL** y **C** al mismo tiempo.
7. Cambie el nombre de la carpeta del proyecto, agregando la palabra project al final, así: **mlappproject**
8. Abra la carpeta del proyecto con un editor de código (Visual Studio Code, Sublime, PyCharm).

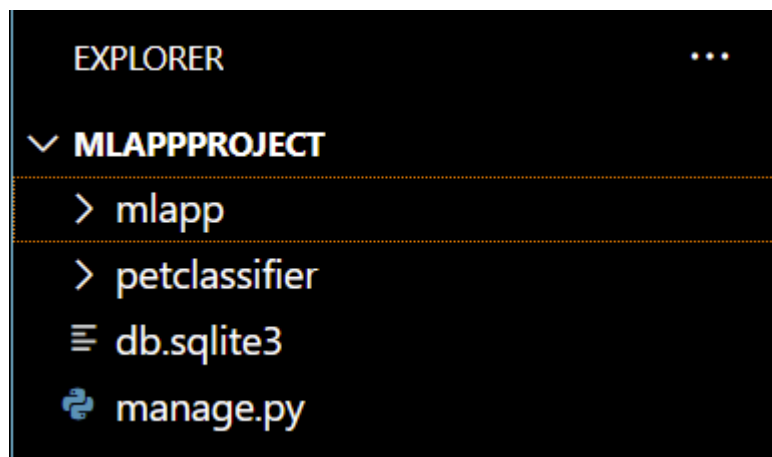


Creación de la aplicación petclasiffier

1. Desde la terminal, ubíquese en la carpeta del proyecto (No en la carpeta interna) y ejecute el comando:

```
python manage.py startapp petclassiffier
```

y verifique que se creo la carpeta petclassifier dentro de la carpeta del proyecto



2. Agregue la app al archivo **settings.py** de **mlapp**, en la sección **INSTALLED_APPS**

```
settings.py X
mlapp > settings.py > ...
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'petclassifier',
41 ]
```

3. En la terminal ejecute: `python manage.py migrate`

```
Símbolo del sistema - pipenv shell
(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>
```

4. Ejecute nuevamente el servidor desde la terminal, ingrese a la dirección del servidor local y verifique que aparece el mensaje que indica que todo está funcionando correctamente.
5. En la carpeta de la app (petclassifier) cree una carpeta llamada **templates** y dentro de esta cree un archivo llamado **home.html**. En este archivo agregue el código HTML de la página principal.




```

<head>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.m
in.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoq
y12QvZ6jIW3" crossorigin="anonymous">
</head>


<h1> Welcome to the PetClassifier App </h1>

```

6. En el archivo **views.py** de **petclassifier** agregue la función home

 settings.py	 views.py ●	 home.html
---	---	---

```

petclassifier >  views.py > ...
1  from django.shortcuts import render
2
3
4  # Create your views here.
5
6  def home(request):
7      |   return render(request, 'home.html')
8
9

```

7. En el archivo **urls.py** de **mlapp** agregue una ruta para la página principal de la app, que invoque la función home de **views.py**

Uso de modelos

1. Dado que los modelos permiten definir las características de los datos en la base de datos, utilizaremos un modelo (de datos) para almacenar los modelos entrenados (aprendizaje de máquina). Agregue el modelo **mlModels** en el archivo **models.py**

```
settings.py  views.py  urls.py  home.html
mlapp > urls.py > ...
1  """mlapp URL Configuration
2
3  The `urlpatterns` list routes URLs to views. For more information please see:
4  |   https://docs.djangoproject.com/en/4.1/topics/http/urls/
5  Examples:
6  Function views
7  |   1. Add an import:  from my_app import views
8  |   2. Add a URL to urlpatterns:  path('', views.home, name='home')
9  Class-based views
10 |   1. Add an import:  from other_app.views import Home
11 |   2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13 |   1. Import the include() function: from django.urls import include, path
14 |   2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from petclassifier import views as petClassifierViews
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', petClassifierViews.home)
23 ]
24
```

8. En el navegador ingrese al enlace <http://localhost:8000/> y visualizará la página principal



9. En la terminal, ejecute

```
python manage.py makemigrations
python manage.py migrate
```

```
Símbolo del sistema - pipenv shell

(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>python manage.py makemigrations
Migrations for 'petclassifier':
  petclassifier\migrations\0001_initial.py
    - Create model mlModels

(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, petclassifier, sessions
Running migrations:
  Applying petclassifier.0001_initial... OK

(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>
```

10. En la terminal ejecute

```
python manage.py createsuperuser
```

Ingresa el nombre de usuario, el correo y la contraseña con el cual se quiere registrar en la interfaz de administrador. Esta interfaz permite manipular los datos de la base de datos.

```
Símbolo del sistema - pipenv shell

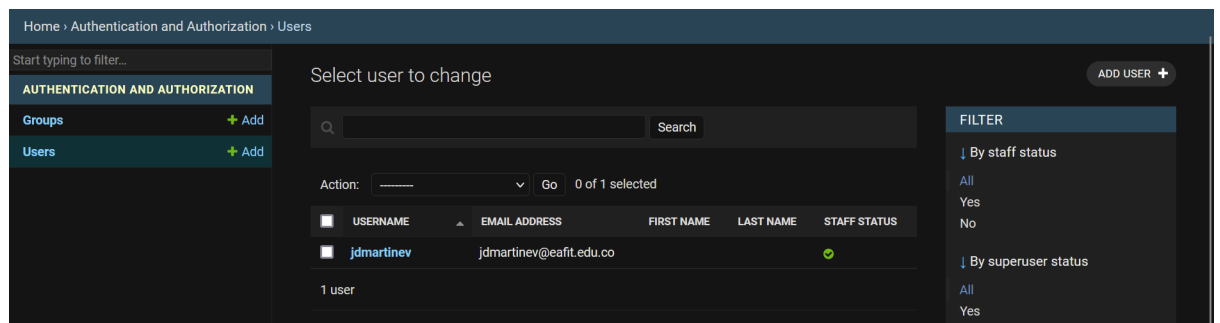
(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>python manage.py createsuperuser
Username (leave blank to use 'juanm'): jdmartinev
Email address: jdmartinev@eafit.edu.co
Password:
Password (again):
Superuser created successfully.

(PI_MLProject-xSoj08WY) C:\Users\juanm\OneDrive\Escritorio\PI_MLProject\mlappproject>
```





11. Ejecute el servidor desde la terminal


```
python manage.py runserver
```

e ingrese en el navegador a <http://localhost:8000/admin> e ingrese las credenciales de administrador que creó previamente. En **users** verifique que se encuentra el usuario que acaba de crear.









12. Agregue el modelo **mlModels** a **admin**. Para esto modifique el archivo **admin.py** de **petclassifier**

 settings.py	 views.py	 urls.py	 models.py
---	--	---	---

```
petclassifier >  admin.py
1  from django.contrib import admin
2  from .models import mlModels
3
4  # Register your models here.
5  admin.site.register(mlModels)
6
7
8
```

13. Configure donde se almacenarán los mlModels. En **settings.py** de **mlapp** agregue MEDIA_ROOT y MEDIA_URL

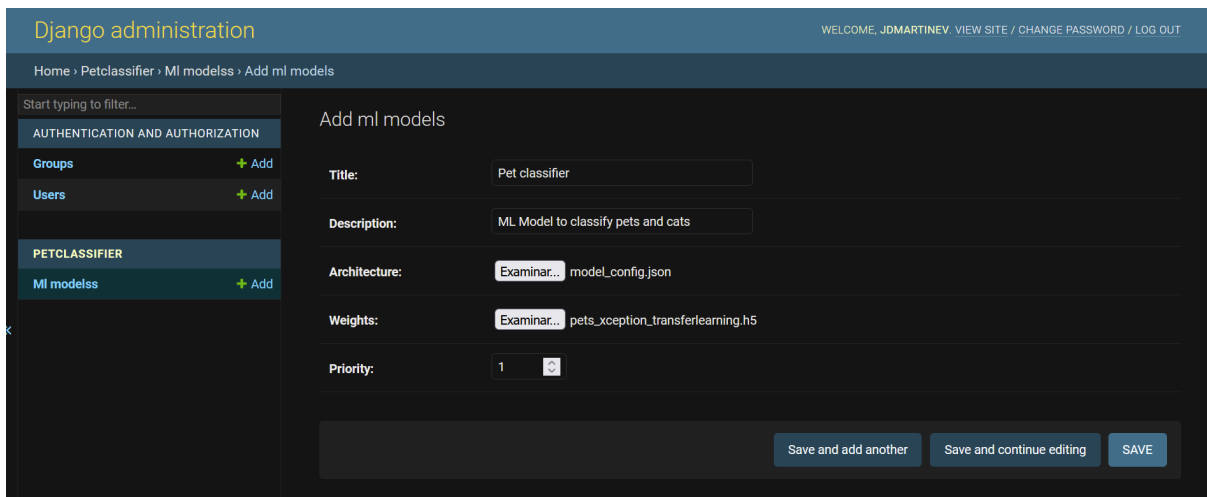
 settings.py X	 views.py	 urls.py	 models.py	 admin.py
---	--	---	---	--

```
mlapp >  settings.py > ...
1  """
2  Django settings for mlapp project.
3
4  Generated by 'django-admin startproject' using Django 4.1.1.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/4.1/topics/settings/
8
9  For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/4.1/ref/settings/
11 """
12
13 from pathlib import Path
14 import os
15
16 # Build paths inside the project like this: BASE_DIR / 'subdir'.
17 BASE_DIR = Path(__file__).resolve().parent.parent
18
19 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
20 MEDIA_URL = '/media/'
21
```


14. En **urls.py** de **mlapp**, habilite el servidor para almacenar datos.

```
mlapp > settings.py views.py urls.py X models.py admin.py home.html
mlapp > urls.py > ...
17 from django.urls import path
18 from petclassifier import views as petClassifierViews
19
20 from django.conf.urls.static import static
21 from django.conf import settings
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', petClassifierViews.home)
26 ]
27
28 urlpatterns += static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```

15. En el navegador ingrese (actualice) al enlace <http://localhost:8000/admin>. En la tabla mlModels, agregue el modelo que utilizamos en la sección 3 para clasificación de mascotas (archivos model_config.json, pets_xception_transferlearning.h5). Para esto, use la opción **Add**, luego **Add Movie** y finalmente **save**. En el campo Architecture debemos adjuntar el archivo .json (arquitectura de la red neuronal) y en el campo Weights debemos adjuntar el archivo .h5 (pesos del modelo entrenado). El campo priority se utilizará para, en caso de tener varios mlModels, el sistema sepa cuál utilizar.



Django administration

WELCOME, JDMARTINEV. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Petclassifier > ML modelss > Add ml models

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

PETCLASSIFIER

- ML modelss + Add

Add ml models

Title: Pet classifier

Description: ML Model to classify pets and cats

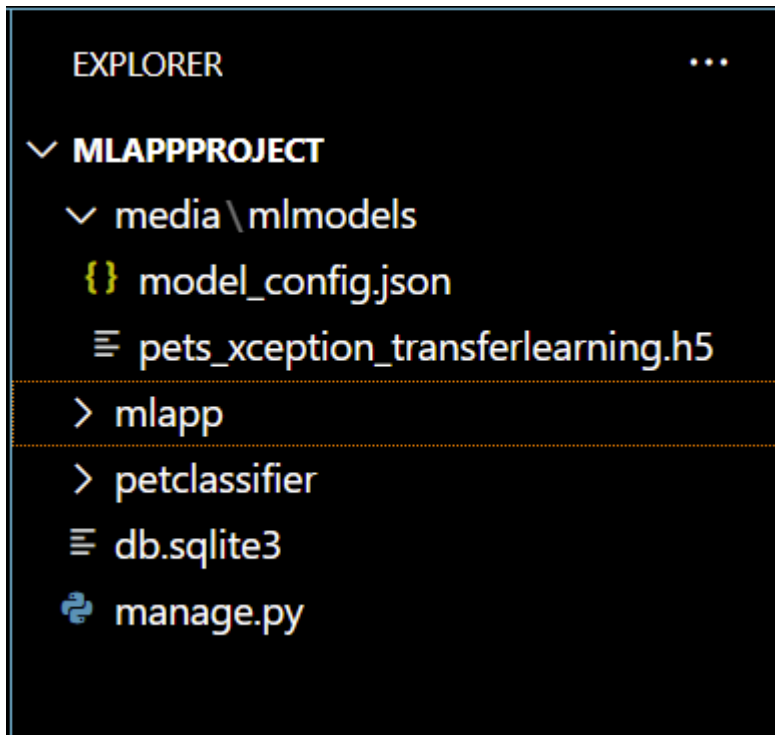
Architecture: Examinar... model_config.json

Weights: Examinar... pets_xception_transferlearning.h5

Priority: 1

Save and add another Save and continue editing SAVE

Debe verificar que los datos cargados se almacenaron correctamente en la ruta media/mlmodels



Uso de los modelos de Aprendizaje de Máquina para clasificar mascotas

1. En el archivo **views.py** de **petclassifier** importe el modelo **mlModels**, tome el modelo con el nivel de prioridad 1. Importe este modelo desde **tensorflow** en la función **home**. Para evitar warnings, importe **tensorflow** como se hizo en la etapa de test. El archivo **views.py** debería quedar de la siguiente forma:

```
from django.shortcuts import render

from .models import mlModels

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import tensorflow as tf
from tensorflow import keras

# Create your views here.

def home(request):

    petClassifierFiles = mlModels.objects.filter(priority=1)[0]
    path_arch = petClassifierFiles.architecture.path
```

```

path_weights = petClassifierFiles.weights.path

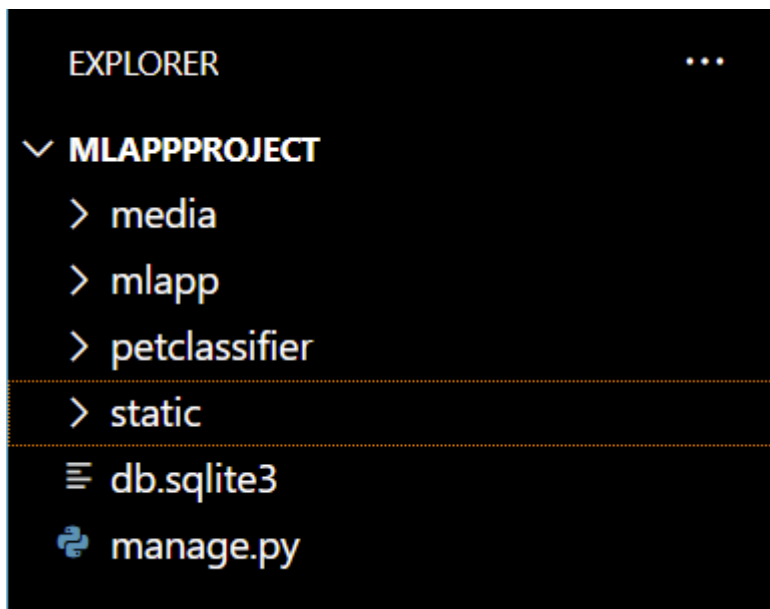
with open(path_arch) as json_file:
    json_config = json_file.read()

model = tf.keras.models.model_from_json(json_config)
model.load_weights(path_weights)

return render(request, 'home.html')

```

2. Dado que no necesitamos almacenar las imágenes que vamos a clasificar, crearemos en la carpeta raíz de nuestro proyecto **mlappproject** una carpeta **static**.



3. En el archivo **settings.py** de **mlapp** tendremos que definir la carpeta que acabamos de crear:

```

STATIC_URL = 'static/'

STATICFILES_DIRS = [
    BASE_DIR / "static",
]

```

4. Modifique el archivo **views.py** de **petclassifier** para que reciba las imágenes que se van a clasificar. Para esto, defina la función **handle_upload_file(f)** y modifique la función **home()**. El archivo **views.py** debe quedar de la siguiente forma.

```

from django.shortcuts import render

```

```

from .models import mlModels

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import tensorflow as tf
from tensorflow import keras

# Create your views here.

def home(request):

    petClassifierFiles = mlModels.objects.filter(priority=1)[0]
    path_arch = petClassifierFiles.architecture.path
    path_weights = petClassifierFiles.weights.path

    with open(path_arch) as json_file:
        json_config = json_file.read()

    model = tf.keras.models.model_from_json(json_config)
    model.load_weights(path_weights)

    if request.method == 'POST':
        handle_uploaded_file(request.FILES['sentFile'])
        image =
tf.keras.preprocessing.image.load_img('static/test.jpg', target_size =
(150,150,3))
        input_arr = tf.keras.preprocessing.image.img_to_array(image)
        input_arr = np.array([input_arr]) # Convert single image to a
batch.

        pred =
tf.keras.activations.sigmoid(model.predict(input_arr))[0][0]
        caption = f'dog prob {pred}, cat prob {1-pred}'
        return render(request, 'home.html', {'caption':caption})

def handle_uploaded_file(f):
    with open('static/test.jpg', 'wb+') as destination:
        for chunk in f.chunks():
            destination.write(chunk)

```

5. Modificamos en **templates** el archivo **home.html**

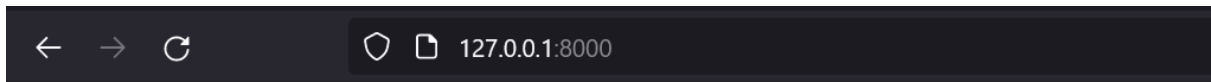
```
<head>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">
  <title>PetClassifierApp</title>
</head>

<body>
{% load static %}
  <h1>Welcome to the Pet Classifier App</h1>
  <form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    <input type="file" name="sentFile" /><br /><br />
    <input type="submit" name="submit" value="Upload" /><br />
    

    <h2>{{caption}}</h2>
  </form>
</body>
```

6. Finalmente, ejecute (actualice) el servidor desde la terminal:

```
python manage.py runserver
```



Welcome to the Pet Classifier App

No se ha seleccionado ningún archivo.



Si sube una imagen, deberá ver algo similar a:

Welcome to the Pet Classifier App

No se ha seleccionado ningún archivo.



dog prob 0.9584529399871826, cat prob 0.04154706001281738