Bitacora

<https://www.delftstack.com/es/howto/django/django-upload-file-or-image/#ir-a-la-url-http-127-0-0-1-8000>

1. Instalar el entorno virtual

pip install virtualenv

1. Instalar Django y Pillow (Pillow es un paquete python para cargar imágenes).

pip install django

pip install Pillow

1. Crear un Proyecto en Django

django-admin startproject proteitm

* 1. ingresamos al cmd y entramos a la carpeta (directorio) proteitm

cd proteitm

* 1. dentro del Proyecto “proteitm” creamos una app

django-admin startapp cargueArchivos

* + 1. Tambien puede funcionar

py manage.py startapp cargueArchivos

* 1. Para registrar la aplicación, ingresamos el nombre de la aplicación dentro de la lista INSTALLED\_APPS en el archivo settings.py

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

"cargueArchivos" # 🡪 aquí

]

* 1. Luego, realizamos migraciones iniciales con el siguiente commando:

py manage.py migrate

* 1. Probamos el servidor:

py manage.py runserver

1. Actualizar el archivo

Debemos establecer donde se va a descargar el archivo

Ingresamos al archivo settings.py

import os

MEDIA\_URL = '/media/

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

1. Para almacenar los archivos cargados, temenos que crear un modelo.

El campo del modelo tiene la ruta del archive cargado, pero no el archive en sí

Crearemos un modelo document para almacenar los detaller sobre los archivos cargados.

cargueArchivos\models.py

from django.db import models

class Document(models.Model):

title = models.CharField(max\_length = 200) #para almacenar un título

uploadedFile = models.FileField(upload\_to = "Uploaded Files/")

dateTimeOfUpload = models.DateTimeField(auto\_now = True) #dice la fecha y hora de la carga

title = para almacenar un título

dateTimeOfUpload = almacenará la fecha y hora de la carga

FileFiel() para almacenar el archive.

Upload\_to se utiliza para definer la carpeta

Antes de usar este modelo se debe realizar migraciones y migrarlas, para eso ejecutamos:

python manage.py makemigrations

python manage.py migrate

1. Creación de formularios

Dentro de la carpeta cargueArchivos, creamos una carpeta llamada “templates”. Dentro de esta, creamos otra carpeta llamada cargueArchivos. Por ultimo, creamos un nuevo archive dentro de esta carpeta llamada páginaPrincipal.html

* 1. En el archive página principal agregamos

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Django File Upload</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');

\* {

font-family: "Roboto";

}

body {

background-color: #F5F5F5;

}

form {

background-color: #FFFFFF;

padding: 25px;

}

table {

width: 100%;

text-align: center;

margin-top: 25px;

padding: 25px;

background-color: #FFFFFF;

}

table tr td {

padding-top: 15px;

}

</style>

</head>

<body>

<form action="{% url 'Core:uploadFile' %}" method="POST" enctype="multipart/form-data">

<input type="text" name="fileTitle" placeholder="Enter a title">

<input type="file" name="uploadedFile">

{% csrf\_token %}

<input type="submit" value="Upload">

</form>

<table>

<tr>

<th>ID</th>

<th>Title</th>

<th>File Path</th>

<th>Upload Date & Time</th>

</tr>

{% for file in files %}

<tr>

<td>{{ file.id }}</td>

<td>{{ file.title }}</td>

<td>{{ file.uploadedFile.url }}</td>

<td>{{ file.dateTimeOfUpload }}</td>

</tr>

{% endfor %}

</table>

</body>

</html>

1. Agregar vistas

Para mostrar la plantilla html y la información crearemos una vista cargueArchivos()

En cargueArchivos\views.py

from . import models

from django.shortcuts import render

def uploadFile(request):

if request.method == "POST":

# Fetching the form data

fileTitle = request.POST["fileTitle"]

uploadedFile = request.FILES["uploadedFile"]

# Saving the information in the database

document = models.Document(

title = fileTitle,

uploadedFile = uploadedFile

)

document.save()

documents = models.Document.objects.all()

return render(request, "Core/upload-file.html", context = {

"files": documents

})

Esto permitirá ver la plantilla html que acabamos de crear. El formulario plantilla se envía a la misma url y se gestona en esta vista.

1. Configurar el archive urls.py

Dentro de la aplicación cargueArchivos creamos un archive llamado urls.py allí agregamos el siguiente Código:

from . import views

from django.urls import path

from django.conf import settings

from django.conf.urls.static import static

app\_name = "Core"

urlpatterns = [

path("", views.uploadFile, name = "uploadFile"),

]

if settings.DEBUG:

urlpatterns += static(

settings.MEDIA\_URL,

document\_root = settings.MEDIA\_ROOT

)

Debemos vincular todas las URL al Proyecto principal

Para hacer eso, agregamos la siguiente declaración dentro de la lista urlpattterns en dinamicaMolecular\urls.py

path("", include("cargueArchivos.urls")),

debe verse así

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path("", include("Core.urls")),

]