

Manual Paso a Paso

{ Proyecto Python
Django }



Autores

Disneyyna Jenniredd Tarazona Flórez

Jhon Davidson Diaz Corzo

Introducción

Bienvenido a este manual de Django y Python, donde te guiaremos paso a paso en la creación de un proyecto práctico y útil: una agenda de contactos. En la era digital, la gestión eficiente de contactos es esencial, y este manual ha sido diseñado para que aprendas a utilizar dos herramientas poderosas, Django y Python, en la construcción de tu propia aplicación de agenda.

¿Por qué una agenda de contactos?

Una agenda de contactos es el punto de partida perfecto para explorar las capacidades de Django y Python. A través de este proyecto, no solo aprenderás los fundamentos del desarrollo web, sino que también obtendrás habilidades prácticas para organizar y mantener tu información de contacto de manera efectiva.

¿Qué hace especial a Django y Python en este contexto?

Django simplifica el desarrollo web al proporcionar una estructura organizada y herramientas integradas que agilizan el proceso. Python, con su sintaxis clara y legible, permite que el desarrollo sea accesible para principiantes y potente para profesionales.

¿Para quién es este manual?

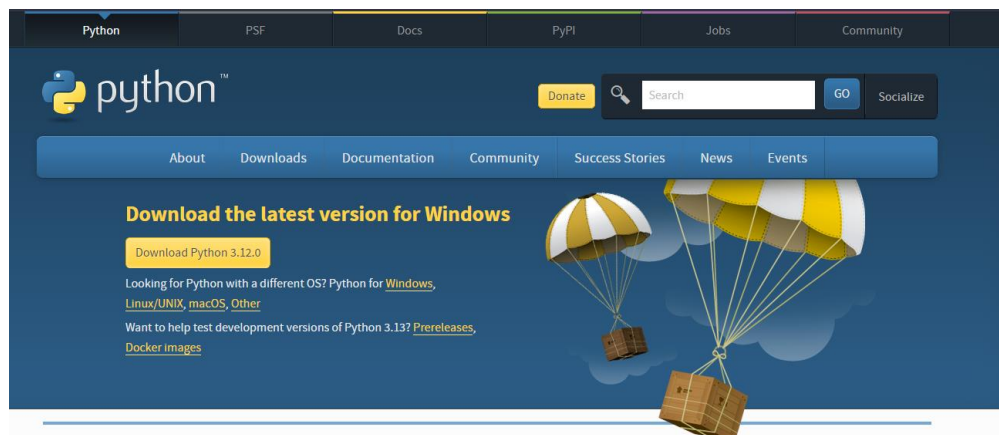
Este manual está diseñado para cualquier persona interesada en aprender a desarrollar aplicaciones web con Django y Python, incluso si no tienes experiencia previa en programación. A lo largo de este proceso, te familiarizarás con la creación de modelos para almacenar información, la implementación de vistas para interactuar con los datos y la construcción de una interfaz de usuario atractiva y funcional.

Lo que lograrás al finalizar este manual:

- Crear una aplicación de agenda de contactos desde cero.
- Comprender los conceptos clave de Django, como modelos, vistas y plantillas.
- Integrar formularios para agregar y editar contactos de manera fácil.
- Almacenar y recuperar datos de manera eficiente utilizando una base de datos.
- Desplegar tu aplicación para que sea accesible en línea.

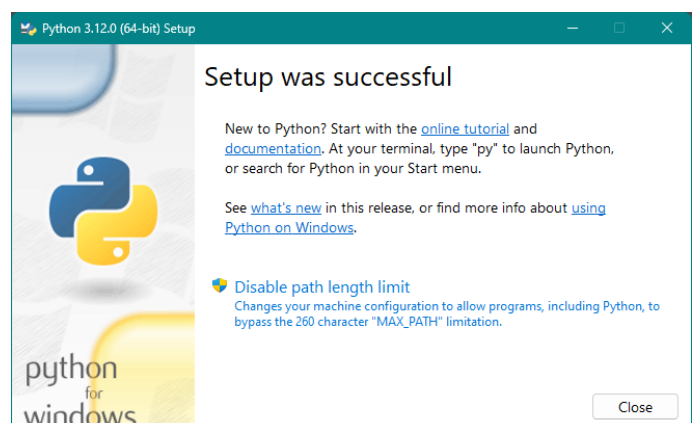
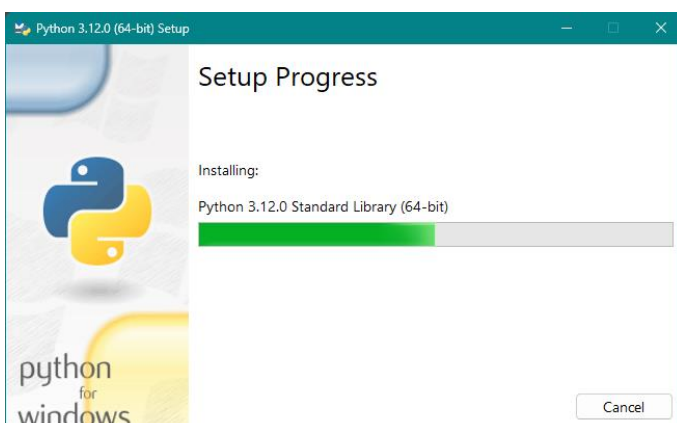
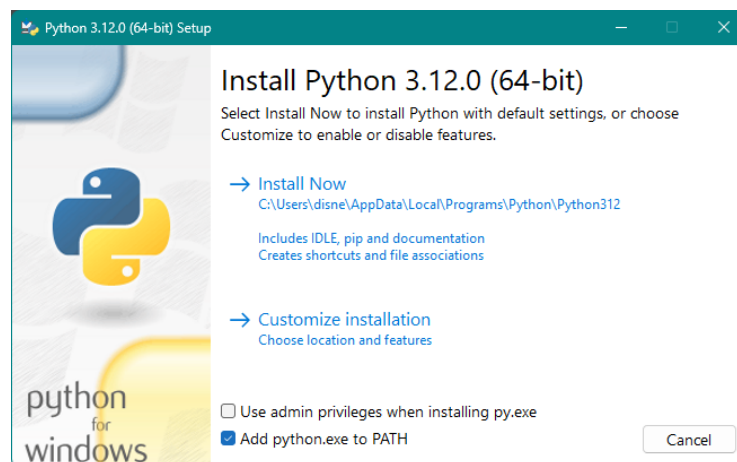
DESCARGA E INSTALACIÓN DE PYTHON

1. Para descargar e instalar Python lo primero que realizaremos será ir a la página de descargas de Python (<https://www.python.org/downloads/>).



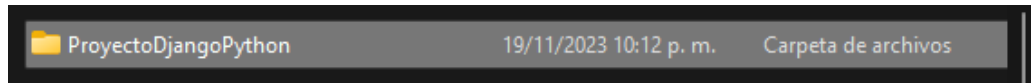
Daremos clic en el botón amarillo para descargar y esperaremos a que finalice el proceso.

2. Una vez finalizada la descarga, ejecutaremos el archivo y en esta ventana, nos aseguraremos de marcar la casilla que dice "Add Python.exe to PATH" antes de hacer clic en "Install Now". Esto facilitará el uso de Python desde la línea de comandos.



INICIACIÓN DEL PROYECTO

1. Crearemos una carpeta de forma manual en el disco local C de Windows



2. Abriremos el cmd desde la carpeta creada anteriormente.
3. En el cmd crearemos otra carpeta, en este caso llamada *contactlist* y al ubicarnos en esta carpeta instalaremos el entorno virtual de Python.

```
C:\Windows\System32\cmd.e X + v - □ X
Microsoft Windows [Versión 10.0.22621.2715]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\ProyectoDjangoPython>mkdir contactlist

C:\ProyectoDjangoPython>cd contactlist

C:\ProyectoDjangoPython\contactlist>python -m venv env

C:\ProyectoDjangoPython\contactlist>|
```

4. Lo siguiente que haremos será activar el entorno virtual

```
C:\ProyectoDjangoPython\contactlist>.\env\Scripts\activate
(env) C:\ProyectoDjangoPython\contactlist>|
```

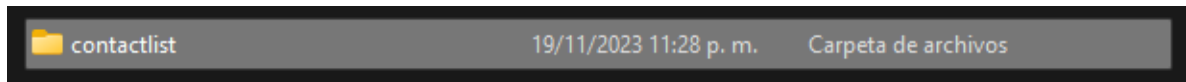
5. Una vez activado el entorno virtual procederemos a descargar django utilizando el gestor de paquetes de Python llamado pip

```
C:\Windows\System32\cmd.e X + v - □ X
(env) C:\ProyectoDjangoPython\contactlist>pip install Django
Collecting Django
  Obtaining dependency information for Django from https://files.pythonhosted.org/packages/2d/6d/e87236e3c7b2f5911d132034177aebb605f3953910cc429df8061b13bf10/Django-4.2.7-py3-none-any.whl.metadata
  Using cached Django-4.2.7-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.6.0 (from Django)
  Obtaining dependency information for asgiref<4,>=3.6.0 from https://files.pythonhosted.org/packages/9b/80/b9051a4a07ad231558fcd8fffc89232711b4e618c15cb7a392a17384bbeef/asgiref-3.7.2-py3-none-any.whl.metadata
  Using cached asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
Collecting sqlparse>=0.3.1 (from Django)
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Collecting tzdata (from Django)
  Using cached tzdata-2023.3-py2.py3-none-any.whl (341 kB)
Using cached Django-4.2.7-py3-none-any.whl (8.0 MB)
Using cached asgiref-3.7.2-py3-none-any.whl (24 kB)
Installing collected packages: tzdata, sqlparse, asgiref, Django
Successfully installed Django-4.2.7 asgiref-3.7.2 sqlparse-0.4.4 tzdata-2023.3

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(env) C:\ProyectoDjangoPython\contactlist>
```

6. Una vez se haya instalado django, lo siguiente será crear el proyecto usando el mismo nombre de la carpeta y especificando que se hará dentro de la misma.



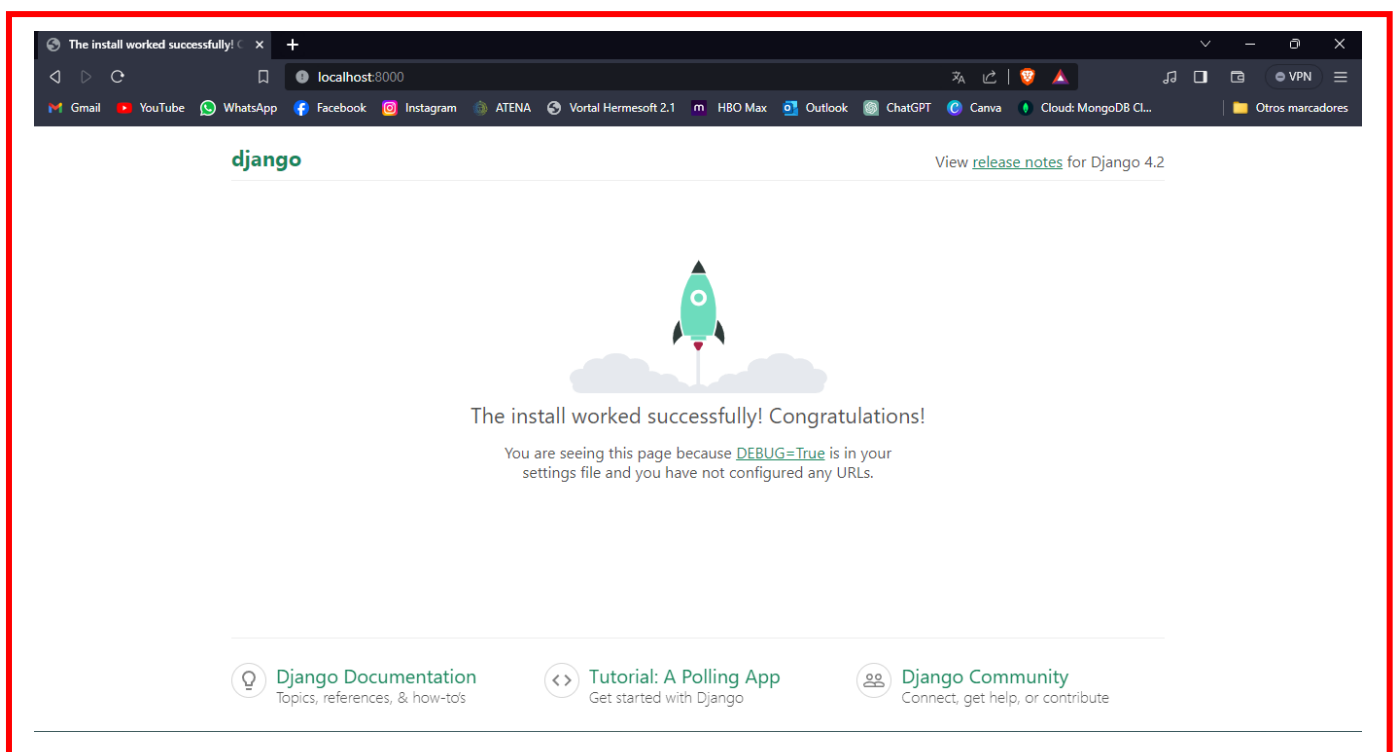
Ya con este comando, visualizaremos nuestro proyecto creado dentro de la carpeta de trabajo.

```
(env) C:\ProyectoDjangoPython\contactlist>django-admin startproject contactlist .  
(env) C:\ProyectoDjangoPython\contactlist>|
```

7. Ahora pasaremos a correr la aplicación

```
(env) C:\ProyectoDjangoPython\contactlist>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.  
November 19, 2023 - 23:33:09  
Django version 4.2.7, using settings 'contactlist.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```

Visualizaremos la ejecución dirigiéndonos a: <http://localhost:8000/> . Aquí podremos ver la página de aterrizaje de django al ejecutar la aplicación correctamente.

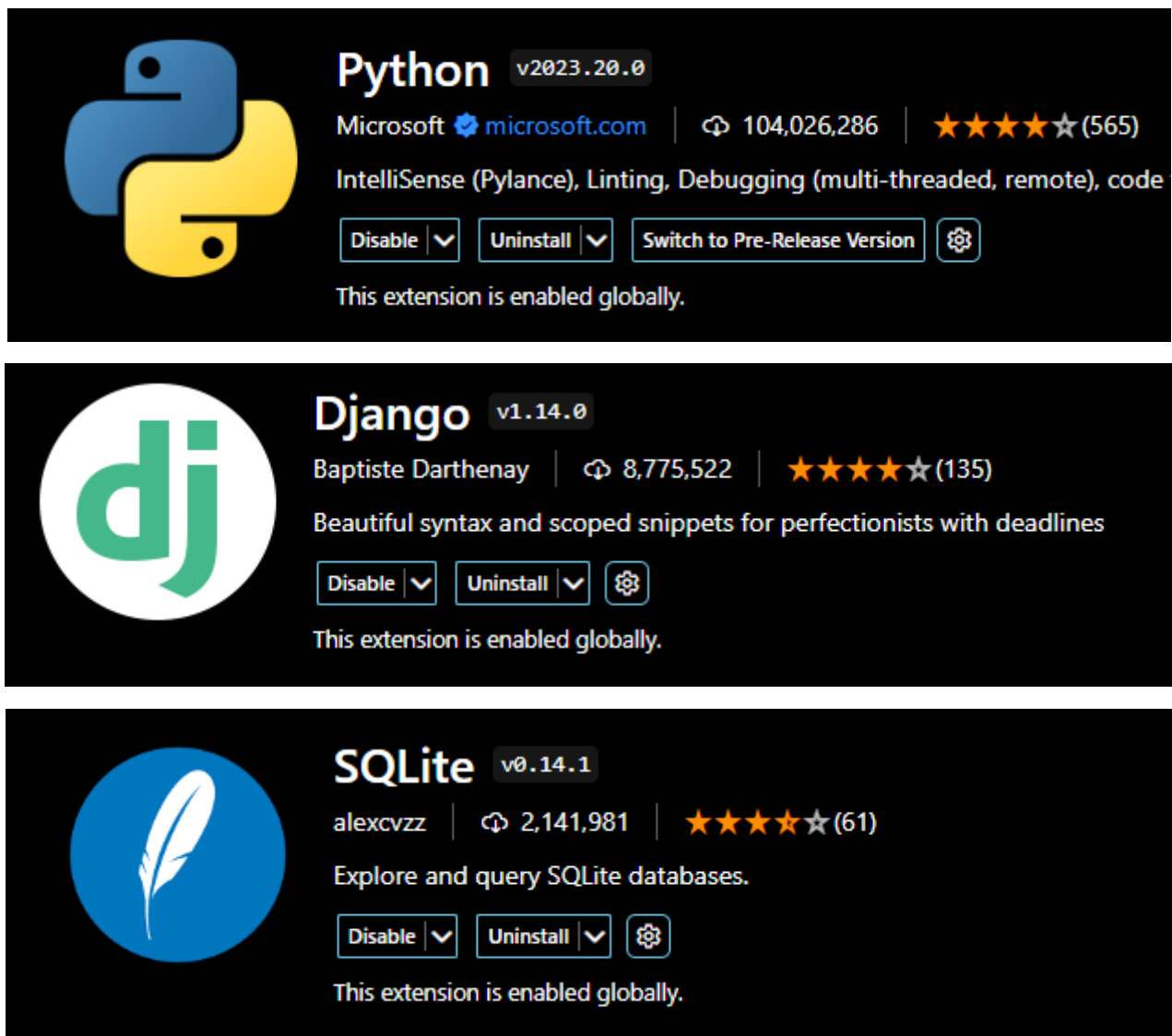


DESARROLLO DEL PROYECTO EN VISUAL STUDIO CODE

Para iniciar a trabajar en el proyecto, se debe tener en cuenta que, el proyecto debe estarse ejecutando de manera correcta en el cmd y que no se podrá cerrar esta terminal con el servidor mientras se esté trabajando en el proyecto.

Lo primero que haremos será abrir nuestra carpeta *contactlist* en nuestro editor de código Visual Studio Code.

Una vez aquí, hablaré de algunas extensiones que son de utilidad al momento de codificar nuestro proyecto. Esas extensiones son:



The image shows three extension cards from the Visual Studio Code marketplace, each with a dark background. The first card is for the 'Python' extension by Microsoft, version v2023.20.0, with 104,026,286 downloads and a 4.5-star rating (565 reviews). It lists features like IntelliSense (Pylance), Linting, and Debugging. The second card is for the 'Django' extension by Baptiste Darthenay, version v1.14.0, with 8,775,522 downloads and a 4.5-star rating (135 reviews). It describes it as having beautiful syntax and scoped snippets. The third card is for the 'SQLite' extension by alexcvzz, version v0.14.1, with 2,141,981 downloads and a 4.5-star rating (61 reviews). It is described as a tool to explore and query SQLite databases. Each card includes icons for 'Disable', 'Uninstall', and 'Switch to Pre-Release Version' (or a settings gear), and a note stating 'This extension is enabled globally.'

Python v2023.20.0
Microsoft microsoft.com | 104,026,286 | ★★★★★ (565)
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), code
[Disable] [Uninstall] [Switch to Pre-Release Version] [Settings]
This extension is enabled globally.

Django v1.14.0
Baptiste Darthenay | 8,775,522 | ★★★★★ (135)
Beautiful syntax and scoped snippets for perfectionists with deadlines
[Disable] [Uninstall] [Settings]
This extension is enabled globally.

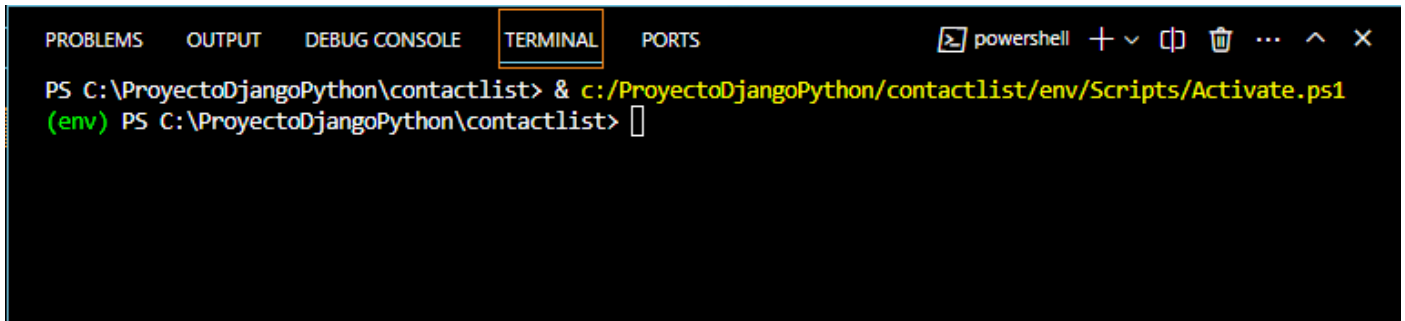
SQLite v0.14.1
alexcvzz | 2,141,981 | ★★★★★ (61)
Explore and query SQLite databases.
[Disable] [Uninstall] [Settings]
This extension is enabled globally.

Estas extensiones nos permitirán hablar en el mismo lenguaje de programación, usando django como framework propio de Python.

Además, SQLite nos permitirá visualizar la base de datos, mostrar y evidenciar las migraciones que se realicen dentro de nuestro proyecto, así como la información que se ingrese a través de este.

Ya en Visual Studio Code, lo siguiente que haremos será abrir el terminal y verificar el comportamiento de nuestro proyecto.

Al iniciar la terminal, veremos que se ejecutó un script inicial que nos permitirá acceder al entorno virtual.

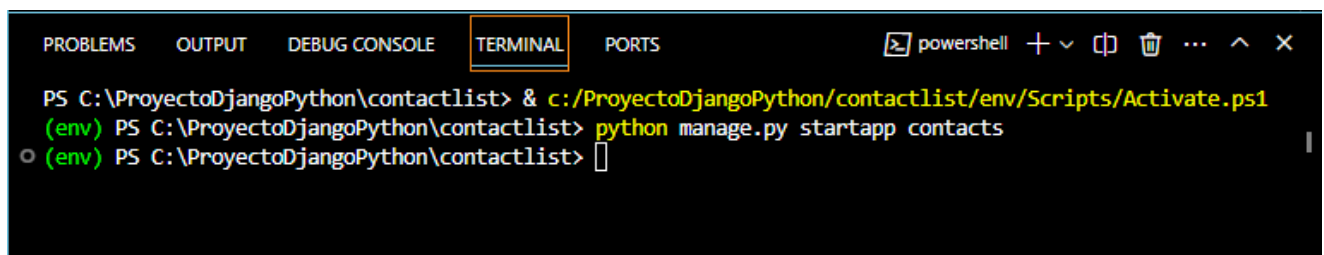


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + v [icon] [icon] ... ^ x
PS C:\ProyectoDjangoPython\contactlist> & c:/ProyectoDjangoPython/contactlist/env/Scripts/Activate.ps1
(env) PS C:\ProyectoDjangoPython\contactlist> 
```

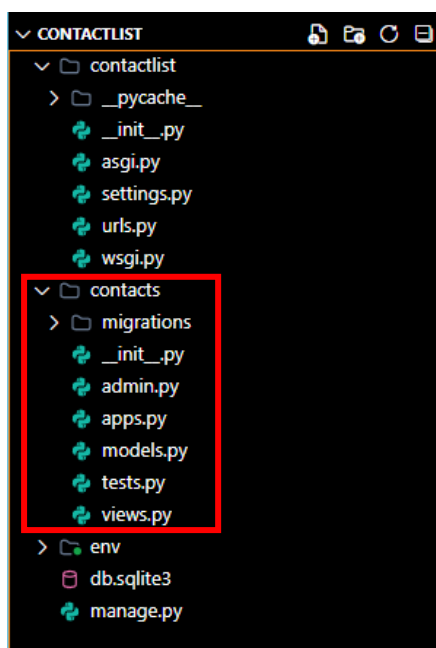
De este modo, podremos empezar a trabajar en nuestro proyecto implementando el entorno virtual desde la terminal de Visual Studio Code.

CREACIÓN DE LA APLICACIÓN

Una vez ubicados en la terminal y ejecutado el entorno virtual pasaremos a crear nuestra aplicación. Hasta el momento hemos creado proyectos y espacios de trabajos, es momento de crear nuestra aplicación en concreto.



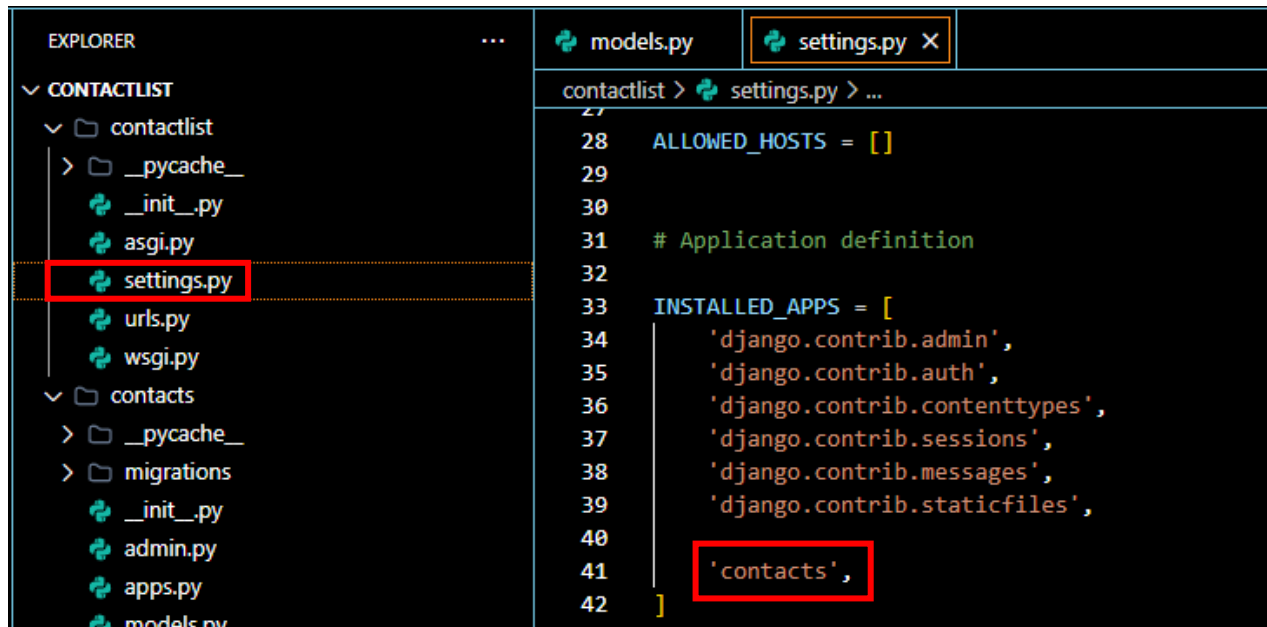
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + v [icon] [icon] ... ^ x
PS C:\ProyectoDjangoPython\contactlist> & c:/ProyectoDjangoPython/contactlist/env/Scripts/Activate.ps1
(env) PS C:\ProyectoDjangoPython\contactlist> python manage.py startapp contacts
(env) PS C:\ProyectoDjangoPython\contactlist> 
```



Aquí, ya podremos ver que el comando anterior se ejecutó correctamente y que realizó la creación de la carpeta correspondiente a nuestra aplicación *contacts*.

Otra cosa importante tener en cuenta, es que, a lo largo del proyecto realizaremos migraciones, estas migraciones nos permiten trasladar los datos dentro de la base de datos.

Para que nuestra aplicación *contacts* sea reconocida precisamente como una aplicación, que haremos será añadirla dentro del archivo de configuración de nuestro proyecto en el apartado de *INSTALLED_APPS*

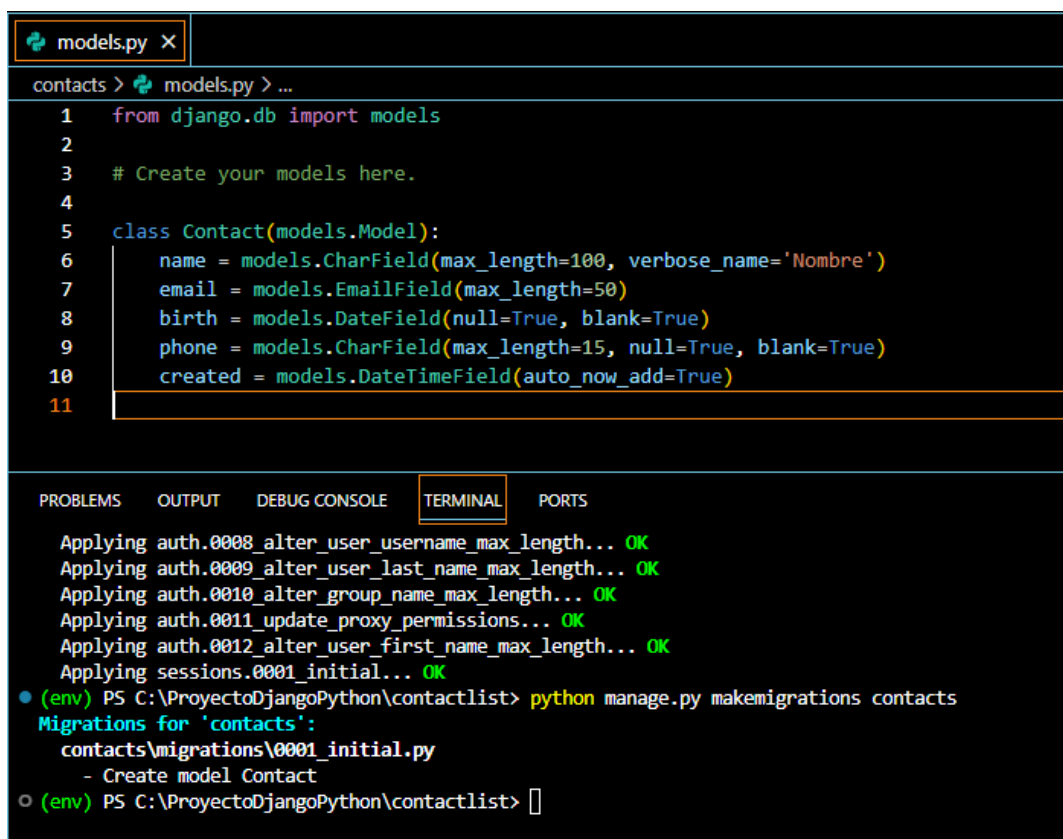


```
EXPLORER
CONTACTLIST
├── contactlist
│   ├── __pycache__
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── contacts
│   ├── __pycache__
│   ├── migrations
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   └── models.py
└── models.py

models.py
settings.py X

contactlist > settings.py > ...
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
41     'contacts',
42 ]
```

Ahora, pasaremos a la codificación de nuestra tabla, para ello, en la carpeta de nuestra aplicación, en el archivo *models.py* crearemos lo siguiente para después realizar la migración.



```
models.py X

contacts > models.py > ...
1 from django.db import models
2
3 # Create your models here.
4
5 class Contact(models.Model):
6     name = models.CharField(max_length=100, verbose_name='Nombre')
7     email = models.EmailField(max_length=50)
8     birth = models.DateField(null=True, blank=True)
9     phone = models.CharField(max_length=15, null=True, blank=True)
10    created = models.DateTimeField(auto_now_add=True)
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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
(env) PS C:\ProyectoDjangoPython\contactlist> python manage.py makemigrations contacts
Migrations for 'contacts':
  contacts\migrations\0001_initial.py
    - Create model Contact
(env) PS C:\ProyectoDjangoPython\contactlist> 
```

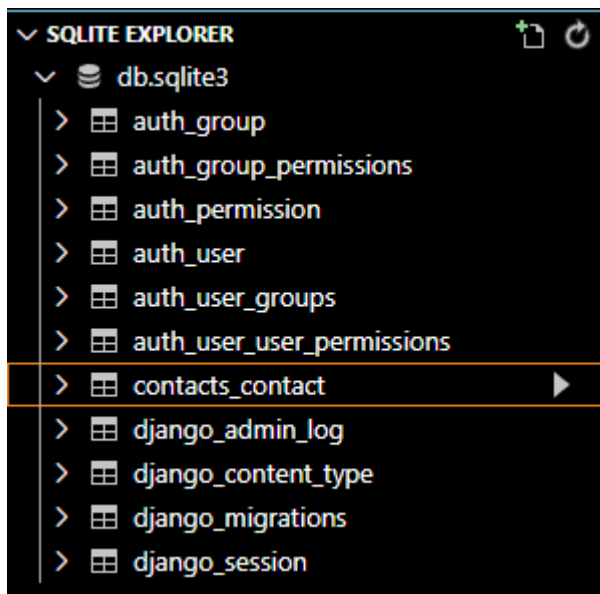


```

• (env) PS C:\ProyectoDjangoPython\contactlist> python manage.py makemigrations contacts
Migrations for 'contacts':
  contacts\migrations\0001_initial.py
    - Create model Contact
• (env) PS C:\ProyectoDjangoPython\contactlist> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contacts, contenttypes, sessions
Running migrations:
  Applying contacts.0001_initial... OK
• (env) PS C:\ProyectoDjangoPython\contactlist>

```

Al realizar la migración correcta, podremos visualizar en la base de datos, la creación de nuestra tabla:



contacts_contact	
id	integer
nombre	varchar(100)
email	varchar(50)
fechaNacimiento	date
telefono	varchar(15)
created	datetime

De este modo podremos ver que se crearon los atributos establecidos dentro de mi clase models.py

Ahora almacenaremos el usuario, que nos permitirá iniciar sesión en nuestro proyecto a través de Django.

```

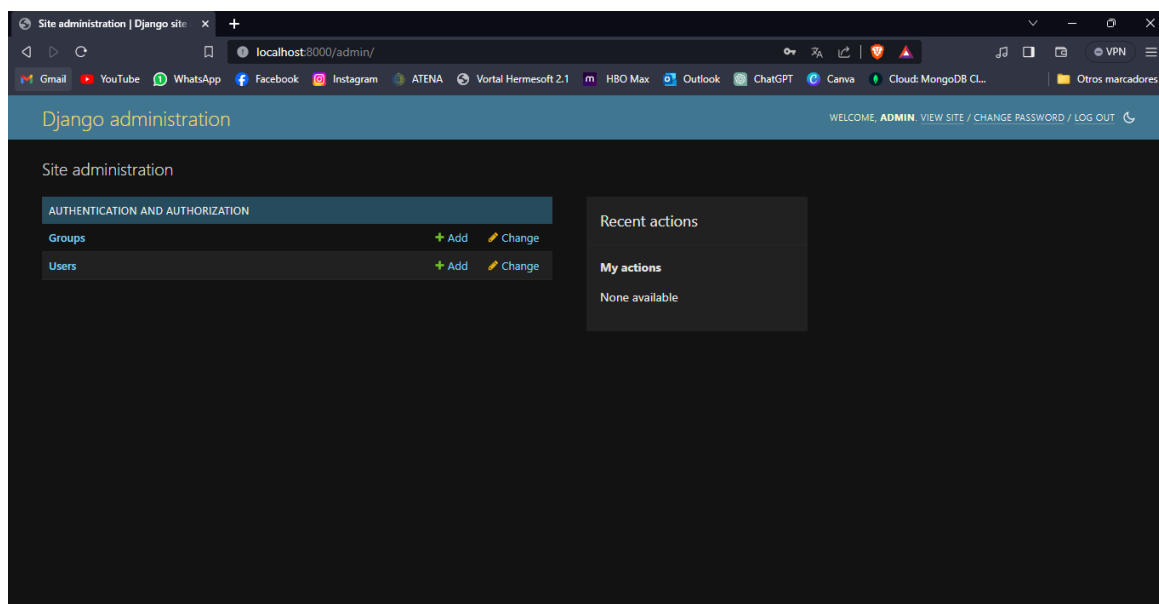
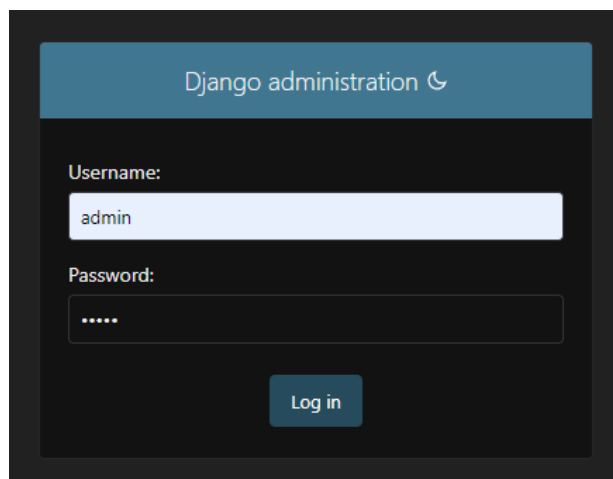
• (env) PS C:\ProyectoDjangoPython\contactlist> python manage.py createsuperuser
Username (leave blank to use 'disne'): admin
Email address: disneyinatf@gmail.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
• (env) PS C:\ProyectoDjangoPython\contactlist>

```

Usando ese comando, podremos crear al usuario, una vez se ejecute, nos preguntará en nombre de usuario, un correo electrónico, que puede ser opciones, y un password. En este caso, el texto en rojo es una advertencia acerca de lo insegura que es mi password. Es decir, que de manera opcional podremos establecer un password con más caracteres y seguridad.

Lo siguiente que haremos será ingresar al panel de administración de django. Para ellos, junto a la dirección en la que estamos, escribiremos: <http://localhost:8000/admin> y veremos el formulario de inicio de sesión.

Para poder ingresar, usaremos el usuario creado anteriormente:

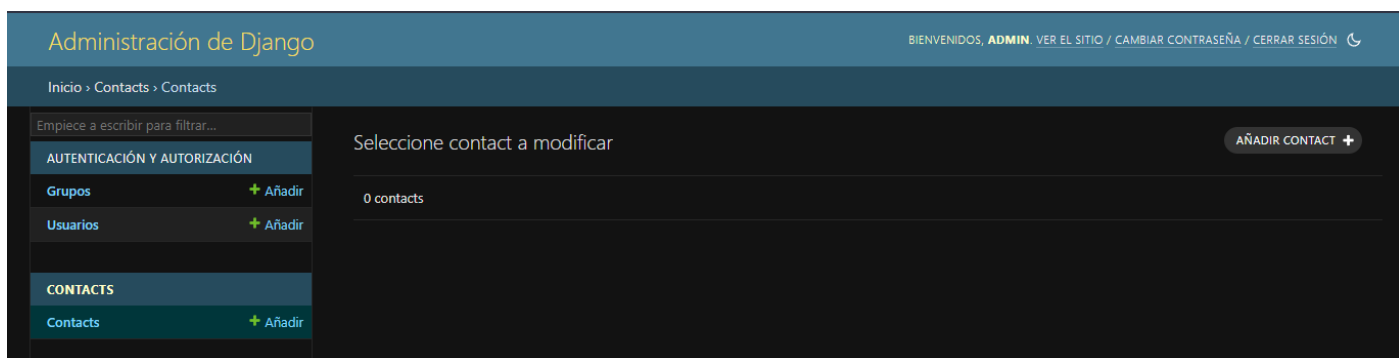


Está será la interfaz de administración que nos recibirá al momento de iniciar sesión.

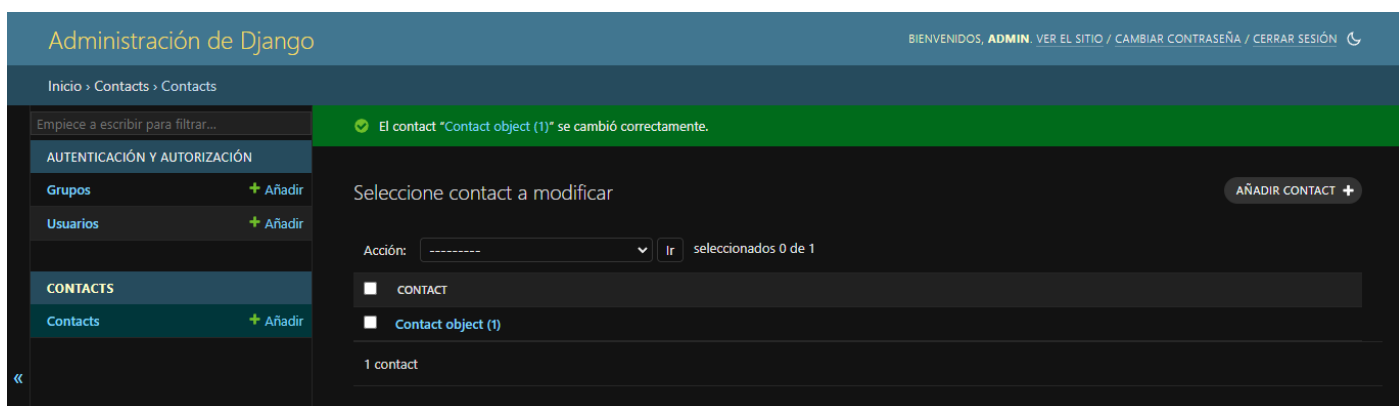
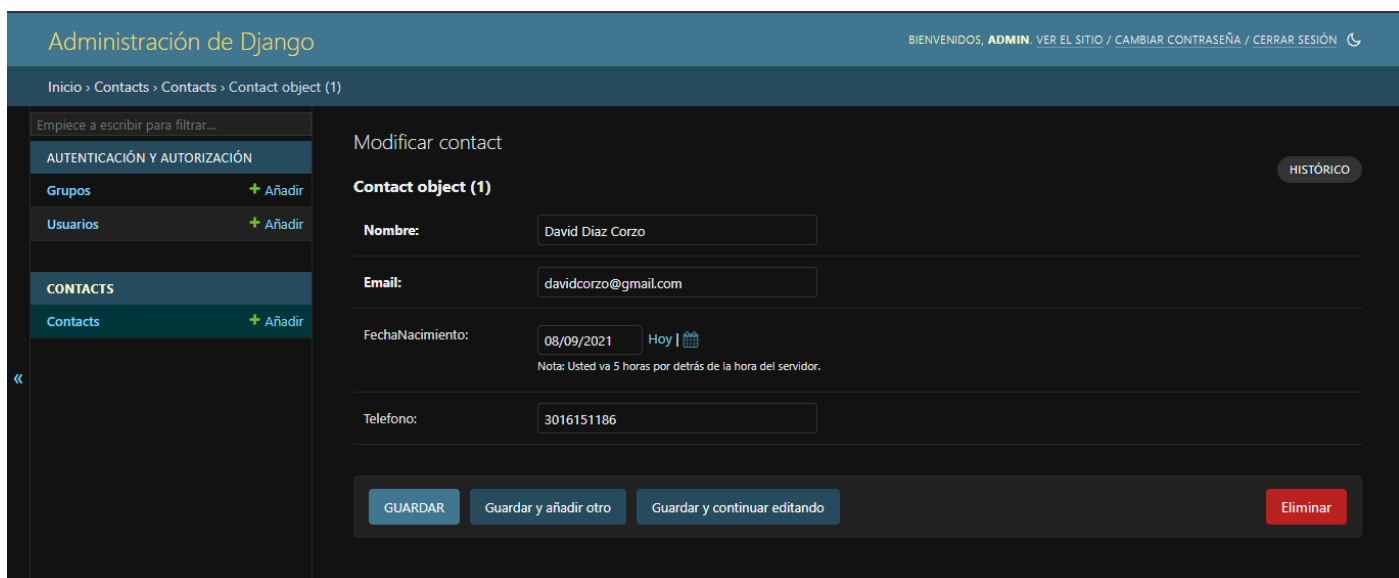
Para añadir nuestra aplicación al panel de administrador, haremos lo siguiente en el archivo *admin.py*

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

Esto nos permitirá
añadir la aplicación
contacts a nuestro
administrador.



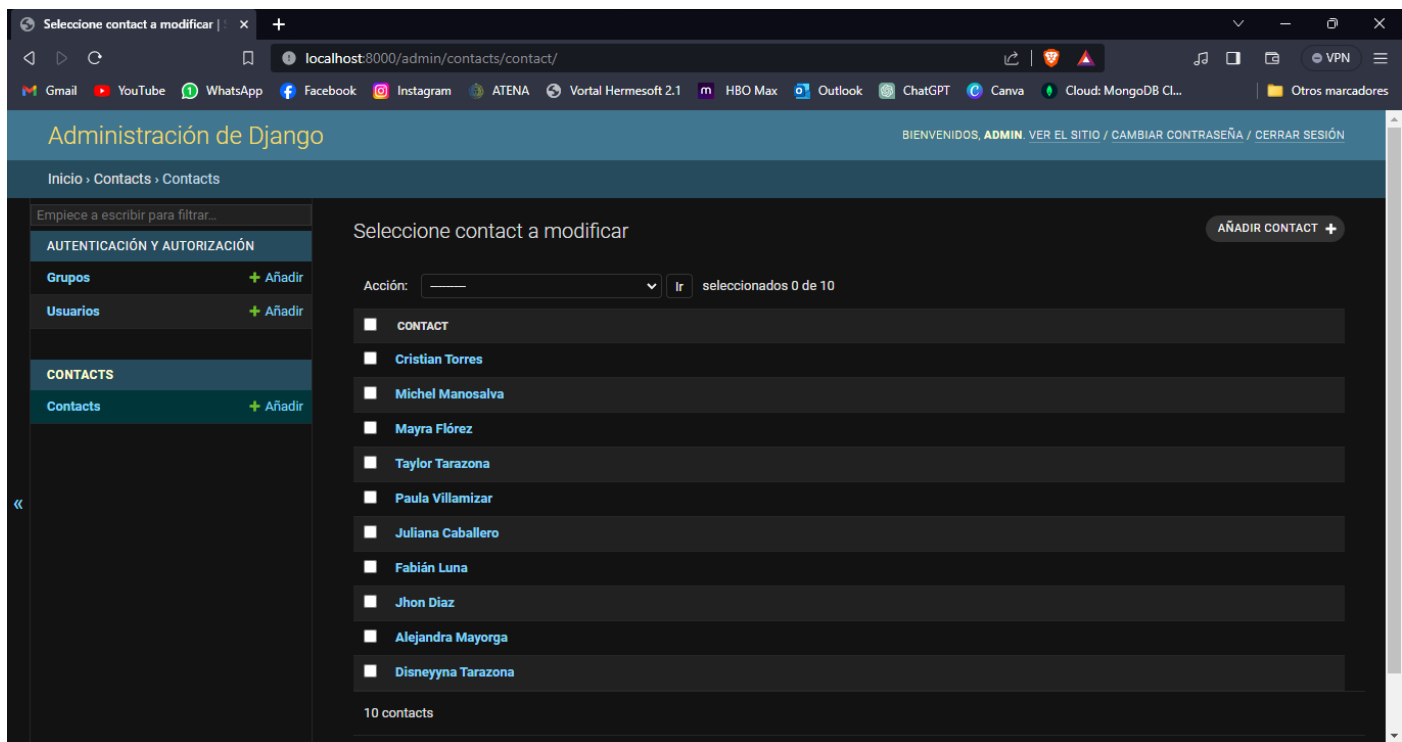
Aquí, ya podremos ver como nuestra aplicación ya está añadida al panel de administración de Django, y que desde la opción AÑADIR CONTACT+ ya podemos realizar la inserción de datos.



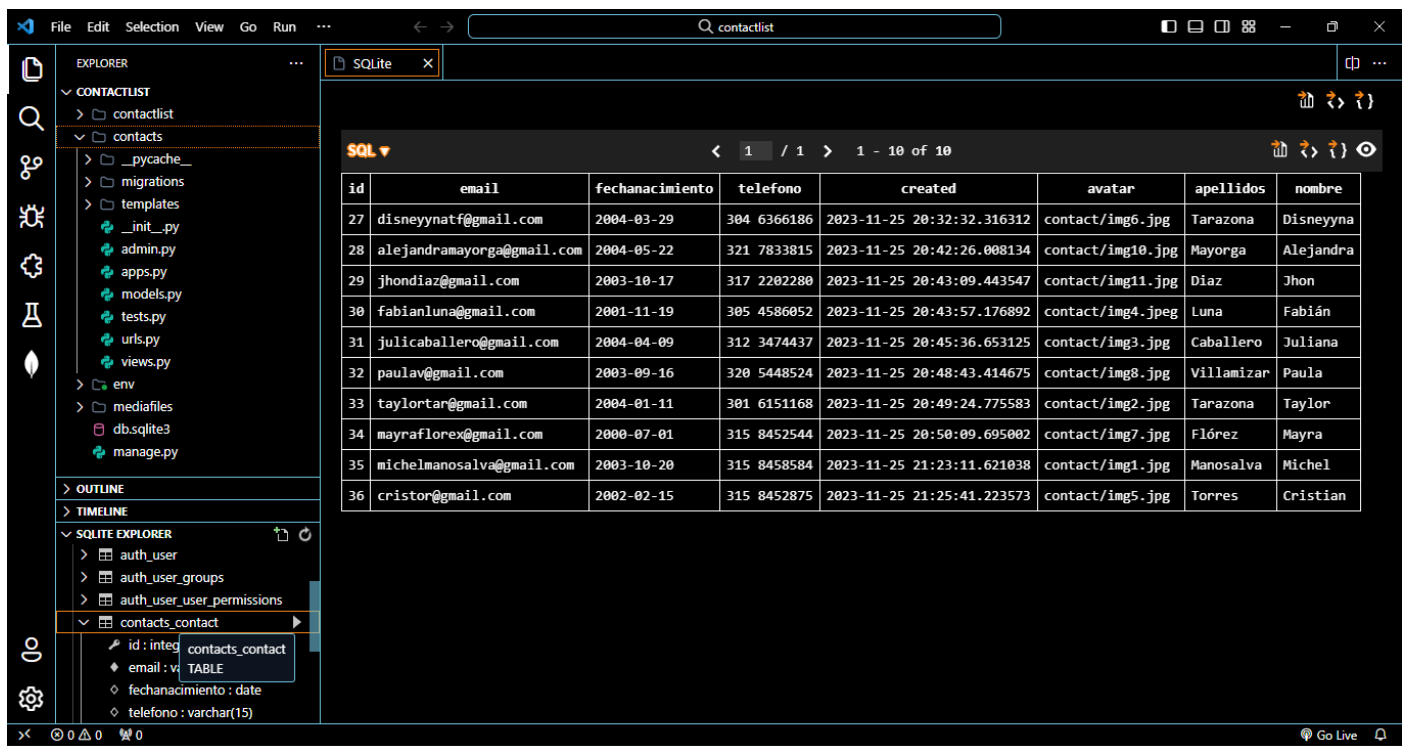
De esta manera, ya podremos observar la implementación de nuestra aplicación y su correcta funcionalidad. (Contacto añadido con éxito)

Aplicando una nueva configuración, podremos ver los nombres de cada registro.

Dentro de este administrador de Django ya podremos crear, editar, eliminar y visualizar los datos que se deseen.



Lo que a través de mi base de datos SQLite podremos visualizar de la siguiente manera también:



Lo siguiente que haremos será construir las interfaces de nuestra aplicación de contactos, haremos todo lo relacionado con el aspecto visual que le daremos a nuestra agenda.

Para pasar a esta parte, mencionaremos que, se realizaron cambios al modelo para ajustar la aplicación a su resultado final, por ende, mostraremos desde el modelo configurador en adelante.

CONFIGURACIÓN PARA NUESTRA APLICACIÓN FINAL

1. Archivo models.py

Para poder pasar al desarrollo de lo que fue nuestra interfaz final, mostraremos primero que todo el *models.py* completo de nuestra aplicación.

```
models.py X
contacts > models.py > ...
1  from django.db import models
2
3  # Create your models here.
4  class Contact(models.Model):
5      avatar = models.ImageField(upload_to='contact', null=True, blank=True)
6      nombre = models.CharField(max_length=100, null=True, blank=True)
7      apellidos = models.CharField(max_length=100, null=True, blank=True)
8      email = models.EmailField(max_length=50)
9      fechanacimiento = models.DateField(null=True, blank=True, verbose_name='Fecha de nacimiento')
10     telefono = models.CharField(max_length=15, null=True, blank=True, verbose_name='Teléfono')
11     created = models.DateTimeField(auto_now_add=True)
12
13     def __str__(self) -> str:
14         # Devuelve el nombre y apellidos si ambos existen
15         if self.nombre and self.apellidos:
16             return f"{self.nombre} {self.apellidos}"
17         elif self.nombre:
18             return self.nombre
19         elif self.apellidos:
20             return self.apellidos
21         else:
22             return self.email
23
24
```

Este archivo, como se puede observar, contiene las variables que manejaremos dentro de nuestra aplicación, tenemos su tipo de dato, su limite de caracteres, y adicional a ello tendremos la traducción de nombre para *fechanacimiento* y *telefono*, con el propósito de mejorar la interfaz más adelante. Además, tenemos la configuración que nos permitió mostrar el nombre y el apellido de cada dato.

2. Archivo views.py

Lo siguiente que mostraremos será la configuración del archivo *views.py*, este archivo contendrá los enlaces de nuestra aplicación para ejecutar de manera correcta los métodos que ejecutarán y completarán el CRUD de nuestra aplicación.

```
views.py X
contacts > views.py > ...
1  from typing import Any
2  from django.shortcuts import render
3  from django.urls import reverse_lazy
4
5  from django.views import generic
6  from django.db.models import QuerySet
7
8  from contacts.models import Contact
9
```

Tendremos las librerías necesarias y las importaciones de nuestro proyecto.

```
views.py x
contacts > views.py > ...
9
10 # Create your views here.
11 class ContactListView(generic.ListView):
12     model = Contact
13     paginate_by = 5
14
15     def get_queryset(self) -> QuerySet[Any]:
16         q = self.request.GET.get('q')
17
18         if q:
19             return Contact.objects.filter(nombre__icontains=q)
20
21         return super().get_queryset()
22
23 class ContactCreateView(generic.CreateView):
24     model = Contact
25     fields = ('avatar', 'nombre', 'apellidos', 'email', 'fechanacimiento', 'telefono',)
26     success_url = reverse_lazy('contact_list')
27
28
29 class ContactUpdateView(generic.UpdateView):
30     model = Contact
31     fields = ('avatar', 'nombre', 'apellidos', 'email', 'fechanacimiento', 'telefono',)
32     success_url = reverse_lazy('contact_list')
33
34
35 class ContactDeleteView(generic.DeleteView):
36     model = Contact
37     success_url = reverse_lazy('contact_list')
38
```

Este archivo, como se puede mostrar, contiene los métodos listar, crear, actualizar y eliminar. Métodos que serán ejecutados por la aplicación y dirigidos a una vista html en específico que veremos más adelante.

3. Archivo urls.py de la carpeta contacts

```
urls.py x
contacts > urls.py > ...
1 from django.urls import path
2
3 from contacts import views
4
5 urlpatterns = [
6     path('', views.ContactListView.as_view(), name='contact_list'),
7     path('new/', views.ContactCreateView.as_view(), name='contact_new'),
8     path('<int:pk>/edit/', views.ContactUpdateView.as_view(), name='contact_edit'),
9     path('<int:pk>/delete/', views.ContactDeleteView.as_view(), name='contact_delete'),
10 ]
```

Este archivo, estará contenido dentro de nuestra aplicación y no dentro de nuestro folder base, esto se aclara, porque más adelante veremos un archivo urls.py con una codificación distinta a esta.

Este archivo contiene las direcciones a cada uno de los html, con el propósito de realizar una acción específica de nuestro CRUD.

4. Archivo settings.py de la carpeta contactlist

Contactlist es mi folder de trabajo, este folder contiene la carpeta contacts, carpeta que contiene la aplicación.

Dentro de contactlist hay un archivo llamado settings.py este archivo contiene toda la configuración de la aplicación, bases de datos, dependencias, lenguaje, etc.

Revisaremos las más importantes:

```
import os

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure--vrwv&@-10xx6ry%y6-+xk41xvi3s(vdit2(3n3!6xoe#ucoed'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['localhost', '127.0.0.1']
```

Aquí tenemos la importación de dos librerías importantes y el host que usaremos para la conexión a la base de datos.

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'imagekit',

    'contacts',
]
```

Aquí agregamos las apps que son necesarias para la implementación, ejecución y diseño de nuestra aplicación.


```
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'ENFORCE_SCHEMA': False,
        'NAME': 'contacts',
        'CLIENT': {
            'host': 'clusterfordistarf.umww0lc.mongodb.net',
            'port': 27017,
            'username': 'distarf1',
            'password': 'distarf1',
            'authSource': 'contacts',
            'authMechanism': 'SCRAM-SHA-1',
        }
    }
}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Dentro de este mismo archivo de settings.py tendremos las conexiones a las bases de datos que se deseen implementar. En este caso, se ha implementado una conexión a Mongo DB y SQLite.

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'mediafiles')
MEDIA_URL = '/media/'
```

Se agregaron las carpetas que contendrán elementos claves, la carpeta static contendrá diseños, y en este caso, mediafiles contendrá las imágenes de nuestra aplicación.

```
# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'es-419'

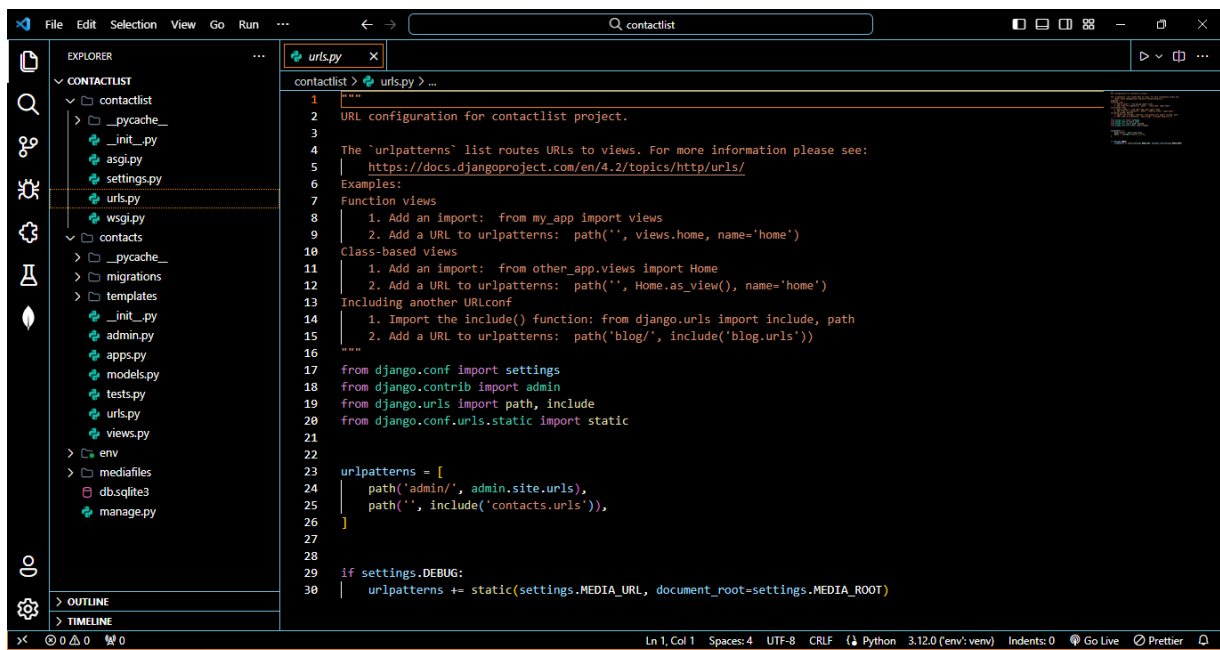
TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True
```

Aquí configuramos el idioma, pasándolo a español y algunas otras configuraciones.

5. Archivo urls.py de contactlist



```
1 """
2 URL configuration for contactlist project.
3
4 The 'urlpatterns' list routes URLs to views. For more information please see:
5 | https://docs.djangoproject.com/en/4.2/topics/http/urls/
6 Examples:
7 Function views
8 1. Add an import: from my_app import views
9 2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11 1. Add an import: from other_app.views import Home
12 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.conf import settings
18 from django.contrib import admin
19 from django.urls import path, include
20 from django.conf.urls.static import static
21
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', include('contacts.urls')),
26 ]
27
28
29 if settings.DEBUG:
30     urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

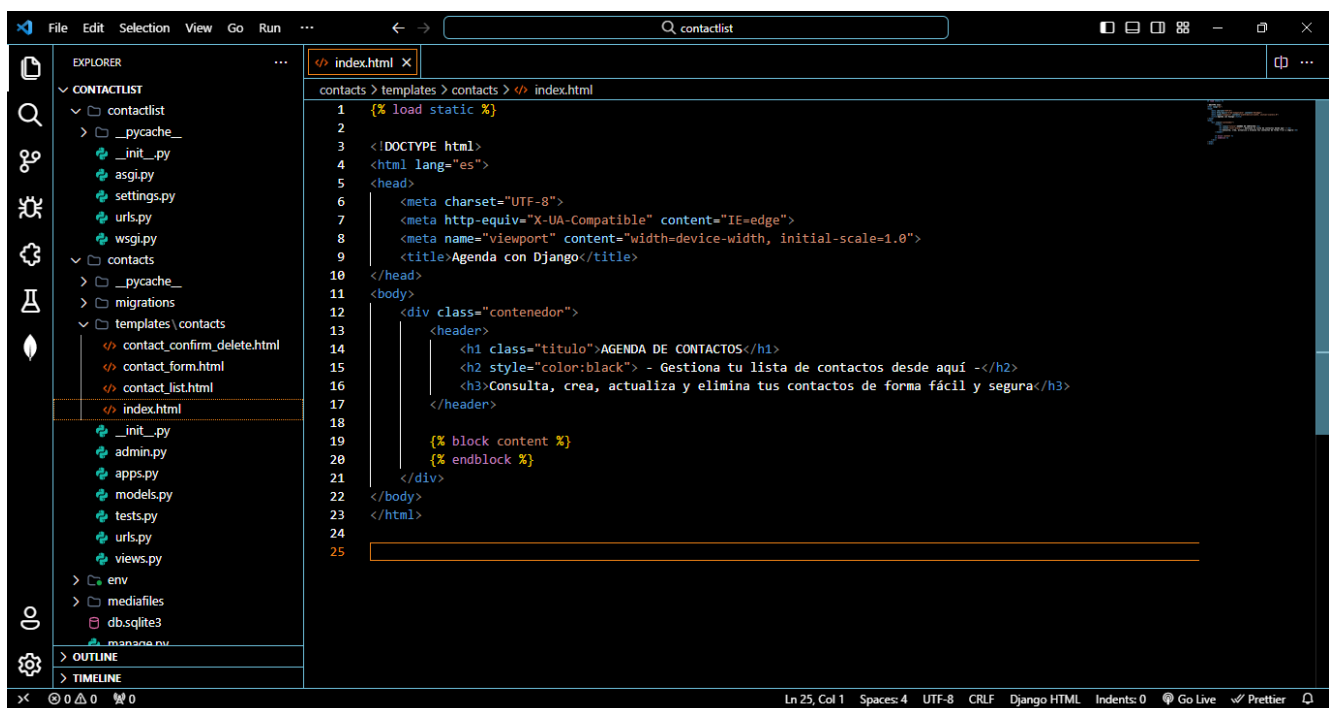
Este archivo de configuración está dentro de nuestro folder base, y contiene el enlace principal para ir a nuestra aplicación de manera directa. Y adicionalmente, la configuración de administración de django.

DESARROLLO DE INTERFACES PARA NUESTRA APLICACIÓN FINAL

1. Creación del folder templates, para los archivos html

Para la codificación de los archivos html, nos ubicaremos dentro de la carpeta de nuestra aplicación (contacts) y crearemos un folder llamado templates y crearemos otra subcarpeta llamada igual que nuestra aplicación (contacts). Aquí almacenaremos todos nuestros archivos.

2. Archivo index.html



```
1 {% load static %}
2
3 <DOCTYPE html>
4 <html lang="es">
5 <head>
6     <meta charset="UTF-8">
7     <meta http-equiv="X-UA-Compatible" content="IE=edge">
8     <meta name="viewport" content="width=device-width, initial-scale=1.0">
9     <title>Agenda con Django</title>
10 </head>
11 <body>
12     <div class="contenedor">
13         <header>
14             <h1 class="titulo">AGENDA DE CONTACTOS</h1>
15             <h2 style="color:black"> - Gestiona tu lista de contactos desde aquí -</h2>
16             <h3>Consulta, crea, actualiza y elimina tus contactos de forma fácil y segura</h3>
17         </header>
18         {% block content %}
19         {% endblock %}
20     </div>
21 </body>
22 </html>
23
24
25
```

Este archivo contiene la base de nuestra página, el encabezado que usaremos para las demás vistas y todo el enlace al contenido principal y recursos estáticos.

3. Archivo `contact_list.html`

```
<div>
  <br>
  <h2 style="color:#fff">Mis contactos</h2>

  <div>
    <form action="">
      <div class="form-container">
        <input type="text" class="form-control" placeholder="Buscar por nombre"
          |   |   |   name="q" value="{{ request.GET.q }}">
        <button class="btn btn-primary">
          |   Buscar
        </button>
      </div>
    </form>
  </div>
  <br>
  <a href="{% url 'contact_new' %}" class="btn btn-primary">
    |   REGISTRAR NUEVO CONTACTO
  </a>
</div>
```

El primer contenedor de este archivo contendrá la barra de búsqueda y el botón que nos permitirá crear un contacto nuevo, usando el método (`contact_new`).






```
<div>
  <table>
    <thead>
      <tr>
        <th></th>
        <th>Nombre</th>
        <th>Apellidos</th>
        <th>Email</th>
        <th>Fecha de nacimiento</th>
        <th>Teléfono</th>
        <th>Fecha de registro</th>
        <th></th>
      </tr>
    </thead>
    <tbody>
      {% for contact in object_list %}
      <tr>
        <td>
          <div>
            {% if contact.avatar %}
            
            {% endif %}
          </div>
        </td>
      </tr>
    </tbody>
  </table>
</div>
```

```

<td>
  <div>
    {% if contact.avatar %}
    
    {% endif %}
  </div>
</td>
<td>{{ contact.nombre|default:'-' }}</td>
<td>{{ contact.apellidos|default:'-' }}</td>
<td>{{ contact.email|default:'-' }}</td>
<td>{{ contact.fechanacimiento|default:'-' }}</td>
<td>{{ contact.telefono|default:'-' }}</td>
<td>{{ contact.created|date:'d/m/Y h:i A' }}</td>
<td>
  <div>
    <a href="{% url 'contact_edit' contact.pk %}" class="btn btn-primary">Editar</a>
    <a href="{% url 'contact_delete' contact.pk %}" class="btn btn-primary">Eliminar</a>
  </div>
</td>
</tr>
{% endfor %}
</tbody>
</table>
</div>
<br>

```

Esta es la codificación necesaria para traer y mostrar los datos que deseemos dentro de nuestra aplicación.

	Nombre	Apellidos	Email	Fecha de nacimiento	Teléfono	Fecha de registro	
	Disneyyna	Tarazona	disneyynatf@gmail.com	29 de marzo de 2004	304 6366186	25/11/2023 08:32 PM	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
	Alejandra	Mayorga	alejandramayorga@gmail.com	22 de mayo de 2004	321 7833815	25/11/2023 08:42 PM	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
	Jhon	Díaz	jhondiaz@gmail.com	17 de octubre de 2003	317 2202280	25/11/2023 08:43 PM	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
	Fabián	Luna	fabianluna@gmail.com	19 de noviembre de 2001	305 4586052	25/11/2023 08:43 PM	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
	Juliana	Caballero	julicaballero@gmail.com	9 de abril de 2004	312 3474437	25/11/2023 08:45 PM	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

El diseño aquí aplicado se realizó implementando hojas de estilos CSS, y algunas etiquetas adicionales de Bootstrap que no serán mostradas en este manual.

De ese modo, el diseño queda sujeto a gusto personal de cada uno.

4. Archivo contact_form.html

```
<body>
  <div class="mt-5">
    <h3>
      {% if contact %}
      Editar
      {% else %}
      Nuevo
      {% endif %}
      contacto
    </h3>
  </div>
  <form action="" method="post" enctype="multipart/form-data" novalidate>
    {% csrf_token %}

    {{ form }}

    <div class="btn-container">
      <a href="{% url 'contact_list' %}" class="btn btn-primary">
        Cancelar
      </a>
      <button class="btn btn-primary">
        {% if contact %}
        Actualizar
        {% else %}
        Crear
        {% endif %}
        contacto
      </button>
    </div>
  </form>
</body>
```

Este archivo nos permitirá implementar un formulario, que mostrará la solicitud de los datos establecidos en el modelo, permitiendo que se realice el registro de nuevos contactos en la aplicación, pero que así mismo, se muestre dicho formulario para la edición de los campos que se desee.

```
{% extends 'contacts/index.html' %}
{% load static %}
{% block content %}
```

Necesitaremos abrir la etiqueta de contenido y así mismo finalizarla, en el archivo.

```
{% endblock content %}
```

Para que el contenido se traiga de manera correcta, será necesario implementar el contenido de nuestra aplicación, a través de estas etiquetas propias de Python y Django.

■ Crear contacto

Nuevo contacto

Avatar:
 Ninguno archivo selec.

Nombre:

Apellidos:

Email:

Fecha de nacimiento:

Teléfono:

■ Editar contacto

Editar contacto

Avatar: Actualmente: [contact/img6.jpg](#) ☐

Limpiar
Modificar:
 Ninguno archivo selec.

Nombre:

Apellidos:

Email:

Fecha de nacimiento:

Teléfono:

Implementación del mismo formulario

5. Archivo contacts_confirm_delete.html

```
{% extends 'contacts/index.html' %}
{% load static %}
{% block content %}
```

```
<body>
  <div>
    <h3>Eliminar contacto</h3>
  </div>

  <div class="alert-container">
    <div class="alert">
      ¿Estás seguro de eliminar el contacto
      <span class="fw-bold">{{ contact.email }}</span>?

      <form method="post" class="mt-3">
        {% csrf_token %}
        <br>
        <a href="{% url 'contact_list' %}" class="btn btn-primary">
          Cancelar
        </a>
        <button class="btn btn-primary">
          Sí, eliminar
        </button>
      </form>
    </div>
  </div>
</body>
</html>

{% endblock content %}
```

En este archivo, codificaremos la vista que nos permitirá aceptar la eliminación de un contacto. Este será el archivo destino al momento de dar clic sobre el botón eliminar en alguno de nuestros registros.

AGENDA DE CONTACTOS

- Gestiona tu lista de contactos desde aquí -

Consulta, crea, actualiza y elimina tus contactos de forma fácil y segura

Eliminar contacto

¿Estás seguro de eliminar el contacto
julicaballero@gmail.com?

CancelarSí, eliminar

De este modo queda completa nuestra aplicación. Ya sabemos que podemos visualizar los datos ingresado a través de SQLite o Mongo DB. Esto, gracias a las dos conexiones realizadas.

EJECUCIÓN DE NUESTRA APLICACIÓN FINAL

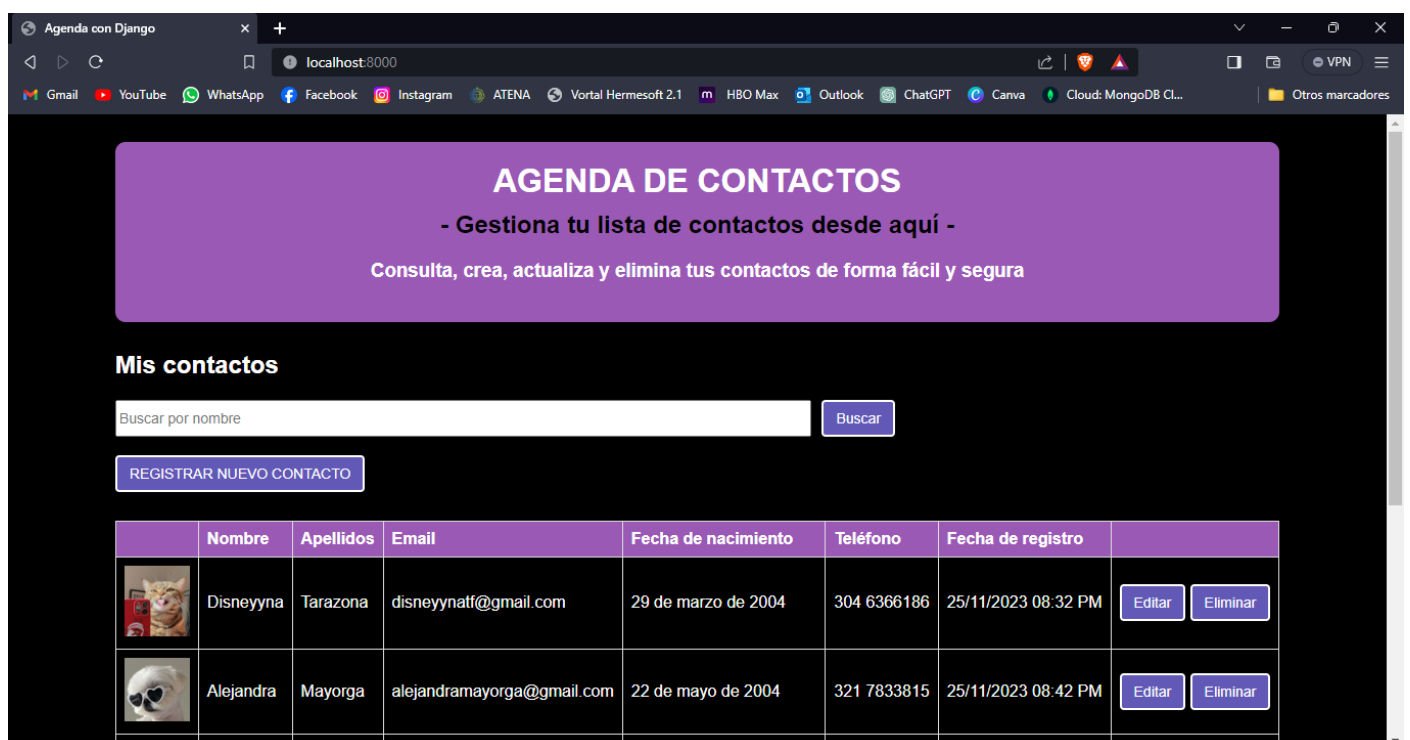
Para la ejecución de nuestra aplicación, abriremos el cmd de nuestra computadora, desde el folder de nuestra aplicación y ejecutaremos el siguiente comando: *python manage.py runserver* . Este comando nos permitirá subir la aplicación para visualizarla a través del localhost:

```
C:\Windows\System32\cmd.e x + v - □ X

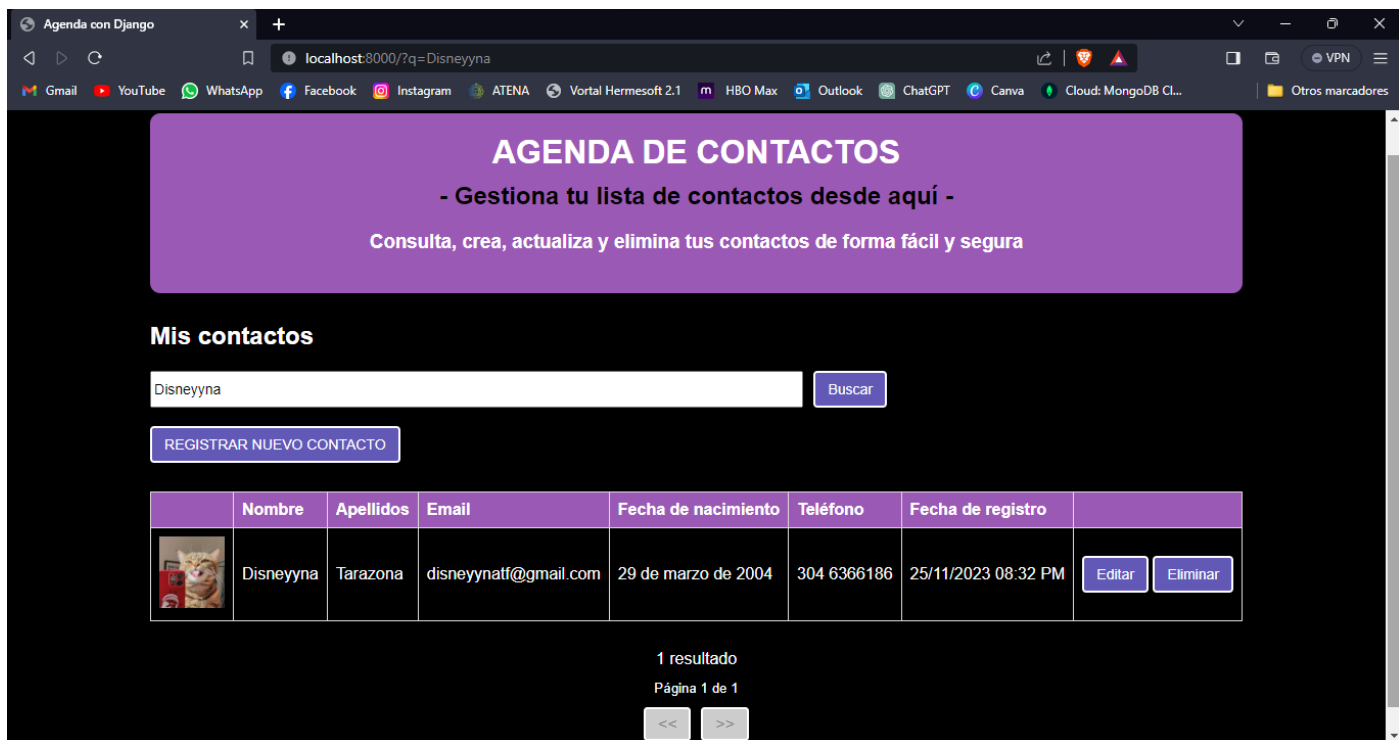
C:\ProyectoDjangoPython\contactlist>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 25, 2023 - 22:48:07
Django version 4.1.13, using settings 'contactlist.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
|
```

Lo siguiente que haremos será ir al navegador y escribiremos localhost:8000. Este será el puerto que nos permitirá acceder a Django.



Esta será la interfaz inicial de nuestra aplicación (Diseño propio).



Al momento de realizar este tipo de consultas, podremos ver que en la consola seguirá ejecutándose el proyecto y evidenciando las acciones realizadas.

```
C:\Windows\System32\cmd.e X + v

System check identified no issues (0 silenced).
November 25, 2023 - 22:48:07
Django version 4.1.13, using settings 'contactlist.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
C:\Users\disne\AppData\Local\Programs\Python\Python312\Lib\site-packages\django\views\generic\list.py:91: UnorderedObjectListWarning: Pagination may yield inconsistent results with an unordered object_list: <class 'contacts.models.Contact'> QuerySet.
  return self.paginator_class(
[25/Nov/2023 22:49:22] "GET / HTTP/1.1" 200 11715
[25/Nov/2023 22:50:17] "GET /?q=Disneyyna HTTP/1.1" 200 7968
[25/Nov/2023 22:51:03] "GET / HTTP/1.1" 200 11715
[25/Nov/2023 22:51:05] "GET /27/edit/ HTTP/1.1" 200 5222
[25/Nov/2023 22:51:07] "POST /27/edit/ HTTP/1.1" 302 0
[25/Nov/2023 22:51:07] "GET / HTTP/1.1" 200 11715
[25/Nov/2023 22:51:09] "GET /?page=2 HTTP/1.1" 200 11708
[25/Nov/2023 22:51:09] "GET /media/contact/img8.jpg HTTP/1.1" 304 0
[25/Nov/2023 22:51:09] "GET /media/contact/img2.jpg HTTP/1.1" 304 0
[25/Nov/2023 22:51:09] "GET /media/contact/img7.jpg HTTP/1.1" 304 0
[25/Nov/2023 22:51:09] "GET /media/contact/img1.jpg HTTP/1.1" 304 0
[25/Nov/2023 22:51:09] "GET /media/contact/img5.jpg HTTP/1.1" 304 0
[25/Nov/2023 22:51:11] "GET /?page=1 HTTP/1.1" 200 11715
```

Esto, porque para que se lleve a cabo la ejecución correcta de nuestra aplicación, Python y Django deben estar corriendo desde la consola de nuestra computadora.

Con esto finalizamos la explicación de como elaborar una aplicación para almacenar contacto, realizando las cuatro acciones básicas del manejo de datos (Create – Read – Update – Delete) CRUD.

Muchas gracias.