**Documentación Creación App Crm Django**

**Pasos:**

**Paso 1: Crear una Aplicación**

Para crear una aplicación dentro de tu proyecto Django, ejecuta el siguiente comando en la terminal (asegúrate de estar en el directorio raíz de tu proyecto):

python manage.py startapp crm

python manage.py: Este comando se utiliza para ejecutar varias tareas administrativas en tu proyecto Django. manage.py es un script que Django genera automáticamente cuando creas un nuevo proyecto y se encarga de interactuar con tu proyecto.

startapp: Esta es la subcomando que le dice a Django que deseas crear una nueva aplicación.

crm: Es el nombre que le das a tu aplicación. Puedes llamarla como quieras, pero en este caso, le estamos llamando "crm".

¿Qué sucede en este paso?

Se crea un nuevo directorio llamado crm dentro de tu proyecto, que contiene la estructura básica de una aplicación Django. Esta estructura incluye archivos y directorios necesarios para el funcionamiento de la aplicación.

**Paso 2: Estructura de la Aplicación**

Dentro del directorio crm, encontrarás varios archivos:

\_\_init\_\_.py: Indica que este directorio debe ser tratado como un paquete Python.

admin.py: Aquí puedes registrar modelos para que aparezcan en la interfaz de administración de Django.

apps.py: Contiene la configuración de la aplicación.

models.py: Aquí defines tus modelos (clases que representan la estructura de tu base de datos).

tests.py: Para escribir pruebas unitarias para tu aplicación.

views.py: Aquí defines las vistas, que son funciones o clases que manejan la lógica de negocio y devuelven respuestas HTTP.

**Paso 3: Registrar la Aplicación**

Debes registrar tu nueva aplicación en el archivo de configuración del proyecto. Abre settings.py (que se encuentra en el directorio del proyecto) y busca la sección INSTALLED\_APPS. Agrega el nombre de tu aplicación:

INSTALLED\_APPS = [

...,

'crm',

]

¿Qué hace este paso?

Al agregar 'crm' a INSTALLED\_APPS, le estás diciendo a Django que reconozca esta aplicación y que debe incluirla en las operaciones de la base de datos, la administración, etc.

**Paso 4: Definir Modelos**

Ahora, en models.py, puedes definir tus modelos. Aquí hay un ejemplo simple de un modelo Cliente:

from django.db import models

class Cliente(models.Model):

nombre = models.CharField(max\_length=100)

email = models.EmailField()

telefono = models.CharField(max\_length=15)

def \_\_str\_\_(self):

return self.nombre

models.Model: Tu clase Cliente hereda de models.Model, lo que indica que es un modelo de base de datos.

CharField y EmailField: Estas son definiciones de campo que indican el tipo de datos que se almacenarán (cadena de caracteres, email, etc.).

\_\_str\_\_: Este método define cómo se representará el objeto cuando lo conviertas a una cadena (por ejemplo, en la interfaz de administración).

¿Qué sucede en este paso?

Estás definiendo la estructura de la tabla Cliente en la base de datos. Cada atributo en la clase representa una columna en la tabla.

Paso 5: Crear Migraciones

Después de definir tus modelos, debes crear migraciones, que son instrucciones para crear o modificar las tablas de la base de datos:

python manage.py makemigrations

¿Qué hace este paso?

Django genera archivos de migración que describen los cambios realizados en los modelos. Estos archivos se guardan en un directorio migrations dentro de tu aplicación.

**Paso 6: Aplicar Migraciones**

Una vez que has creado las migraciones, aplica esos cambios a la base de datos ejecutando:

python manage.py migrate

¿Qué hace este paso?

Aplica las migraciones pendientes y crea las tablas correspondientes en la base de datos. En este caso, se creará la tabla Cliente.

**Paso 7: Acceder a la Interfaz de Administración**

Para ver tus modelos en la interfaz de administración, primero necesitas registrarlos en admin.py. Agrega lo siguiente a crm/admin.py:

from django.contrib import admin

from .models import Cliente

admin.site.register(Cliente)

¿Qué hace este paso?

Le dice a Django que registre el modelo Cliente en la interfaz de administración, lo que permitirá gestionar instancias de este modelo desde allí.

**Paso 8: Crear un Superusuario**

Para acceder a la interfaz de administración, necesitas crear un superusuario. Ejecuta el siguiente comando:

python manage.py createsuperuser

Se te pedirá que ingreses un nombre de usuario, un correo electrónico y una contraseña.

¿Qué hace este paso?

Crea un usuario con permisos de administrador que puede acceder a la interfaz de administración de Django.

**Paso 9: Ejecutar el Servidor**

Finalmente, para ver tu aplicación en acción, ejecuta el servidor de desarrollo:

python manage.py runserver

¿Qué hace este paso?

Inicia un servidor web de desarrollo que te permite ver tu aplicación en el navegador. Por defecto, estará disponible en http://127.0.0.1:8000/.

**Paso 10: Acceder a la Interfaz de Administración**

Abre tu navegador y ve a http://127.0.0.1:8000/admin.

Inicia sesión con el superusuario que creaste.

Una vez dentro, podrás ver y gestionar el modelo Cliente.

**Base de datos relacional de la pagina**

**Usuarios creados**

Admin

## Usuario: nicolas

email: nicoladmoraled123

contra: Nicolas3

## Usuario: Edward

Contra: moralesmo1