Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Configuración Inicial del Entorno

Creación del Entorno Virtual:

Abrimos la terminal en Visual Studio Code.

Ejecutamos el comando python -m venv .env para crear un entorno virtual llamado .env.

Activamos el entorno con source .env/Scripts/activate en Windows. La terminal mostrará el nombre del entorno activado como prefijo.

Texto

Descripción generada automáticamente

 Clonación del Repositorio:

Clonamos el repositorio del proyecto desde GitHub con git clone https://github.com/flatplanet/Django-CRM.

Este comando descarga todos los archivos necesarios para el proyecto.



Texto

Descripción generada automáticamente

Estructura del Proyecto:

Observamos la estructura de carpetas en Visual Studio Code para verificar que el proyecto se haya clonado correctamente.

En la estructura tenemos el directorio principal Django-CRM con carpetas para la aplicación (website) y configuración (dcrm).

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Configuración de la Base de Datos en PostgreSQL

Creación del Usuario y la Base de Datos en PostgreSQL:

Creamos un usuario llamado joaodev con contraseña 12345.

Creamos la base de datos CRMBD asignando a joaodev como propietario.

Otorgamos todos los privilegios sobre CRMBD al usuario joaodev.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Instalación de Dependencias

Instalación de Django y psycopg2:

Instalamos Django ejecutando pip install django.

Instalamos el adaptador psycopg2-binary necesario para la conexión de Django con PostgreSQL con pip install psycopg2-binary.

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Ejecución del comando python manage.py makemigrations para crear las migraciones en Django. La terminal indica "No changes detected", lo que significa que no hay nuevos cambios en los modelos para migrar en este momento.

Texto

Descripción generada automáticamente

Ejecución del comando python manage.py migrate para aplicar las migraciones en la base de datos. La terminal muestra que todas las migraciones, incluidas auth, contenttypes, sessions, y website, se aplicaron correctamente.

Texto

Descripción generada automáticamente

 Visualización de las tablas generadas en la base de datos CRMBD después de aplicar las migraciones, usando la interfaz de administración de PostgreSQL. Se observan tablas como auth\_group, auth\_permission, django\_admin\_log, django\_session, y website\_record, que corresponden a los modelos creados en el proyecto Django.



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Inserciones en la tabla website\_record con datos de ejemplo. Se muestran varias filas insertadas con valores en columnas como first\_name, last\_name, address, city, state, zipcode, phone, email, y created\_at.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ejecución del comando python manage.py createsuperuser para crear un superusuario en Django. El usuario joaodev es creado con el correo joaovaldiglesias@gmail.com y una contraseña definida por el usuario.

Texto

Descripción generada automáticamente

Proceso de creación del superusuario joaodev, donde se establece el correo electrónico y se genera una contraseña. La terminal también muestra advertencias sobre la contraseña, indicando que es corta, común y numérica, pero se fuerza la creación del superusuario.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

 Ejecución del comando python manage.py runserver para iniciar el servidor de desarrollo de Django en 127.0.0.1:8000. El servidor se ejecuta correctamente y está listo para recibir conexiones.



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Pantalla de inicio de sesión de la aplicación Django CRM, donde el superusuario joaodev ingresa su nombre de usuario y contraseña para acceder al sistema.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Vista de la página principal de la aplicación después de iniciar sesión. Se muestra una lista de registros con columnas como Name, Email, Phone, Address, City, State, Zipcode, Created At, e ID, junto con un mensaje de éxito que indica que el usuario ha iniciado sesión.

Ejecución de la consulta SQL ALTER SEQUENCE website\_record\_id\_seq RESTART WITH 501; en la interfaz de PostgreSQL. Esta consulta reinicia la secuencia del campo id de la tabla website\_record para que comience desde 501 en adelante.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

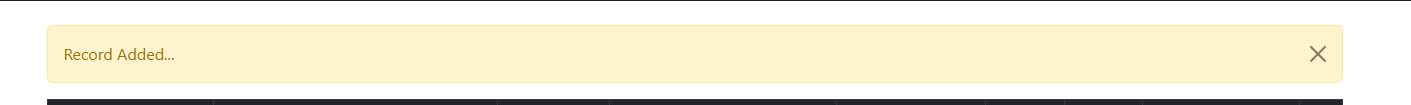
Descripción generada automáticamente

Formulario de adición de un nuevo registro en la aplicación. Los campos incluyen First Name, Last Name, Email, Phone, Address, City, State, y Zipcode. Se muestra la información de un ejemplo de usuario ingresado para el registro.

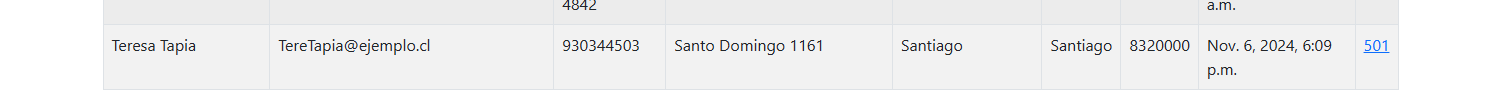
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Mensaje de confirmación que indica "Record Added..." después de agregar un nuevo registro en la base de datos exitosamente. Este mensaje confirma que el registro fue agregado y muestra una notificación en la parte superior de la pantalla.



Visualización del nuevo registro agregado en la tabla principal de registros en la aplicación. Los detalles incluyen el nombre, correo electrónico, teléfono, dirección, ciudad, estado, código postal, y el ID asignado (en este caso, 501) al nuevo registro.



Para proponer las búsquedas y filtros de manera interactiva y un usuario que no conozca nada sobre rutas y apis propongo implementar lo siguiente:  
modificar home.html:

Código HTML del formulario de filtros en home.html dentro de Django, donde se crean campos de texto para State y City, así como casillas de verificación para filtrar nombres que "empiecen con A y terminen en e" y que "contengan 'ma'". El formulario permite realizar búsquedas avanzadas en los registros.  
Texto

Descripción generada automáticamente

Código de la tabla de registros en el archivo home.html. Esta tabla muestra los datos de los registros con columnas como Name, Email, Phone, Address, City, State, Zipcode, Created At, e ID. Cada fila corresponde a un registro con sus datos específicos y el ID tiene un enlace a los detalles de cada registro.

Texto

Descripción generada automáticamente

Código JavaScript para manejar la limpieza de filtros en la vista principal. Este script elimina el contenido de los campos de filtro cuando se selecciona otro filtro para evitar múltiples filtros simultáneos, asegurando que sólo un criterio de búsqueda se use a la vez.

Captura de pantalla de computadora

Descripción generada automáticamente

Código de la función home(request) en el archivo views.py de Django. La función permite aplicar filtros de búsqueda según los parámetros recibidos en la solicitud GET, como state, city, name\_start\_end, y name\_contains. La función devuelve los registros filtrados a la plantilla home.html.

Texto

Descripción generada automáticamente

Página de resultados en la aplicación, donde se aplicó el filtro de estado "TX" para mostrar solo los registros de Texas. Se muestran las columnas de datos de los registros filtrados.

Captura de pantalla de un celular

Descripción generada automáticamente

Página de resultados de la aplicación donde se ha aplicado un filtro de City igual a "Dallas" y State igual a "TX". La tabla muestra únicamente los registros de empleados que residen en la ciudad de Dallas, Texas. Las columnas incluyen Name, Email, Phone, Address, City, State, Zipcode, Created At, e ID.

Imagen que contiene Tabla

Descripción generada automáticamente

Página de resultados en la aplicación con el filtro de City establecido en "Austin" y State en "TX". La tabla muestra solo los registros de empleados de Austin, Texas. Los datos incluyen detalles como nombre, correo electrónico, teléfono, dirección y más.

Tabla

Descripción generada automáticamente con confianza media

Página de resultados en la aplicación con el filtro City configurado en "Chicago". La tabla muestra exclusivamente los registros de empleados de Chicago, Illinois. Las columnas cubren datos básicos como nombre, correo electrónico, dirección, y otros detalles relevantes de los empleados.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Página de resultados que aplica el filtro para "Name Starts with A and Ends with e", mostrando solo registros de personas cuyos nombres empiezan con "A" y terminan en "e". La tabla incluye registros de empleados de diversas ciudades y estados con nombres que cumplen el criterio.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Página de resultados que aplica el filtro "Name Contains 'ma'", mostrando los registros donde el nombre contiene la secuencia "ma" en alguna parte del mismo. La tabla incluye detalles completos de los empleados filtrados.

Captura de pantalla de computadora

Descripción generada automáticamente