



Formando líderes para la construcción
de un nuevo país en paz

DESARROLLO WEB DE IMAGEN, INFORMACIÓN Y SERVICIOS INSTITUCIONALES DE COLOMBIAN KIF SAS

Autor

Andres Fernando Mogollon Cabeza

C.C 1007618597

Director

Avilio Villamizar Estrada

Ingeniero de sistemas

INGENIERÍA DE SISTEMAS

FACULTAD DE INGENIERÍA Y ARQUITECTURA

2022-2



Tabla de contenido

1. Introducción	4
2. Planteamiento del problema	4
3. Justificación	5
4. Objetivos	5
Objetivo General	5
Objetivos específicos	5
5. Acotaciones	5
6. Marco teórico	6
Páginas web corporativas	6
Python	6
Framework	6
Django	7
Modelo-Vista-Template	7
ORM de django	8
Postgresql	9
Bootstrap	10
Git	10
Github	11
Cloudinary	11
Sendgrid	11
Render	11
7. Resumen del proyecto	11



8. Desarrollo	12
Información de la empresa	12
Entidad relación	12
Entorno virtual	13
Proyecto django	14
Instalacion y creacion de proyecto	14
Archivos estáticos en django	15
Aplicación del proyecto	15
Plantilla base	17
Home	17
Quienes Somos	19
Servicios	20
Blog	21
Contacto	22
Despliegue del proyecto	23
Alojamiento de imágenes	23
Personalizar de errores	24
PostgreSQL en render	25
Despliegue de django en render	25
Configuración de dominio	26
Final	27
9. Conclusiones	27
10. Biografías	28
11. Anexos	29



1. Introducción

Las páginas web en la actualidad son instrumentos muy utilizados en el mundo por organizaciones, instituciones, casas y empresas, que tienen como propósito, comunicar, expresar, publicitar, informar y ahora hasta interactuar, se caracterizan por contar con conectividad a través de hipervínculos que hacen que el documento sea más dinámico, además de ser accesible, cualquier página web cuenta con información compilada que posteriormente se presenta mediante un ordenador que está conectado a internet. (Guzmán Y Valle et al., 2018)

Ahora bien, el desarrollo de páginas web ha tomado gran impacto en los últimos años, gracias a las grandes ventajas que ofrece a los usuarios que interactúan con ella, siendo multiplataforma y accesibles desde cualquier punto, facilitan el trabajo colaborativo, están disponibles en cuanto a hora y lugar, mantienen centralizada la información, ahorran dinero y tiempo. (Guzmán Y Valle et al., 2018) Su importancia radica en el proceso de lectura, ya que posibilita y visualiza la información en distintos formatos (texto, imagen, sonidos, videos, animaciones). (Palacios et al., 2018) Por lo cual, las páginas web se han convertido en una fuente de información primaria para clientes y otros públicos con un grado de credibilidad elevado que lleva a muchos usuarios a obtener información como paso previo al adquirir algún producto o servicio, ya que les permite seleccionar de forma más activa, oportuna, actualizada y con fácil acceso. (DUEÑAS et al., 2015)

Es así que las empresas a nivel mundial que cuentan con una página web han tomado grandes ventajas, ya que les permiten un mayor posicionamiento en el mercado, Además de ser más competitivos, ofreciendo sus productos y servicios en la red, el sitio web con el que deben contar las páginas tiene que ser de alta calidad en colores y agradable a la vista del usuario. Por último algunos beneficios de la creación de sitios web es que acarrea bajos costos económicos y agilidad en los procesos de la empresa, dándose a conocer en el mundo desde la comodidad de sus hogares y trabajos.

2. Planteamiento del problema

En los últimos años las empresas se encuentran ofertando sus productos y servicios a través de páginas web, permitiéndoles llegar a más clientes de forma dinámica fidelizándolos, incrementando sus ventas, impactando y posicionando nuevos mercados, es así que las empresas que no cuentan con esta herramienta tienen desventajas competitivas en comparación a las que disponen de un sitio web. Por lo cual surge la problemática de la empresa colombiana KIF SAS siendo una empresa dedicada al cultivo y producción de cannabis medicinal con ubicación principal en Pamplona y sede operacional en Bochalema, actualmente esta empresa no cuenta con una herramienta que los usuarios puedan utilizar accediendo a la web a través de internet.



3. Justificación

Este proyecto se realizará con el fin de solucionar la problemática de la falta de herramientas informáticas que permitan la constante relación con los clientes, siendo necesario enfrentar las necesidades que se presenten, el crecimiento competitivo del mercado y los aspectos de posicionamiento de la empresa colombian kif SAS, por esta razón es indispensable el desarrollo de una aplicación web que pueda ser accedida a través de internet, reflejando la información institucional, productos ofrecidos, contacto, panel de administración y blog de noticias relevantes de la empresa mediante el uso de la herramienta Python con ayuda del framework Django.

4. Objetivos

Objetivo General

Desarrollar una aplicación web de imagen, información y servicios institucionales, aumentando la visualización y el crecimiento competitivo en el mercado de la empresa Colombian kif SAS mediante el uso de Python empleando el framework Django.

Objetivos específicos

Diseñar las bases de datos relacionales necesarias mediante la creación de un modelo entidad relación.

Implementar las urls, vistas y modelos necesarios para la página web que contenga información institucional, servicios ofrecidos, contacto, blog y panel de administración de la empresa colombian Kif SAS.

Realizar las pruebas, configuraciones y despliegue de la aplicación web llevándola a un entorno productivo.

5. Acotaciones

Las herramientas que se van a utilizar en el proyecto de desarrollo de una web corporativa son Python, Bootstrap, Git y un servicio que permita almacenar toda la información de la web en un servidor, con el fin de cumplir con los objetivos planteados, actualmente se cuenta con un dominio para la aplicación web.

6. Marco teórico

Páginas web corporativas

Las páginas web corporativas son las webs diseñadas para describir y representar a una empresa, marca de internet o negocio. Los elementos básicos que tienen estos tipos de páginas son: página de inicio, página



para dar a conocer al negocio, página del equipo de trabajo, página de productos y/o servicios y página de contacto, adicionalmente también puede contar con portafolios, blogs, tiendas virtuales entre otros (Guzmán Y Valle et al., 2018)

Python

Python fue creado como sucesor del lenguaje ABC, las principales características con las que cuenta python son:

- Es un lenguaje de programación interpretado y multiplataforma cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
- Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.
- Se trata de un lenguaje de propósito general.

Framework

El propósito de los framework es normalizar y estructurar el código del sistema, facilitando un esquema para el desarrollo y/o la implementación de aplicaciones. El uso de frameworks para cualquier tipo de desarrollo reduce el tiempo de elaboración e implementación y ayuda a hacer un trabajo mantenible y escalable, según las características del mismo.

Un framework agrega funcionalidad extendida a un lenguaje de programación, automatiza muchos de los patrones de programación para orientarlos a un determinado propósito, proporcionando una estructura al código, mejorándolo, haciéndolo más entendible, sostenible, y permite separar en capas la aplicación.

En general, divide la aplicación en tres capas:

- La lógica de presentación que administra las interacciones entre el usuario y el software.
- La Lógica de datos que permite el acceso a un agente de almacenamiento persistente u otros.
- La lógica de dominio o de negocio, que manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.

Django

Django es un marco de trabajo para el desarrollo de aplicaciones web usando Python, django considera algunas funcionalidades listas para usar facilitando el desarrollo de aplicaciones web. Como resultado, no es necesario escribir todo el código ni usar tiempo para buscar errores de código en el framework. Es decir, mediante Django, el desarrollo de sistemas de información web puede ser rápido, seguro, escalable y también fáciles de mantener. (Vidal-Silva et al., 2021a)



Los marcos de trabajo de desarrollo web tales como PHP soportan un patrón de diseño MVC (Model-View-Controller) de arquitectura de software para desarrollar aplicaciones web. Django implementa el concepto de MTV (Model-Template-View) que es ligeramente diferente de MVC.

Django posibilita definir proyectos y aplicaciones para sistemas de información web. Cuando se define un proyecto Django, automáticamente se genera una serie de archivos, y uno de los más relevantes es settings.py. El archivo setting.py se utiliza para definir variables de configuración del sistema web tales como las aplicaciones que incluye, parámetros de conexión a base de datos, idioma, entre otros. Una vez que se establezca un proyecto Django, es posible definir aplicaciones para dicho proyecto. Para cada aplicación de proyecto Django, se pueden crear modelos que representan tablas en la base de datos establecida (archivo models.py) junto con parámetros de visualización de estos modelos. (SANCHEZ & SILVA, 2017)

Además, en un proyecto Django se pueden incluir plantillas con extensión .html para la visualización de datos, además de plantilla de administración para operaciones CRUD para los modelos (tablas) que se indiquen en el archivo admin.py.

Modelo-Vista-Template

El modelo vista template es una variación del modelo vista controlador que fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales están dadas por el hecho de que, el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. Es así que en ambos patrones, el modelo representa la capa de acceso a datos. La vista en MVC representa la interfaz de usuario, mientras la plantilla de MTV representa la interfaz de usuario. El controlador de MVC es el encargado de definir la lógica de negocio, mientras en Django la vista representa una capa intermedia para definir dicha lógica. La principal diferencia entre MVC y MTV es que, en este último, Django se encarga de tareas propias del controlador (para controlar las interacciones entre el modelo y la vista).

El modelo define los datos almacenados, se encuentra en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros. Todo esto permite indicar y controlar el comportamiento de los datos. (Vidal-Silva et al., 2021b)

La vista se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados, entre otras cosas más. La vista también se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos y la validación de datos a través de formularios. Lo más importante a entender con respecto a la vista es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla.

La plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solamente crea contenido en HTML (también XML, CSS, Javascript, CSV, etc).



ORM de django

Django permite, de manera predeterminada, utilizar una capa de mapeo relacional de objetos para la interacción con aplicaciones Django con bases de datos relacionales

El ORM de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista.

Por lo cual, permite utilizar una interfaz sencilla para crear, leer, modificar y eliminar objetos. Django ORM contiene por defecto:

- Un mecanismo para serializar y validar datos de formularios HTML para almacenamiento en la base de datos
- Un sistema de serialización que puede producir y leer representaciones XML y JSON de instancias de modelos,
- Un mecanismo de protección contra ataques CSRF e inyecciones de SQL.

Es indispensable establecer que el CSRF: (falsificación de solicitud entre sitios) es un ataque en el que un programa malicioso envía comandos del nombre de un usuario que el sistema conoce y en el que confía. Las víctimas de este ataque pueden ser tanto los usuarios como el sistema.(Tehnică Moldovei, n.d.)

Esto significa que no tiene que preocuparse por la base de datos subyacente que está utilizando, o incluso si desea cambiar de una base de datos a otra, puede hacerlo sin cambiar una sola línea de la lógica de su aplicación, solo cambie la cadena de conexión a la base de datos desde settings.py , descargue los datos antiguos y debería estar listo(Tehnică Moldovei, n.d.)

Además, Django ORM es compatible con los siguientes sistemas de gestión de bases de datos: PostgreSQL, MySQL, SQLite, Oracle, Microsoft SQL Server. Una versión no oficial también es compatible con MongoDB

Postgresql

PostgreSQL es un sistema de gestión de bases de datos orientado a objetos y relacional que hace énfasis en la extensibilidad y conformidad con los estándares, este se ha convertido en una de las bases de datos de elección en infinidad de proyectos, proporcionando al usuario algunas prestaciones del mismo nivel que las que ofrecen sistemas gestores de bases de datos comerciales como Informix y Oracle.

Las características más destacadas de PostgreSQL son:

- Soporte para transacciones.
- Subconsultas.
- Soporte de vistas.
- Integridad referencial.



- Herencia de tablas.
- Tipos definidos por el usuario.
- Columnas como vectores que pueden almacenar más de un valor.
- Añadir campos a tablas en tiempo de ejecución.
- Funciones de agregación (como sum() o count()) definibles por el usuario.
- Diseño web responsive o adaptativo

El diseño web responsive es una técnica para lograr la correcta visualización de un sitio web en las pantallas de los diferentes dispositivos donde se carga la aplicación, sean estos computadores de escritorio, computadores portátiles, smartphone, tablets y otros. El diseño responsive también es conocido como diseño adaptativo, por la adaptación que se logra de los formularios webs sin esfuerzos de programación, ya que hoy en día existen herramientas de software que permiten lograr ese objetivo.(Orrala & Castro, 2018)

Es así, que este enfoque quiere centrarse en el contenido, y en el cliente, en su experiencia de usuario, si deja de trabajar con su equipo de sobremesa y quiere continuar navegando en la misma página web desde una tablet o smartphone. Responsive Web Design quiere eliminar la necesidad de diseños diferentes y nuevos desarrollos para distintas resoluciones y por el contrario, sugiere que nuestro desarrollo debe dar soporte y responder a la necesidad del contexto sobre el que se esté ejecutando, teniendo en cuenta parámetros como el tamaño de pantalla, el tipo de dispositivo o la orientación.

Además al crear un sitio con Responsive Web Design solo necesitamos una única versión de HTML y CSS que funcionará adecuadamente en cualquier tipo de dispositivo y resolución.

Bootstrap

Bootstrap es un popular Framework front-end que fue desarrollado como un producto de código abierto que fue lanzado inicialmente para emplearlo en Twitter. Es un proyecto que está basado en los estándares de CSS, JavaScript y HTML. Permite personalizar el desarrollo frontal de las aplicaciones según las necesidades ya que puede incluir funciones tanto de CSS y JavaScript como formularios, botones, menús y otros componentes necesarios facilitando su desarrollo. Bootstrap fue creado basado en el concepto “First Mobile” debido a que el avance tecnológico fue incrementando y existía una gran demanda de dispositivos móviles inteligentes desde los cuales se accedía a los diferentes sitios web, pero debido al tamaño de pantalla del dispositivo los sitios alteraban su interfaz provocando malestar en los usuarios, es así que Bootstrap gracias a su arquitectura basada en diseño responsivo permite desarrollar sitios flexibles que se adapten a todo dispositivo.(Ulloa Revelo, 2020)

Entre las características más relevantes de Bootstrap se tiene:



- **Diseño Responsivo:** Es una de las características más importantes ya que permite redimensionar automáticamente la interfaz web a los diferentes dispositivos móviles facilitando así no perder funcionalidad.
- **Documentación:** Bootstrap cuenta con mucha información en línea en donde se detalla el funcionamiento de sus diferentes componentes, permitiendo que los desarrolladores puedan utilizarlos de una manera más fácil y sencilla.
- **Estilos globales:** Aquí se tiene una serie de elementos que vienen pre-compilados, dando un tratamiento especial para la interfaz del sitio web.
- **Sistema Grid:** Bootstrap utiliza el método denominado sistema de rejillas que permite diseñar una interfaz principalmente para los dispositivos móviles.
- **CSS y JavaScript:** Cuenta con hojas de estilo CSS y una librería JavaScript que brinda elementos adicionales que utilicen los componentes de manera ágil. (Ulloa Revelo, 2020)

Git

Git es un sistema avanzado de control de versiones distribuido. Git permite rastrear el progreso de un proyecto a lo largo del tiempo ya que hace capturas del mismo a medida que evoluciona y los cambios se van registrando. Esto permite ver qué cambios se hicieron, quién los hizo, e incluso volver a versiones anteriores. Además, Git facilita el trabajo en paralelo de varios participantes. Mientras que en otros sistemas de control de versiones hay un servidor central y cualquier cambio hecho por un usuario se sincroniza con este servidor y de ahí con el resto de usuarios, Git es un control de versiones distribuido que permite a todos los usuarios trabajar en el proyecto paralelamente e ir haciendo capturas del trabajo de cada uno para luego unirlos. (Astigarraga & Cruz-Alonso, 2022)

Github

GitHub es un servidor de alojamiento en línea o repositorio remoto para albergar proyectos basados en Git que permite la colaboración entre diferentes usuarios o con uno mismo. Un repositorio es un directorio donde desarrollar un proyecto que contiene todos los archivos necesarios para el mismo. Aunque existen distintos repositorios remotos con funcionalidad similar, GitHub es hoy en día el más utilizado. GitHub registra el desarrollo de los proyectos de manera remota, permite compartir proyectos entre distintos usuarios y proporciona la seguridad de la nube entre otras funciones. Cuando se trabaja en proyectos colaborativos, la base de la interacción entre Git y GitHub es que todos los colaboradores de un proyecto están de acuerdo en que GitHub contiene la copia principal del proyecto, es decir, GitHub contiene la copia centralizada del control de versiones distribuido o descentralizado. (Astigarraga & Cruz-Alonso, 2022)



Cloudinary

Es un servicio de API que permite el almacenamiento de medios como imágenes y videos. En lugar de almacenar los archivos en la base de datos de la aplicación, almacena los archivos en Cloudinary y, a su vez, este proporciona una URL pública para ver la imagen o el archivo.(Yaremko, 2022)

Sendgrid

SendGrid es un servicio de correo electrónico basado en la nube que ofrece un sistema confiable de entrega de correo electrónico transaccional, escalabilidad y análisis en tiempo real junto con API flexibles que facilitan la integración personalizada.(*Email Delivery, API, Marketing Service* | *SendGrid*, 2022)

Render

Render es una nube unificada para crear y ejecutar todas sus aplicaciones y sitios web con certificados TLS gratuitos, una CDN global, protección DDoS, redes privadas e implementaciones automáticas desde Git.(*Cloud Application Hosting for Developers* | *Render*, n.d.)

7. Resumen del proyecto

El proyecto se centra en el desarrollo de una aplicación web corporativa que está diseñada para describir y representar a la empresa colombiana Kif SAS. Los elementos principales de esta página son: página de inicio, página para dar a conocer la empresa, página de productos y/o servicios, página de contacto y blog de noticias. Siendo esta implementada con el lenguaje de programación Python usando el framework django. Además, llevándola a un entorno de producción en el cual pueda ser accedida a través de una conexión a internet.

8. Desarrollo

Para llevar a cabo la práctica con la empresa colombiana Kif SAS se tuvo en cuenta las siguientes actividades que realizan durante el desarrollo del semestre permitiendo así el cumplimiento de los objetivos anteriormente planteados:

1. Obtener la información necesaria de la empresa para el desarrollo del proyecto.
2. Diseño de las bases de datos relacionales según corresponda.
3. Implementación de vistas, url, modelos y migraciones que sean necesarios para la página de información institucional.
4. Implementación de vistas, url, modelos y migraciones que sean necesarios para la página de servicios.
5. Implementación de vistas, url, modelos y migraciones que sean necesarios para la página de contacto.
6. Implementación de vistas, url, modelos y migraciones que sean necesarios para la página de blog.



7. Realizar las configuraciones necesarias para el panel de administración y llevar a un entorno de producción la aplicación.
8. Informe de los resultados finales.

teniendo en cuenta las actividades planteadas se comenzó con el desarrollo de la práctica de la siguiente manera

Información de la empresa

En la primera actividad se obtuvo la información necesaria para el desarrollo del proyecto, donde la empresa proporcionó los datos institucionales que permiten comenzar con el desarrollo de la página.

Entidad relación

Una vez obtenida la información necesaria para el desarrollo del proyecto se prosiguió a la siguiente actividad, en la cual se diseñó la base de datos relacional con el uso de la herramienta DbVisualizer generando el siguiente diagrama de entidad relación:

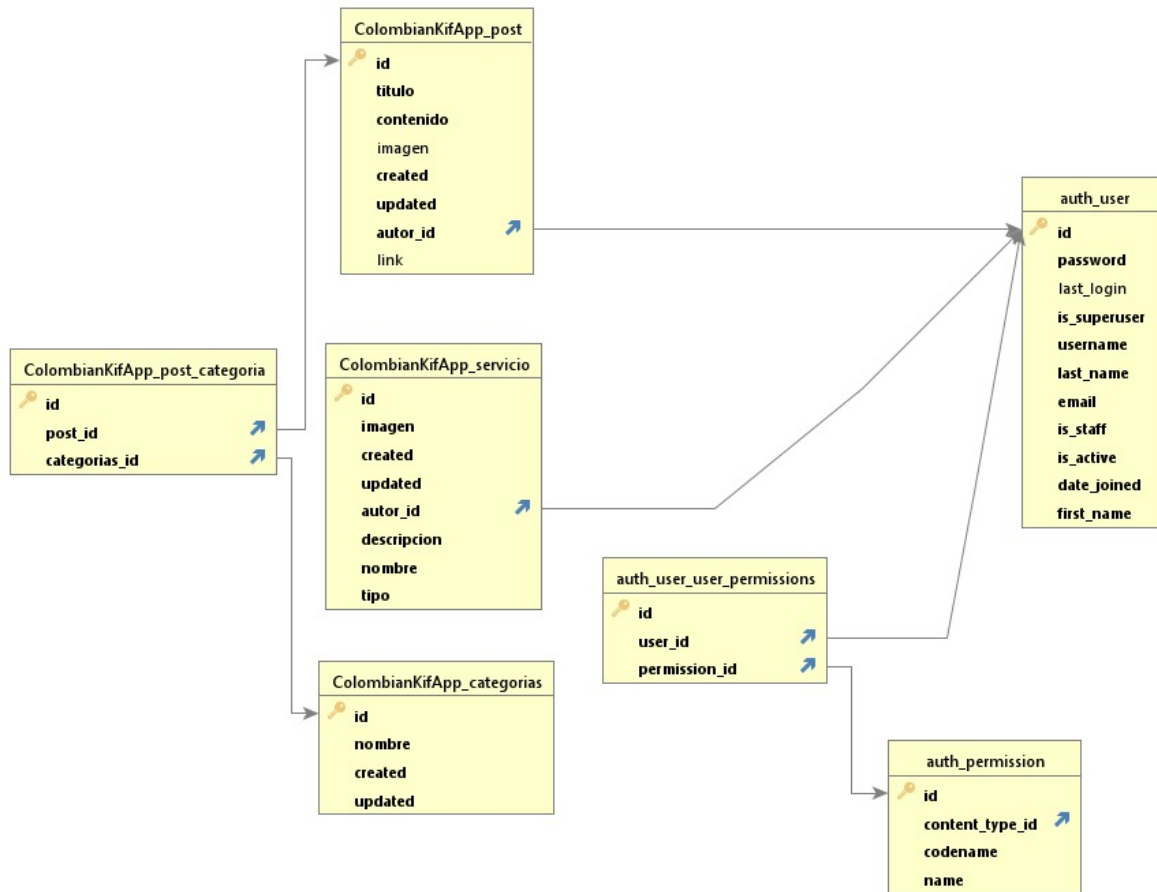


Figura 1: Diagrama entidad-relación. Fuente: autor

Entorno virtual

Posteriormente se creó un entorno virtual usando python venv con el fin tener un espacio donde podemos instalar paquetes específicos para el proyecto. Este permite tener un conjunto de paquetes/librerías aislados de la instalación principal de Python en nuestro sistema. El entorno virtual se elaboró con el comando `python -m venv colombianKif` este genera las carpetas y archivos como se observa en la Figura 2. Para activar el entorno virtual se usa el comando `.\colombianKif\Scripts\Activate.ps1`

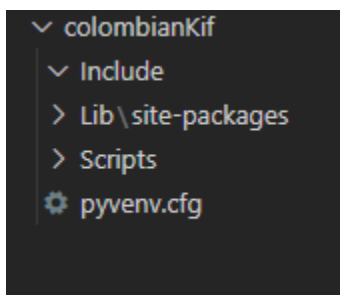


Figura 2: Entorno virtual Fuente: El autor

Proyecto django

Instalacion y creacion de proyecto

Teniendo el entorno virtual activo se hizo la instalación de django con “pip install django” para luego crear un nuevo proyecto usando este framework con “django-admin startproject colombianKif” el cual nos genera los archivos base para la aplicación, el que se puede observar en la Figura 3

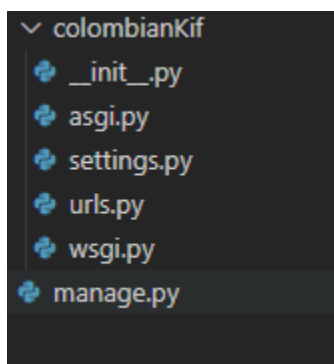


Figura 3: Archivos base django Fuente:El autor

Además, al iniciar el servidor con ayuda del archivo manage.py y el comando “python manage.py runserver” vemos que django nos creó una página base que corre en el puerto 8000 como se contempla en la Figura 4

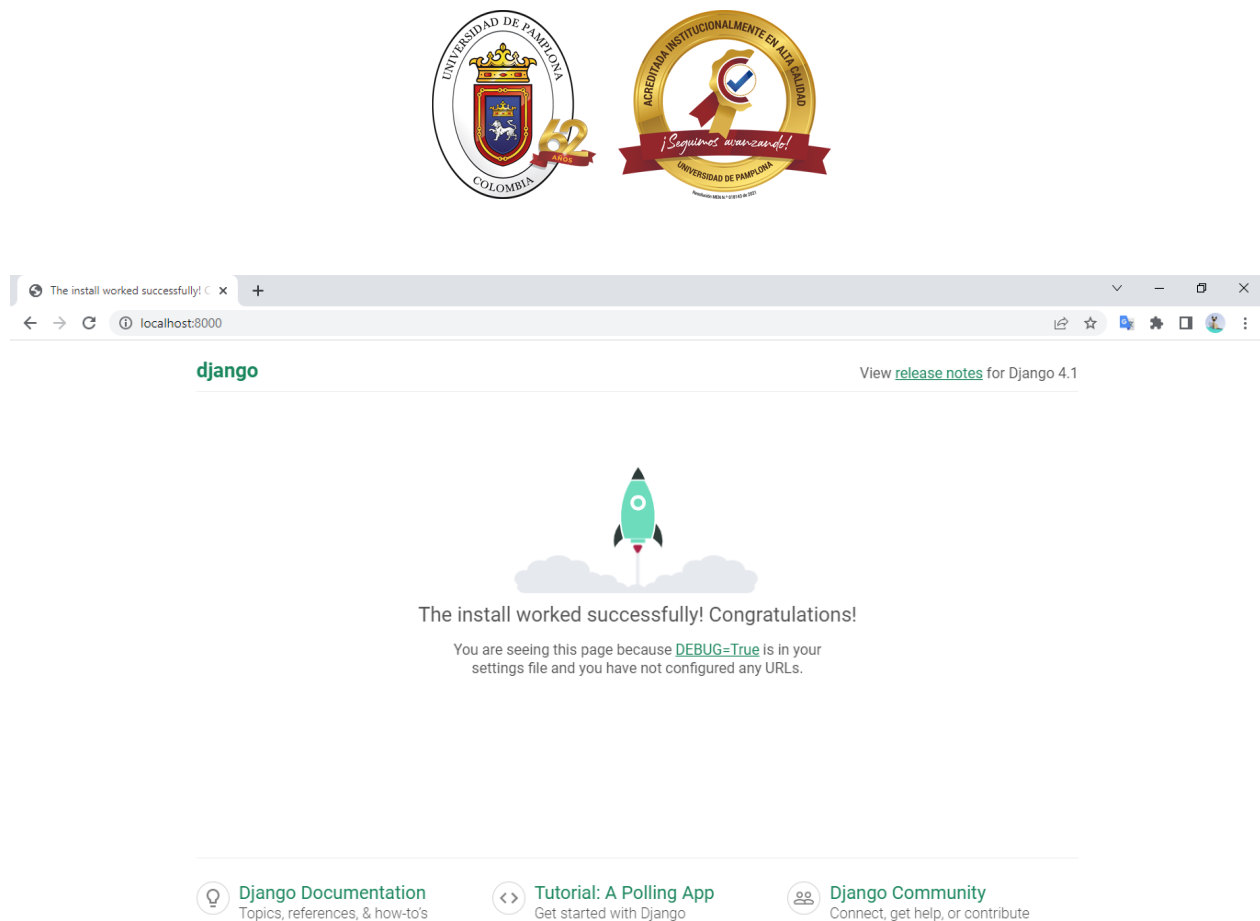


Figura 4: Inicio del servidor, Fuente: El autor

Archivos estáticos en django

se creó una carpeta en la raíz del proyecto con el nombre de static, la cual va a contener todos los archivos estáticos de la aplicación y debe definirse en el archivo settings.py

Aplicación del proyecto

se generó la aplicación colombiankifApp con el comando “python manage.py startapp colombianKifApp” quien va a contener las vistas, modelos, formularios y plantillas de la página web. Después de crear la aplicación se configuró el archivo settings.py, como se observa en la Figura 5 y 6

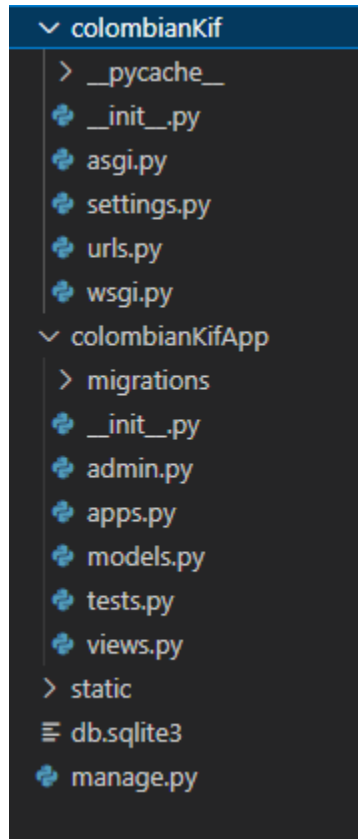


Figura 5: se genera la aplicación, Fuente:El autor

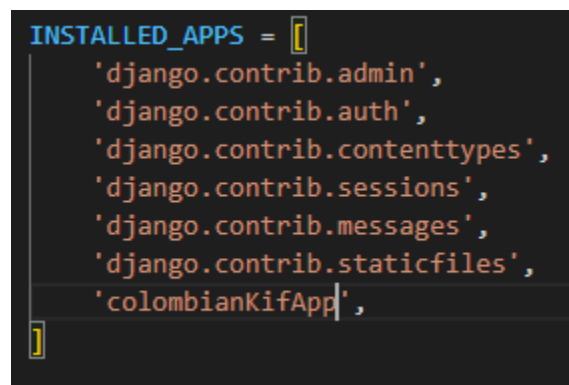


Figura 6: Configuración, Fuente:El autor



Plantilla base

En colombianKifApp se creo la carpeta templates y en ella el archivo base.html, luego en views.py se creo una función que será la vista, la cual retorna un indicador a la plantilla html, por último se configura el archivo urls.py de la aplicación y de configuración, el código utilizado se puede observar en el anexo 1,11 y 15.

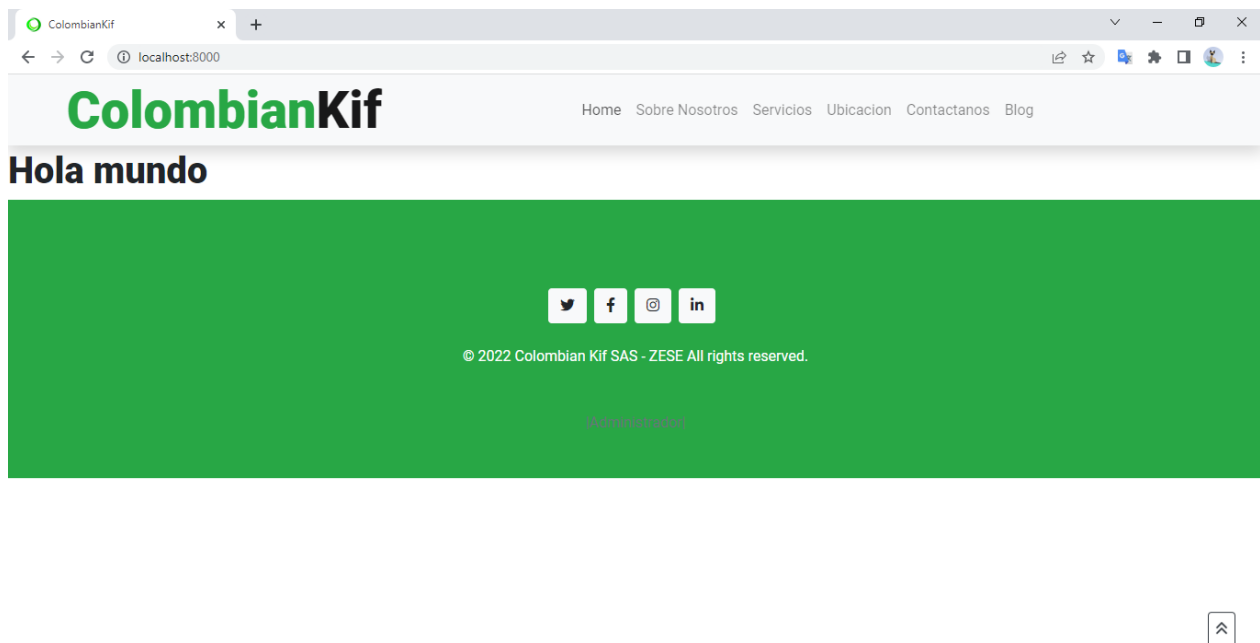


Figura 7:html base, Fuente:El autor

Home

Dentro de la carpeta templates creamos una subcarpeta denominada Home esta contiene el archivo home.html, posteriormente se uso el cargador de plantillas de django en el archivo base.html con {% include "Home/home.html"%} de esta manera incluimos la plantilla home en el contenido de la base como se observa en la figura 8. el código utilizado se puede observar en el anexo 1 y 6.

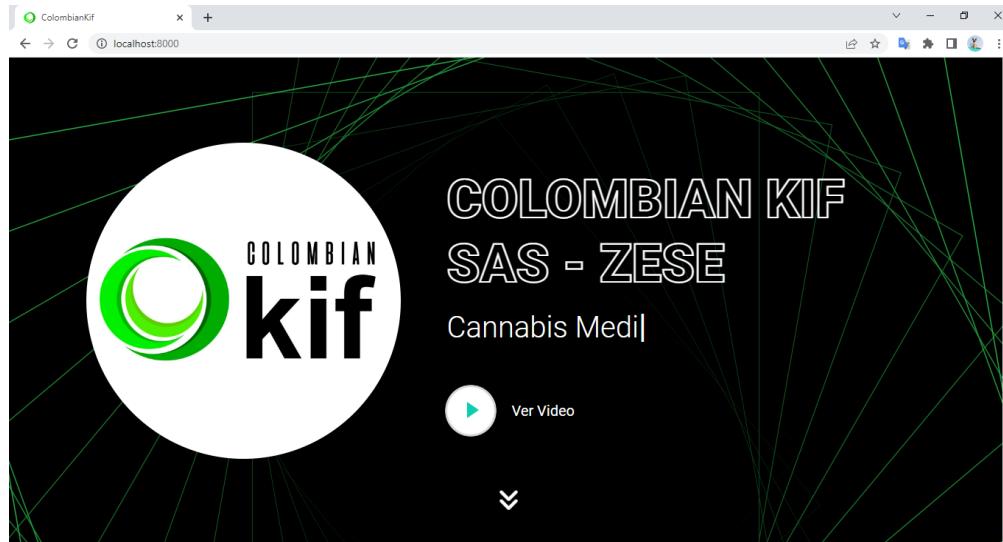


Figura 8: Página de inicio, Fuente:El autor

Ahora bien, para el botón ver video de home se usa un modal de bootstrap el cual inserta un video de facebook, representado en la figura 9

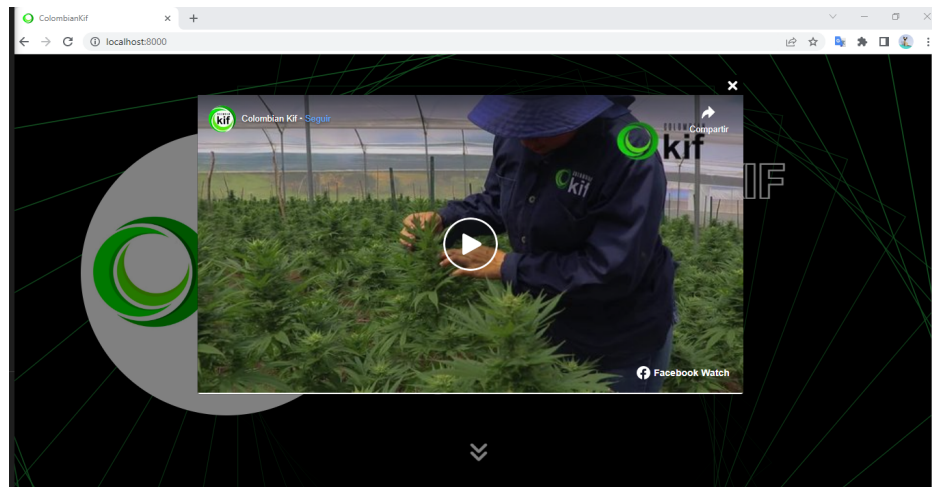


Figura 9:Video de Facefacebook, Fuente:El autor



Quienes Somos

De igual manera se creó en templates las subcarpetas ¿Quienes Somos? con los archivos mision_vision.html y objetivos.html y Ubicación con ubicacion.html y se incrustan en el archivo base.html con `{% include "QuienesSomos/mision_vision.html" %}` y `{% include "Ubicacion/ubicacion.html"%}` de esta manera usando los include podemos ir acomodando nuestro sitio web como se observa en la figura 10 y 11. el código utilizado se puede observar en el anexo 1, 2, 4 y 5



Figura 10:Mision y vision, Fuente:El autor



Figura 11:Ubicaciones, Fuente:El autor



Servicios

En cuanto a los servicios inicialmente se modificó el archivo models.py, en donde se agregó en forma de clases cada clase representa una tabla, para este caso se creó la clase servicios que contiene una llave foránea hacia el modelo User el cual ya está integrado desde django, para poder cargar imágenes se instaló “pip install pillow”. También se modificó el archivo admin.py, para que un usuario con permisos pueda hacer el crud este debe iniciar sesión con un correo y contraseña válidas. El código utilizado se puede observar en el anexo 17 y 18.

Para que las tablas se vean reflejadas en la base de datos se necesita hacer las migraciones, estas se realizaron con los comandos “python manage.py makemigrations” y “python manage.py migrate” esto genera las tablas en la base de datos que se establezca en la cadena de conexión de settings.py

Por medio de la siguiente gráfica se observa como en el panel de administración ya se encuentra con el modelo de servicios

The screenshot shows the administration interface for 'COLOMBIANKIFAPP'. At the top, there's a header with the app name and a '+ Add' button. Below this is a section titled 'Añadir servicio ó producto'. The form includes fields for 'Nombre:' (with a placeholder 'Nombre del servicio ó producto'), 'Descripción:' (with a placeholder 'Descripción del servicio ó producto'), 'Tipo:' (a dropdown menu), 'Autor:' (a dropdown menu with a '+ Add' icon and placeholder 'Quien añade el producto'), and 'Imagen:' (a file selection button labeled 'Seleccionar archivo' and placeholder 'Ninguno archivo selec. foto del servicio ó producto'). At the bottom, there are three buttons: 'Grabar y añadir otro', 'Grabar y continuar editando', and 'GRABAR'.

Figura 12 : Panel de administración Fuente:El autor



una vez creado correctamente el modelo y la migraciones, en view.py se ideó una función que con ayuda del orm de django, retorna todos los servicios que están registrados, la función se ejecuta desde la vista principal, la cual envía estos servicios como parámetros a la plantilla, para esto creamos la plantilla servicios.html dentro de una subcarpeta en template, se debe usar una etiqueta for de django para iterar los servicios que recibe, la etiqueta for tiene una estructura de `{% for dato in datos %} {% endfor %}` teniendo el template listo se incrusta en la plantilla base. El código utilizado se puede observar en el anexo 1, 3, 11 y 13

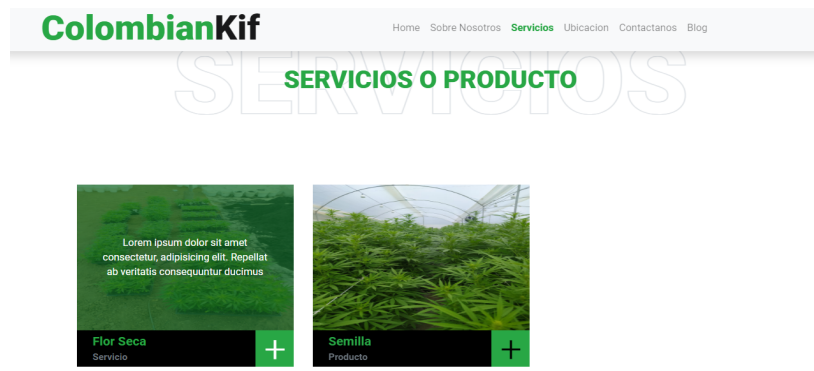


Figura 13: Servicios o productos Fuente:El autor

Blog

De manera similar se generó el blog, para su realización primero se debe editar el archivo models.py, agregar los modelos post y categoría como clases de python, luego agregarlos a el archivo admin.py y hacer las migraciones. El código utilizado se puede observar en el anexo 16 y 18.

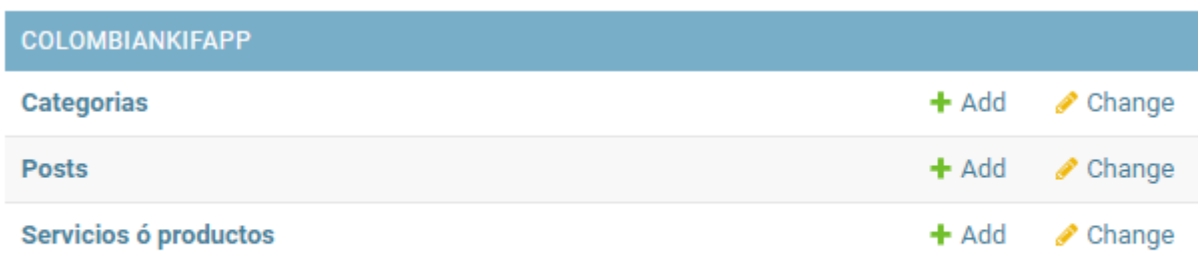


Figura 14:Blog , Fuente:El autor



Posteriormente se origina en view.py una función que retorna todos los post y los envía a la plantilla blog.html como parámetro, la plantilla debe crearse en la carpeta templates en una subcarpeta llamada Blog, esta carpeta utiliza el for de django para iterar el arreglo de blogs y a su vez otro for anidado para iterar las categorías de cada blog y de esta manera mostrar la información en el html, esta plantilla se debe incrustar en base.html. El código utilizado se puede observar en el anexo 1, 10 y 13



Figura 12: Blog, Fuente: El autor

Contacto

Para la página de contacto creamos un archivo llamado forms.py, en este implementamos una clase de python que contiene los campos del formulario y el orm de django nos ayuda a crear, validar y proteger de ataques la información, este formulario se inicializa y envía como parámetro a la plantilla, para esto en templates se realizó una subcarpeta Contactanos con la plantilla contactanos.html en la etiqueta form se debe agregar la función de django `{% csrf_token %}` para su correcto funcionamiento. El código utilizado se puede observar en el anexo 9, 11, 12 y 19.

Por su parte, para el envío de correo se usa el servicio web sendgrid este nos proporciona un key para poder enviar correos automáticamente en forma de html, para usar este servicio se debe registrar en sendgrid e instalar con “pip install sendgrid”.

En view.py se creó una función que se encarga de hacer el envío de correos mediante sendgrid esta función recibe como parámetros el nombre, correo, asunto y contenido y retorna un booleano. El código utilizado se puede observar en el anexo 11.



Mediante el método post verificamos que el usuario solicitó el envío de correo posteriormente validamos los datos y se llama a la función enviar correo que devuelve un booleano este al ser exitoso nos redirecciona a la ruta ?éxito, esta ruta es validada en el template con {% if %} {% endif %} y al cumplirse la condición, este ejecuta el código que se encuentra dentro del condicional.

Figura 13: Contactanos, Fuente: El autor

Despliegue del proyecto

Alojamiento de imágenes

Por su parte, en la producción no se desea guardar las imágenes en la base de datos ni en el servidor, por lo cual se usó un servicio web de alojamiento de imágenes y archivos llamado cloudinary, para poder utilizar este servicio se necesita “pip install django-cloudinary-storage”, además de hacer la configuración en settings.py y en urls.py, cloudinary nos genera las carpetas para almacenar las imágenes de los modelos que establecimos como se representa en la figura 14. El código utilizado se puede observar en el anexo 15, 20 y 22

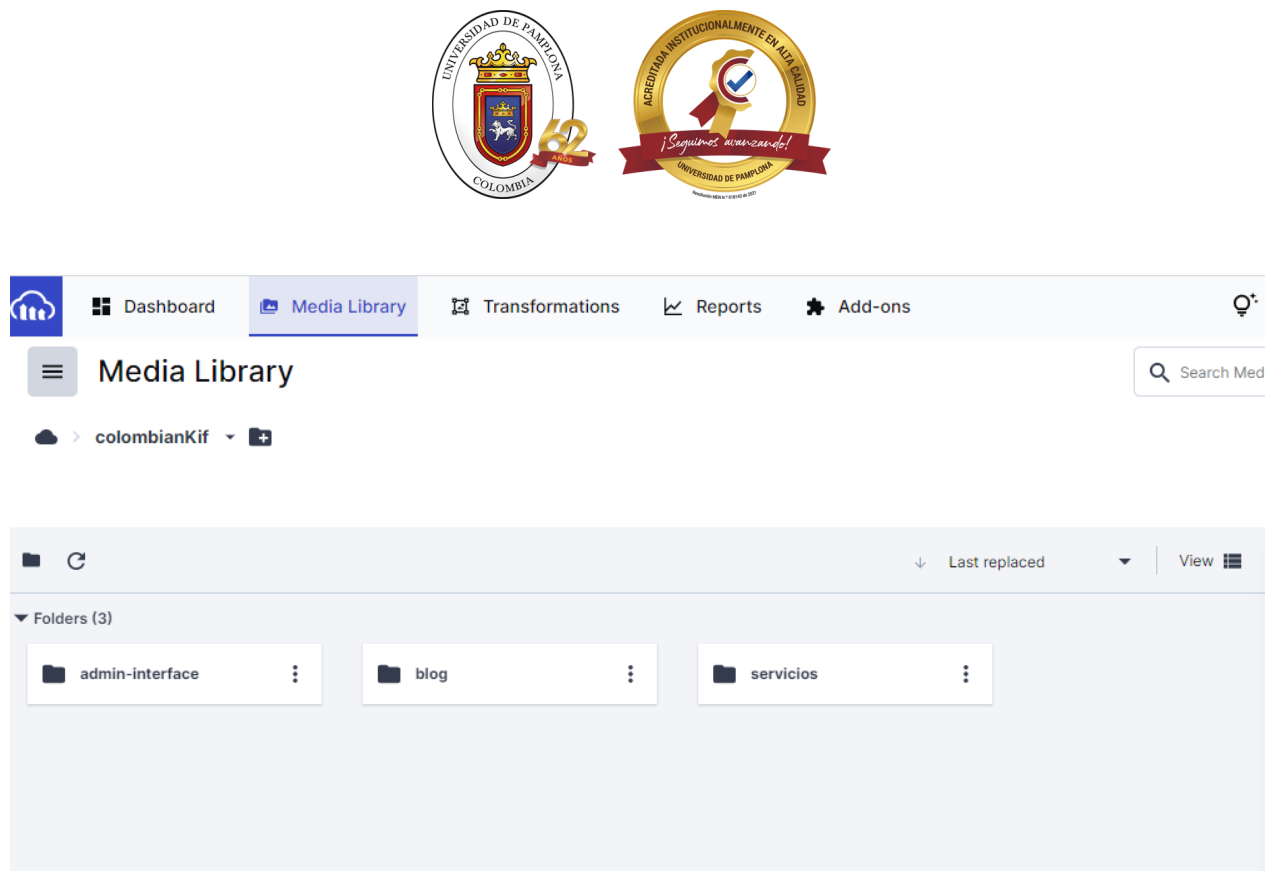


Figura 14:Alojamiento, Fuente:El autor

Personalizar de errores

Para la personalización del error 404 y 500 django nos proporciona las funciones handler404, handler500 estás necesitan de dos vistas una para cada uno que se debe implementar en la aplicación, y en el archivo urls.py se deben inicializar ambos errores apuntando a las vistas, estos solo serán visualizados en el entorno de producción. El código utilizado se puede observar en el anexo 7, 8, 14 y 15.

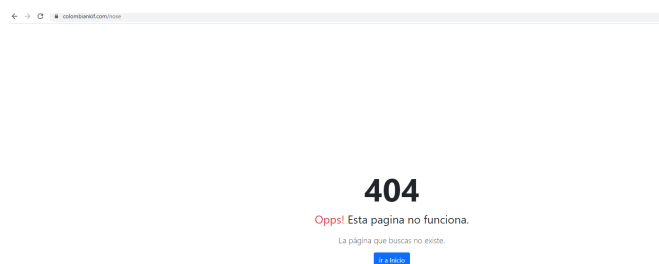


Figura 15: Error , Fuente:El autor



PostgreSQL en render

Ahora bien, al momento de desplegar la aplicación se utilizó render.com como, primero se creó una base de datos postgresql la cual es la que se va a usar en producción, esta genera un enlace para conectarla a la aplicación posteriormente, el cual se debe añadir en el archivo settings para poder conectarse correctamente y poder hacer las migraciones. El código utilizado se puede observar en el anexo 21.

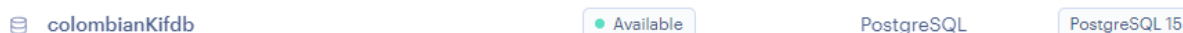


Figura 15:Render bd, Fuente:El autor

Despliegue de django en render

Además, para subir el proyecto a render se hizo unas configuraciones, en primera instancia las de setting.py, debug, secret key y demás datos sensibles los cuales se configuraron para que sean leídos desde variable de entorno del servidor, también se configuró la cadena de conexión a la base de datos creada en postgres para esto se instaló dj-database-url y psycopg2-binary, así como también whitenoise facilitándonos la carga de archivos estáticos, se creó un archivo que contiene los paquetes y sus versiones necesarios para que el proyecto corra correctamente, esto se hizo con el comando “pip freeze > requirements.txt”, para finalizar se estableció un ejecutable llamado build.sh el cual nos permitió hacer la instalación de los paquetes y las migraciones a la base de datos en el servidor web. El código utilizado se puede observar en el anexo 20, 21, 22, 23 y 24.

Posteriormente se subió el proyecto a github con ayuda de git ya que render nos pide conectarnos a el repositorio del proyecto que queremos desplegar, teniendo esto listo creamos un web service en render y conectamos el repositorio, establecemos el nombre del servidor, ejecutamos el build.sh y con gunicorn le decimos a render cuál es el archivo de inicio del proyecto, por último añadimos las variables de entorno requeridas y le dimos a create web service, de esta manera ya se encuentra la página web publicada.



Figura 16:Render web, Fuente:El autor

Configuración de dominio

por último se configuró el dominio para ello nos dirigimos a la aplicación en render y vamos a settings en esta encontraremos custom domain y agregamos nuestro dominio personalizado, esto nos dio los datos de conexión de dns como se observa en la figura 16



Custom Domains

Your server is always available on <https://sendgrid.onrender.com>.

You can also point **custom domains** you own to this app. See [DNS configuration instructions](#).

www.colombiankiff.com
redirects to colombiankiff.com

● DNS update needed [Verify](#)

Add a CNAME record for **www** pointing to **sendgrid.onrender.com**

DNS configuration instructions

colombiankiff.com

● DNS update needed [Verify](#)

Delete

Add an ANAME or ALIAS record pointing to **sendgrid.onrender.com**

If your DNS provider does not support ANAME or ALIAS records, use an A record pointing to **216.24.57.1**

DNS configuration instructions

Add Custom Domain

Figura 16: Configuración Dominio , Fuente: El autor

Es así que nos dirigimos a nuestro proveedor de dominios y configuramos el dns cambiando el CNAME y el ANAME al configurarlo correctamente render nos muestra la vista que se observa en la figura 17 la cual nos informa que los certificados han sido emitidos

Custom Domains

Your server is always available on <https://colombiankiff.onrender.com>.

You can also point **custom domains** you own to this app. See [DNS configuration instructions](#).

colombiankiff.com

Verified Certificate Issued

Delete

www.colombiankiff.com
redirects to colombiankiff.com

Verified Certificate Issued

Add Custom Domain

Figura 17: Configuración Fuente: El autor

Final

Para finalizar, el sitio web puede ser accedido a través de internet ingresando a el siguiente link: www.colombiankiff.com



Figura 18: link de la página Fuente:El autor

Una vez culminada la página se hizo entrega del informe final donde se dio a conocer como se realizó el desarrollo de la página así como también las herramientas con las cuales cuenta, cumpliendo así los objetivos propuestos, ejecutando la práctica de forma favorable y además mejorando aspectos importantes dentro de la empresa colombian kif SAS

9. Conclusiones

Para terminar se dio cumplimiento a la creación de la aplicación web con el lenguaje de programación Python empleando el framework Django, siendo accedida a través de internet contando con páginas que dan soporte a la información institucional, servicios ofrecidos, blog y contacto con la empresa, además de un panel de administración de la misma.

Finalmente el desarrollo de la página web benefició a la empresa colombian Kif SAS permitiéndole tener una mejor relación con los clientes favoreciendo al crecimiento competitivo y posicionamiento, así como también se beneficio el acceso a la imagen, información y servicios institucionales. Además la práctica realizada implementó y amplió los conocimientos y habilidades en el área de desarrollo web.

10. Biografías

Astigarraga, J., & Cruz-Alonso, V. (2022). ¡Se puede entender cómo funcionan Git y GitHub! *Ecosistemas*, 31(1). <https://doi.org/10.7818/ECOS.2332>

Cloud Application Hosting for Developers | *Render*. (n.d.). Retrieved November 29, 2022, from <https://render.com/>



DUEÑAS, P., VEGA GONZALEZ, C., & TERAN FRANCO, J. (2015). *La interactividad de las webs en las pequeñas y medianas empresas*. 3(1012–1587), 735–750.

Email Delivery, API, Marketing Service | SendGrid. (n.d.). Retrieved November 29, 2022, from <https://sendgrid.com/>

Guzmán Y Valle, E., Máter, A., Magisterio, D., Facultad, N., Ciencias, D. E., Muñoz, P., & Raquel, L. (2018). *Introducción, conceptos, creación de sitios Web, diseño de páginas WEB, herramientas para generar páginas WEB, tipos, principales funciones, aplicaciones*.

Orrala, W., & Castro, R. (2018). “DESARROLLO DE UNA APLICACIÓN WEB RESPONSIVE PARA SISTEMATIZAR LA GESTIÓN DE PLANIFICACIÓN Y SEGUIMIENTO DEL PROCESO DE VISITAS ÁULICAS PARA LA CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES DE LA FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS DE LA UNIVERSIDAD DE GUAYAQUIL” *PROYECTO DE TITULACIÓN*.

Palacios, D., Guaman, J., & Contenido, S. (2018). Análisis del rendimiento de librerías de componentes Java Server Faces en el desarrollo de aplicaciones web. *NOVASINERGIA REVISTA DIGITAL DE CIENCIA, INGENIERÍA Y TECNOLOGÍA*, 1(2), 54–59. <https://doi.org/10.37135/unach.ns.001.02.06>

SANCHEZ, A., & SILVA, C. (2017). *Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django*. 4, 17–35.

Tehnică Moldovei, U. (n.d.). *DJANGO ORM VERSUS DOCTRINE ORM FLOREA VICTORIA*. https://en.wikipedia.org/wiki/Data_mapper_pattern

Ulloa Revelo, H. R. (2020). *IMPLEMENTACIÓN DEL MÓDULO DE PERSONALIZACIÓN DE FRONTALES PARA EL SISTEMA INTEGRADO DE ACTIVIDAD DOCENTE (SIAD) DE LA CARRERA DE SOFTWARE DE LA UNIVERSIDAD TÉCNICA DEL NORTE, APLICANDO TECNOLOGÍAS FACELETS CON BOOTSTRAP Y LA CARACTERÍSTICA DE USABILIDAD DEL ESTÁNDAR ISO/IEC 25010*.

Vidal-Silva, C. L., Sánchez-Ortiz, A., Serrano, J., & Rubio, J. M. (2021a). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Formacion Universitaria*, 14(5), 85–94. <https://doi.org/10.4067/S0718-50062021000500085>

Vidal-Silva, C. L., Sánchez-Ortiz, A., Serrano, J., & Rubio, J. M. (2021b). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Formacion Universitaria*, 14(5), 85–94. <https://doi.org/10.4067/S0718-50062021000500085>

Yaremko, I. (2022). *BOOKSWAP APPLICATION PROJECT REPORT*.



11. Anexos

ANEXO 1

```
<!-- base -->
<nav class="navbar fixed-top shadow navbar-expand-lg bg-light navbar-light py-3 py-lg-0 px-lg-5" style="display: none;">
  <a href="#" class="navbar-brand ml-lg-3 volver">
    <h1 class="m-0 display-4 d-inline"><span class="text-success">Colombian</span>Kif</h1>
  </a>
  <button type="button" class="navbar-toggler" data-toggle="collapse" data-target="#navbarCollapse">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse px-lg-3" id="navbarCollapse">
    <div class="navbar-nav m-auto py-0">
      <a href="#home" class="nav-item nav-link active">Home</a>
      <a href="#about" class="nav-item nav-link">Sobre Nosotros</a>
      <a href="#servicios" class="nav-item nav-link">Servicios</a>
      <a href="#ubicacion" class="nav-item nav-link">Ubicacion</a>
      <a href="#contact" class="nav-item nav-link">Contactanos</a>
      <a href="#blog" class="nav-item nav-link">Blog</a>
    </div>
  </div>
</nav>
<!-- fin base -->
{% include "Home/home.html" %} <!-- incrusta plantilla home -->
{% include "QuienesSomos/mision_vision.html" %} <!-- incrusta plantilla mision_vision -->
{% include "ServiciosProductos/serviciosProductos.html" %} <!-- incrusta plantilla serviciosProductos -->
{% include "Ubicacion/ubicacion.html" %} <!-- incrusta plantilla ubicacion -->
{% include "Contactanos/contactanos.html" %} <!-- incrusta plantilla contactanos -->
{% include "Blog/blog.html" %} <!-- incrusta plantilla blog -->
<!-- base -->
<i class="fa fa-2x fa-angles-down text-white scroll-to-bottom"></i>
<a href="#" class="btn btn-outline-dark px-0 back-to-top volver"><i class="fa fa-angle-double-up"></i></a>

<div class="container-fluid bg-success text-white py-5 px-sm-3 px-md-5">
  <div class="container text-center py-5">
    <div class="d-flex justify-content-center mb-4">
      <a class="btn btn-light btn-social mr-2" href="https://twitter.com/ColombianKif" target="_blank"><i class="fa
      <a class="btn btn-light btn-social mr-2" href="https://www.facebook.com/colombiankif" target="_blank"><i clas
      <a class="btn btn-light btn-social mr-2" href="https://instagram.com/colombiankif?igshid=YmMyMTA2M2Y=" target
      <a class="btn btn-light btn-social mr-2" href="https://www.linkedin.com/company/colombian-weed-group/" target
    </div>
    <p class="m-0">&copy; {% now "Y" %} Colombian Kif SAS - ZESE All rights reserved.</p>
  </div>
  <div class="text-center">
    <a href="/admin" class="text-secondary" target="_blank">|Administrador|</a></div>
</div>
<!-- fin base -->
```

ANEXO 2



```
<!-- inicio plantilla ubicacion -->
{% load static %}
{% block content %}
<div class="container-fluid py-5" id="ubicacion">
  <div class="container">
    <div class="position-relative d-flex align-items-center justify-content-center mb-4">
      <h1 class="display-1 text-uppercase text-center" style="-webkit-text-stroke: 1px solid #dee2e6;">Ubicacion</h1>
      <h1 class="position-absolute text-uppercase text-success text-center">Ubicaciones</h1>
    </div>
    <div class="row align-items-right">
      <div class="col-lg-5 text-center">
        <div class="mt-2">
          <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3956.8185359602035!2d-72.65098828478394!3d7.37422549" width="auto" height="360" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>
        </div>
        <p>Oficina Principal</p>
      </div>
      <div class="col-lg-7">
        <h3>COLOMBIAN KIF SAS - ZESE</h3>
        <h6 class="mb-4">NIT: <span class="text-secondary">901.329.640-6 </span></h6>
        <p>Esta empresa fué constituida como sociedad por acciones simplificada y se dedica a Cultivo de especias y de plantas a
        <h5>Oficina Principal</h5>
        <div class="row mb-3">
          <div class="col-sm-6 py-2"><h6>Departamento: <span class="text-secondary">Norte de Santander</span></h6></div>
          <div class="col-sm-6 py-2"><h6>Municipio: <span class="text-secondary">Pamplona</span></h6></div>
          <div class="col-sm-6 py-2"><h6>Direccion: <span class="text-secondary">Carrera 6 #7-63</span></h6></div>
          <div class="col-sm-6 py-2"><h6>Local: <span class="text-secondary">11</span></h6></div>
        </div>
        <h5>Sede Operativa Invernaderos Bochalema</h5>
        <div class="row">
          <div class="col-sm-6 py-2"><h6>Departamento: <span class="text-secondary">Norte de Santander</span></h6></div>
          <div class="col-sm-6 py-2"><h6>Municipio: <span class="text-secondary">Bochalema</span></h6></div>
          <div class="col-sm-6 py-2"><h6>Predio: <span class="text-secondary">El Espejo Vereda PeñaViva</span></h6></div>
          <div class="col-sm-6 py-2"><h6>C: <span class="text-secondary">7°36'28" N 72°38'22" W 1.36 Km</span></h6></div>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
<!-- fin plantilla ubicacion -->
```

ANEXO 3



```
<!-- inicio plantilla servicio o productos -->
{% load static %}
{% block content %}
<div class="container-fluid py-5" id="servicios">
  <div class="container">
    <div class="position-relative d-flex align-items-center justify-content-center">
      <h1 class="display-1 text-uppercase text-center" style="-webkit-text-stroke: 1px #dee2e6;">servicios</h1>
      <h1 class="position-absolute text-uppercase text-success text-center">Servicios o producto</h1>
    </div>
  </div>

  <div class="row service">
    {% for servicio in servicios%}
    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.1s">
      <div class="service-item">
        <div class="service-img">
          
          <div class="service-overlay">
            <p>
              {{servicio.descripcion}}
            </p>
          </div>
        </div>
        <div class="service-text">
          <h3>{{servicio.nombre|title}}<span class="text-secondary" style="font-size: 15px!important"><br>{{servicio.descripcion}}</h3>
          <a class="btn" href="{{servicio.imagen.url}}" data-lightbox="service">+</a>
        </div>
      </div>
    </div>
    {% endfor %}
  </div>
</div>
{% endblock%}
<!-- fin plantilla servicio o productos -->
```

ANEXO 4



```

<!-- inicio plantilla objetivos -->
{% load static %}
{% block content %}
<h3 class="font-weight-bold m-0 text-center mt-4">OBJETIVOS INSTITUCIONALES</h3>
<div class="row" >
    <div class="col-sm-6 py-2" >
        <ul class="list-group list-group-flush">...
        </ul>
    </div>
    <div class="col-sm-6 py-2">
        <ul class="list-group list-group-flush">...
        </ul>
    </div>
</div>
<h3 class="font-weight-bold m-0 text-center mt-4">INOCUIDAD DEL CULTIVO DEL CANNABIS MEDICINAL</h3>
<div class="row" >
    <div class="col-lg-6 py-2">
        <div class="text-justify">
            <h5 class="font-weight-bold m-0 text-center mt-4">OBJETIVOS</h5>
            <div class="position-relative owl-carousel testimonial-carousel">
                <div>
                    <ul class="list-group list-group-flush" >...
                    </ul>
                </div>
                <div>
                    <ul class="list-group list-group-flush" >...
                    </ul>
                </div>
                <div>
                    <ul class="list-group list-group-flush" >...
                    </ul>
                </div>
            </div>
        </div>
    </div>
    <div class="col-lg-6 py-2" >
        <div class="text-justify m-4">
            <h5 class="font-weight-bold m-0 text-center">POLITICA</h5>
            <h5 class="font-weight-light text-justify">Colombian KIF es una empresa colombiana de can
        </div>
    </div>
</div>
{% endblock %}
<!-- fin plantilla objetivos -->

```

ANEXO 5



```
<!-- inicio plantilla mision_vision -->
{% load static %}
{% block content %}
<div class="container-fluid py-5" id="about" >
  <div class="container">
    <div class="position-relative d-flex align-items-center justify-content-center mb-4">
      <h1 class="display-1 text-uppercase text-center" style="-webkit-text-stroke: 1px #dee2e6;">Nosotros</h1>
      <h1 class="position-absolute text-uppercase text-success text-center">Sobre Nosotros</h1>
    </div>
    <div class="row" >
      <div class="col-sm-6 py-2" >
        <h3 class="font-weight-bold m-0 text-center">MISIÓN</h3>
        <h5 class="font-weight-light text-justify">Somos una empresa colombiana de cannabis medicinal sujeta a la ley</h5>
      </div>
      <div class="col-sm-6 py-2">
        <h3 class="font-weight-bold m-0 text-center">VISIÓN</h3>
        <h5 class="font-weight-light text-justify">Ser una compañía líder a nivel nacional, con influencia internacional</h5>
      </div>
    </div>
    {% include "QuienesSomos/objetivos.html"%} <!-- incrusta plantilla de objetivos -->
  </div>
</div>
{% endblock %}
<!-- fin plantilla mision_vision -->
```

ANEXO 6

```
<!-- inicio plantilla home -->
{% load static %}
{% block content %}
<div class="modal fade id="videoModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-body">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <div class="embed-responsive embed-responsive-16by9">
          <iframe class="embed-responsive-item" src="" id="video" style="border:none;overflow:hidden" scrolling="no" frameborder="0" >
        </div>
      </div>
    </div>
  </div>
</div>
<div class="container-fluid bg d-flex align-items-center mb-5 py-5" id="home" style="min-height: 100vh;">
  <div class="container">
    <div class="row align-items-center">
      <div class="col-lg-5 px-5 pl-lg-0 pb-5 pb-lg-0">
        
      </div>
      <div class="col-lg-7 text-center text-lg-left">
        <h1 class="h1 display-3 text-uppercase mb-2" style="-webkit-text-stroke: 2px #ffffff;">COLOMBIAN KIF SAS - ZESE </h1>
        <h1 class="typed-text-output d-inline font-weight-lighter text-white"></h1>
        <div class="typed-text d-none">Cannabis Medicinal, Cultivamos, Procesamos</div>
        <div class="d-flex align-items-center justify-content-center justify-content-lg-start pt-5">
          <button type="button" class="btn-play" data-toggle="modal">
            <span></span>
          </button>
          <h5 class="font-weight-normal text-white m-0 m-1-4 d-none d-sm-block">Ver Video</h5>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

ANEXO 7



```
<!-- plantilla error 404 -->
<!DOCTYPE html>
<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>404</title>
    {% load static %}
    <link rel="shortcut icon" type="image/png" href="{% static 'colombianKif/img/favicon.ico' %}" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <div class="d-flex align-items-center justify-content-center vh-100">
      <div class="text-center">
        <h1 class="display-1 fw-bold">404</h1>
        <p class="fs-3"> <span class="text-danger">Oops!</span> Esta pagina no funciona.</p>
        <p class="lead">
          La página que buscas no existe.
        </p>
        <a href="/" class="btn btn-primary">ir a Inicio</a>
      </div>
    </div>
  </body>
</html>
```

ANEXO 8

```
<!-- plantilla error 500 -->
<!DOCTYPE html>
<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>505</title>
    {% load static %}
    <link rel="shortcut icon" type="image/png" href="{% static 'colombianKif/img/favicon.ico' %}" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <div class="d-flex align-items-center justify-content-center vh-100">
      <div class="text-center">
        <h1 class="display-1 fw-bold">505</h1>
        <p class="fs-3"> <span class="text-danger">Oops!</span> Page not found.</p>
        <p class="lead">
          500 Internal Server Error
        </p>
      </div>
    </div>
  </body>
</html>
```

ANEXO 9



```
<!-- plantilla contactanos -->
{% load static %}
{% block content %}
<div class="container-fluid py-5" id="contact">
    <div class="container">
        <div class="position-relative d-flex align-items-center justify-content-center mb-4">
            <h1 class="display-1 text-uppercase text-center" style="-webkit-text-stroke: 1px solid #dee2e6;">Cor
            <h1 class="position-absolute text-uppercase text-success text-center">Contactanos</h1>
        </div>
        <div class="row justify-content-center mt-2">
            <div class="col-lg-5 mb-4">
                <h3 class="font-weight-bold m-0 text-center mb-4 mt-4">Redes</h3>
                <div class="mb-4">...
            </div>
            <div class="mb-4">...
            </div>
            <div class="mb-4">...
            </div>
            <div class="mb-4">...
            </div>
            <div class="mb-4">...
            </div>
            <div class="mb-4">...
            </div>
        </div>
        <div class="col-lg-5 text-center">
            <div class="container-form">
                <!-- condicional para verificar envio de correo -->
                {% if 'exito' in request.GET %}
                <script LANGUAGE="javascript">
                    alert("Informacion enviada correctamente");
                    window.location.href = "/";
                </script>
                {% endif %}
                {% if 'error' in request.GET %}
                <script>
                    alert("A ocurrido un problema intenta de nuevo");
                    window.location.href = "/";
                </script>
                {% endif %}
                <h3 class="font-weight-bold m-0 text-center">Envia un correo</h3>
                <!-- formulario con csrf -->
                <form action="" method="POST">{% csrf_token %}
                    {{formulario}}
                <!-- formulario generado por el orm -->
                <button class="btn-send"><span>Enviar</span><i></i></button>
                </form>
            </div>
        </div>
    </div>
</div>
```

ANEXO 10



```

<!-- plantilla blog -->
{% load static %}
{% block content %}
<div class="container-fluid py-5" id="blog">
    <div class="container">
        <div class="position-relative d-flex align-items-center justify-content-center">...
        </div>
        <div class="owl-carousel testimonial-carousel-2 wow fadeInUp" style="background:transparent!important;;"
            data-wow-delay="0.6s">
            <!-- for anidado -->
            {% for post in posts %}
            <div class="testimonial-item my-4">
                <div class="d-flex align-items-center border-bottom pt-5 pb-4 px-5">
                    {% if post.imagen %}
                    
                    {% else %}
                    
                    {% endif %}
                    <div class="ps-4">
                        <h4 class="mb-1">{{post.titulo|upper}}</h4>
                        {% for categoria in post.categoria.all %}
                        <small class="text-uppercase">
                            {% if not forloop.first %}
                            -
                            {% endif %}
                            {{categoria.nombre}}
                        </small>
                        {% endfor %}
                    </div>
                </div>
                <div class="pt-4 px-4 prim col-9">
                    {{post.contenido}}
                    <p class="pb-4" style="font-size: 12px;">
                        Autor: {{post.autor|title}}
                    </p>
                </div>
                <div class="col-2 pt-4">
                    {% if post.link %}
                    <a href="{{post.link}}" class="btn btn-success" target="_blank"
                        rel="noopener noreferrer">Ver</a>
                    {% endif %}
                </div>
            </div>
            </div>
        </div>
    </div>
</div>

```

ANEXO 11



```
def home(request):
    if(request.method == "POST"): #post de boton correo
        formulario = FormularioContacto(data = request.POST) # validar datos del formulario
        if (formulario.is_valid()): # si los datos son validos
            # obtener datos del formulario
            nombre = request.POST.get('nombre')
            correo = request.POST.get('correo')
            asunto = request.POST.get('asunto')
            contenido = request.POST.get('contenido')
            # ejecutar funcion de envio de correo
            enviado = sendCorreo(nombre, correo, asunto, contenido)
            # redireccionar segun resultado
            if enviado:
                return redirect('/?exito')
            else:
                return redirect('/?error')
        formulario = FormularioContacto() #inicialziar el formulario
        posts = blog() #obtener todos los blogs
        servicios = serviciosProductos() # obtener todos los servicios
        # renderizar la pagina y enviar los datos que requiere
        return render(request, 'base.html', {'formulario':formulario, 'posts':posts, 'servicios':servicios})
```

ANEXO 12

```
# funcion para enviar correos con sendgrid
def sendCorreo(nombre, correo, asunto, contenido):
    # dar formato html al contenido
    contenido = """
    <h1>Buenos días, mis datos son:</h1>
    <h2>Nombre:</h2>
    <h3>{}</h3>
    <h2>Correo:</h2>
    <h3>{}</h3>
    <h2>Contenido:</h2>
    <h3>{}</h3><br><br><br>
    <p>No responda a este correo</p>""".format(nombre, correo, contenido)
    # establecer parametros de envio de correo leyendolos de las variables de entorno
    message = Mail(
        from_email=os.environ.get('FROM_EMAIL'),
        to_emails=os.environ.get('TO_EMAIL'),
        subject=asunto,
        html_content=contenido)
    try:
        # enviar el correo con sendgris api
        sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
        sg.isbodyhtml = True
        sg.send(message)
        return True
    except:
        return False
```

ANEXO 13



```
# funcion para obtener todos los blogs usando el orm
def blog():
    posts = post.objects.all()
    return posts

# funcion para obtener todos los servicios usando el orm
def serviciosProductos():
    servicios = servicio.objects.all()
    return servicios
```

ANEXO 14

```
# vista para manejar el error 404
def pag_404_not_found(request, exception):
    return render(request, 'Errores/404.html')
#vista para manejar el error 500
def pag_500_error_server(request):
    return render(request, 'Errores/500.html')
```

ANEXO 15



```
# url de aplicacion
from django.urls import path
from .views import *

urlpatterns = [
    path('', home, name="home"),
]
```

```
# url de aplicacion
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings
from django.conf.urls import handler404, handler500

from ColombianKifApp.views import pag_404_not_found, pag_500_error_server

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('ColombianKifApp.urls')),
]

# configuracion de archivos estaticos
urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
# configuracion de archivos media con clouinary
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
# configuracion de paginas de error
handler404 = pag_404_not_found
handler500 = pag_500_error_server
```

ANEXO 16



```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.

# modelo para categorias de los blogs
class categorias(models.Model):
    nombre = models.CharField(max_length=60)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name = 'categoria'
        verbose_name_plural = 'categorias'

    def __str__(self):
        return self.nombre

# modelo para los blogs
class post(models.Model):
    titulo = models.CharField(max_length=60, help_text='Titulo del post')
    contenido = models.CharField(max_length=100, help_text='Contenido del post')
    autor = models.ForeignKey(User, on_delete=models.CASCADE, help_text='Quien publica este post')
    categoria = models.ManyToManyField(categorias, help_text='Categoria del post')
    imagen = models.ImageField(upload_to = 'blog', null = True, blank = True, help_text='Imagen del post')
    link = models.URLField(max_length=200, null=True, blank=True, help_text='Link de una publicacion')
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name = 'post'
        verbose_name_plural = 'posts'

    def __str__(self):
        return self.titulo
```

ANEXO 17



```
# modelo para los servicios
class servicio(models.Model):
    TIPOS = (
        ("Servicio", "Servicio"),
        ("Producto", "Producto")
    )
    nombre = models.CharField(max_length=60, help_text='Nombre del servicio ó producto')
    descripcion = models.CharField(max_length=100, help_text='Descripcion del servicio ó producto')
    tipo = models.CharField(max_length=10, choices=TIPOS)
    autor = models.ForeignKey(User, on_delete=models.CASCADE, help_text='Quien añade el producto')
    imagen = models.ImageField(upload_to = 'servicios', help_text="foto del servicio ó producto")
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name = 'servicio ó producto'
        verbose_name_plural = 'servicios ó productos'

    def __str__(self):
        return self.nombre
```

ANEXO 18

```
from django.contrib import admin
from .models import categorias, post, servicio
# Register your models here.

#configurar el admin de django para categorias
class categoria_admin(admin.ModelAdmin):
    readonly_fields=('created','updated')
#configurar el admin de django para post
class post_admin(admin.ModelAdmin):
    readonly_fields=('created','updated')
#configurar el admin de django para servicios
class servicio_admin(admin.ModelAdmin):
    readonly_fields = ("created","updated")
# registrar los modelos en el admin de django
admin.site.register(categorias, categoria_admin)
admin.site.register(post, post_admin)
admin.site.register(servicio, servicio_admin)
```

ANEXO 19



```
from django import forms

# formulario de contacto
class FormularioContacto(forms.Form):
    nombre = forms.CharField(label=False, max_length=100, required=True, widget=forms.TextInput(attrs={'placeholder': 'Tu nombre', 'class': 'inputs'}))
    correo = forms.EmailField(label=False, required=True, widget=forms.EmailInput(attrs={'placeholder': 'Tu correo', 'class': 'inputs'}))
    asunto = forms.CharField(label=False, max_length=100, required=True, widget=forms.TextInput(attrs={'placeholder': 'Asunto', 'class': 'inputs'}))
    contenido = forms.CharField(label=False, required=True, widget=forms.Textarea(attrs={'placeholder': 'Contenido', 'class': 'inputs'}))
```

ANEXO 20

```
from pathlib import Path
import os
import dj_database_url

BASE_DIR = Path(__file__).resolve().parent.parent

# secret key leídos desde variable de entorno
SECRET_KEY = os.environ.get('SECRET_KEY', default='your secret key')
# Configurar debug desde variable de entorno
DEBUG = 'DEBUG' not in os.environ

# definir el host de alojamiento de la aplicación leyendo desde variable de entorno
# si no existe la variable de entorno se define como localhost
ALLOWED_HOSTS = []
RENDER_EXTERNAL_HOSTNAME = os.environ.get('RENDER_EXTERNAL_HOSTNAME')
if RENDER_EXTERNAL_HOSTNAME:
    ALLOWED_HOSTS.append(RENDER_EXTERNAL_HOSTNAME)
EXTERNAL_HOSTNAME = os.environ.get('EXTERNAL_HOSTNAME')
if EXTERNAL_HOSTNAME:
    ALLOWED_HOSTS.append(EXTERNAL_HOSTNAME)

# Application definition

INSTALLED_APPS = [
    'admin_interface',
    'colorfield',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cloudinary_storage', #iniciar cloudinary-storage
    'cloudinary', #iniciar cloudinary
    'ColombianKifApp', #iniciar la aplicación
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware', #iniciar whitenoise para archivos estáticos
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

ANEXO 21



```
# conexion a base de datos leyes desde variable de entorno
# si no existe la variable de entorno se usa la base de datos local
DATABASES = {
    'default': dj_database_url.config(
        default='sqlite:///db.sqlite3',
        conn_max_age=600
    )
}
```

ANEXO 22

```
# configurar la ruta de los archivos estaticos para que se almacenen en la carpeta staticfiles
# si no esta en modo debug se usa la ruta de la carpeta static
STATIC_ROOT = BASE_DIR / 'staticfiles'
STATIC_URL = '/static/'
STATICFILES_DIRS = (
    BASE_DIR/ 'static',
)
STATIC_URL = '/static/'
if not DEBUG:
    STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
    STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

X_FRAME_OPTIONS = "SAMEORIGIN"
SILENCED_SYSTEM_CHECKS = ["security.W019"]

# configuracion de clouinary
MEDIA_URL = '/colombianKif/'
DEFAULT_FILE_STORAGE = 'cloudinary_storage.storage.MediaCloudinaryStorage'

CLOUDINARY_STORAGE = {
    'CLOUD_NAME' : os.environ.get('CLOUD_NAME'),
    'API_KEY' : os.environ.get('API_KEY'),
    'API_SECRET' : os.environ.get('API_SECRET')
}
```

ANEXO 23



```
# ejecutar con ./build.sh
set -o errexit

pip install -r requirements.txt

python manage.py collectstatic --no-input
python manage.py makemigrations
python manage.py migrate
```

ANEXO 24

Archivo requirements.txt generado con “pip freeze > requirements.txt”

```
asgiref==3.5.2
Brotli==1.0.9
certifi==2022.9.24
charset-normalizer==2.1.1
cloudinary==1.30.0
dj-database-url==1.0.0
Django==3.2.16
django-admin-interface==0.22.1
django-cloudinary-storage==0.3.0
django-colorfield==0.7.2
django-flat-responsive==2.0
django-flat-theme==1.1.4
dnspython==2.2.1
gunicorn==20.1.0
idna==3.4
Pillow==9.3.0
psycpg2==2.9.5
psycpg2-binary==2.9.5
pymongo==4.3.3
python-http-client==3.3.7
pytz==2022.6
requests==2.28.1
sendgrid==6.9.7
six==1.16.0
sqlparse==0.2.4
starkbank-ecdsa==2.2.0
tzdata==2022.6
urllib3==1.26.12
whitenoise==6.2.0
```