
PROYECTO 3 DE IPC 2, RED SOCIAL

202100119 – Samuel Isaí Muñoz Pereira

Resumen

Python es un lenguaje de programación de alto nivel que es utilizado ampliamente en diferentes ámbitos como el desarrollo web, científico, de automatización, entre otros. Flask y Django son frameworks populares de Python para el desarrollo web.

Flask es un micro-framework que se utiliza para crear aplicaciones web pequeñas y medianas. Proporciona un conjunto mínimo de herramientas para el desarrollo, lo que lo hace fácil de aprender y utilizar. Flask es muy adecuado para proyectos pequeños y medianos.

Por otro lado, Django es un framework completo que se utiliza para construir aplicaciones web grandes y complejas. Proporciona una estructura de proyecto rígida y un conjunto completo de herramientas para el desarrollo. Django es muy adecuado para proyectos grandes y complejos,

Palabras clave

GUI: Interfaz gráfica.

Algoritmo: pasos para realizar determinada acción

Abstract

Python is a high-level programming language that is widely used in different fields such as web development, scientific computing, automation, among others. Flask and Django are popular Python frameworks for web development.

Flask is a micro-framework used to create small and medium-sized web applications. It provides a minimal set of tools for development, making it easy to learn and use. Flask is well-suited for small and medium-sized projects.

On the other hand, Django is a full-stack framework used to build large and complex web applications. It provides a rigid project structure and a complete set of tools for development. Django is well-suited for large and complex projects.

Keywords

GUI: Graphical User Interface.

Algorithm: Steps to perform a certain action.

Introducción

El presente proyecto tiene como objetivo desarrollar habilidades para la creación y utilización de frameworks para la creación de paginas web, con esto el estudiante se introduce al mundo web, conociendo así la creación de APIs, manejo de archivos para la utilización de bases de datos, utilización de librerías para todo tipo de lecturas de archivos, manejo de endpoints y peticiones a través de postman u otro tipo de clientes

Desarrollo del tema

Flask

Es un micro-framework de Python utilizado para crear aplicaciones web pequeñas y medianas. Aunque Flask es uno de los frameworks web más simples y minimalistas de Python, es muy poderoso y es utilizado ampliamente para crear aplicaciones web rápidas y eficientes.

Una de las principales ventajas de Flask es su flexibilidad y simplicidad. A diferencia de otros frameworks web de Python, Flask no tiene una estructura de proyecto rígida y proporciona un conjunto mínimo de herramientas para el desarrollo. Esto lo hace fácil de aprender y utilizar, lo que lo hace muy adecuado para proyectos pequeños y medianos.

Otra ventaja de Flask es su capacidad para integrarse fácilmente con otras herramientas y tecnologías. Flask proporciona una amplia variedad de extensiones y módulos que pueden ser utilizados para añadir características específicas a la aplicación, como autenticación de usuarios, formularios, base de datos, entre otros.

Flask también cuenta con una gran comunidad de desarrolladores y usuarios que contribuyen al desarrollo de paquetes y extensiones. Esto significa que la documentación de Flask es muy completa y existen muchas soluciones disponibles para problemas comunes.

Flask utiliza un patrón de diseño llamado Modelo-Vista-Controlador (MVC) para separar la lógica de la aplicación en componentes separados y reutilizables. Flask proporciona un enrutador integrado y una plantilla de motor para la gestión de solicitudes y respuestas HTTP.

Django

Django es un framework de Python utilizado para el desarrollo web que se enfoca en la eficiencia, la escalabilidad y la seguridad. Django proporciona una estructura de proyecto rígida y un conjunto completo de herramientas para el desarrollo de aplicaciones web complejas y grandes.

Una de las principales ventajas de Django es su capacidad para manejar grandes volúmenes de datos y tráfico. Django proporciona una capa de abstracción para la base de datos que permite el acceso a varias bases de datos relacionales. Además, Django proporciona herramientas de caché que permiten la gestión eficiente de la caché de la aplicación.

Otra ventaja de Django es su seguridad. Django proporciona medidas de seguridad integradas que ayudan a proteger la aplicación contra ataques como inyecciones de SQL y ataques de fuerza bruta. Además, Django proporciona herramientas de autenticación y autorización para la gestión de usuarios y permisos.

Django también es muy adecuado para el desarrollo colaborativo. Django proporciona un sistema de

control de versiones integrado que permite a los desarrolladores trabajar juntos en un proyecto de manera eficiente y colaborativa. Además, Django proporciona un sistema de administración de la aplicación que permite a los desarrolladores gestionar los usuarios, grupos, permisos y datos de la aplicación desde una interfaz de usuario web.

Django utiliza un patrón de diseño llamado Modelo-Vista-Controlador (MVC) para separar la lógica de la aplicación en componentes separados y reutilizables. Django proporciona un enrutador integrado y un potente sistema de plantillas que permiten la gestión eficiente de solicitudes y respuestas HTTP.

POO:

La Programación Orientada a Objetos (POO) es un paradigma de programación que se enfoca en la creación de objetos que interactúan entre sí para realizar tareas. Cada objeto es una instancia de una clase, que define sus atributos (variables) y métodos (funciones). Los objetos interactúan entre sí mediante el envío de mensajes, lo que significa que un objeto llama a un método de otro objeto.

Para este proyecto fue muy necesaria la programación orientada a objetos, ya que se utilizaron muchas clases para guardar cada uno de los elementos en sus respectivas listas.

Normalmente había muchos datos para guardar en cada una de las listas, por ejemplo en la lista de máquinas, existe los datos del nombre de la máquina, numero de pines, numero de elementos, y la lista de elementos, debió a la naturaleza de las listas enlazadas se puede guardar un único dato en un nodo, por lo que la solución fue utilizar múltiples clases con sus respectivos métodos constructores para asignar

cada uno de los valores en sus atributos, para al final poder cargar el objeto clase en la lista enlazada.

También fue muy necesarios para acceder a cada uno de los atributos de los objetos, para luego utilizarlo en cada una de las aplicaciones.

Librerías utilizadas

xml.etree.ElementTree:

es una biblioteca de Python que permite analizar, manipular y crear archivos XML (Lenguaje de Marcado Extensible). Permite leer archivos XML y crear un árbol de elementos que puede ser recorrido para acceder a la información y modificarla. Además, la biblioteca también permite la creación de archivos XML desde cero, lo que es muy útil para generar archivos de configuración, archivos de datos estructurados, etc.

La biblioteca es fácil de usar y tiene una sintaxis simple y clara. Es parte de la biblioteca estándar de Python, lo que significa que está disponible en todas las instalaciones de Python por defecto, sin necesidad de instalar ninguna biblioteca adicional.

Para este proyecto fue necesario la utilización de esta librería ya que el archivo de entrada es de la extensión xml, con esta librería la lectura del archivo es mucho mas simple ya que funciona con un árbol que tiene múltiples ramas. En cada una de las ramas fue necesario utilizar una lista simple, o doble según sea el caso para guardar cada uno de los datos, como elementos, numero atómico, nombre del elemento etc.

También se utilizó esta librería para la creación del reporte de pasos, la cual tiene una extensión xml, muy bien organizada en forma de árbol, con los

respectivos segundos o pasos, su movimiento y el pin que se está ejecutando en ese momento

```
app.route('/Mensaje', methods=['POST'])
def MensajeExaminar():
    mensajeEntrada = request.data

    mensajeEntrante = Mensaje(mensajeEntrada)
    mensajeEntrante._leerMensaje()

    rai2 = r['raiz']('respuesta')
    perfilesNuevos = r.SubElement(rai2, 'usuarios')
    perfilesNuevos.text = f'Se procesaron mensajes para {len(mensajeEntrante.listaUsuarios)} usuarios distintos'

    perfilesExistentes = r.SubElement(rai2, 'mensaje')
    perfilesExistentes.text = f'Se procesaron {mensajeEntrante.contadorMensajes} mensajes en total'

    # Crea la estructura de xml para luego enviarla
    xml_respuesta = r.tostring(rai2, encoding='utf-8', method='xml', xml_declaration=True)
    respuesta = make_response(xml_respuesta)
    respuesta.headers.set('Content-Type', 'application/xml')

    return respuesta
```

Graphviz:

Es una biblioteca de visualización de grafos de código abierto y gratuita, que permite crear diagramas y representaciones gráficas de redes, relaciones y estructuras de datos complejas. La biblioteca se basa en la descripción de los grafos mediante lenguaje de descripción de gráficos DOT

La biblioteca Graphviz proporciona una variedad de herramientas y algoritmos para generar gráficos automatizados y personalizados. Estas herramientas incluyen la capacidad de definir nodos y aristas, etiquetas, formas, colores y estilos. También es posible ajustar la posición de los nodos y aristas, y aplicar diseños automáticos de grafos para obtener un mejor aspecto visual.

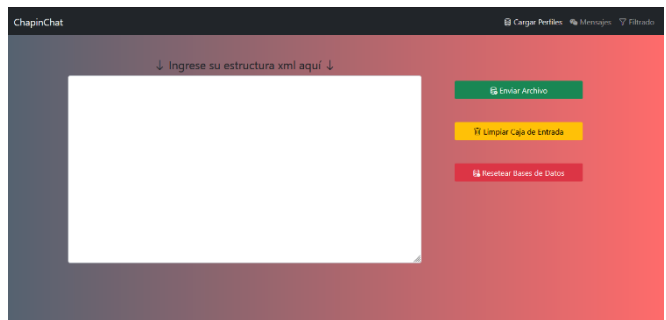
Graphviz es una herramienta popular en áreas como la informática, matemáticas, ciencias sociales, biología, ingeniería y muchas otras. Es compatible con varios formatos de salida, incluyendo PNG, PDF, SVG, entre otros.

Graphviz es una biblioteca ya utilizada en proyectos y practicas anteriores, por lo que es sumamente indispensable en la creación de reportes y de grafos, para este proyecto se utilizó en la creación de las

tablas para la filtración de usuarios, con Graphviz es mucho mas simple crear todo tipo de tablas, nodos, etc. por lo que en este proyecto no fue la excepción

Parte Grafica

Carga Perfiles



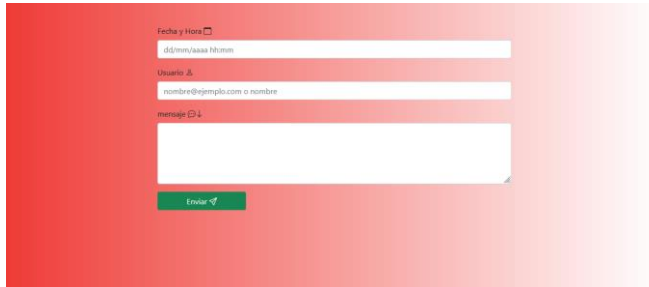
En esta parte el usuario debe de introducir la estructura del XML que desea cargar, la estructura del XML debe de ser parecida a la siguiente

```
<?xml version="1.0" encoding="utf-8"?>
<configuracion>
  <perfiles>
    <perfil>
      <nombre>Deportista</nombre>
      <palabrasClave>
        <palabra>fútbol</palabra>
        <palabra>balonmano</palabra>
        <palabra>baloncesto</palabra>
        <palabra>balompié</palabra>
        <palabra>football</palabra>
        <palabra>gimnasio</palabra>
      </palabrasClave>
    </perfil>
  </perfiles>
```

```
<descartadas>
  <palabra>a</palabra>
  <palabra>un</palabra>
  <palabra>una</palabra>
  <palabra>en</palabra>
  <palabra>para</palabra>
</descartadas>
</configuracion>
```

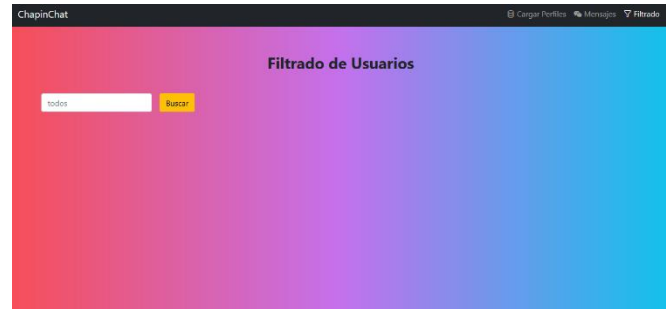
La pagina integra botones que le permiten limpiar la caja de texto o la opción de resetear bases de datos, que borra completamente cualquier dato cargado en los archivos de configuración

Mensajes

El formulario de mensajes está ubicado en la parte superior izquierda de la interfaz. Incluye un campo para la fecha y hora con un icono de calendario, un campo de texto para el usuario con un icono de persona, y un campo de texto para el mensaje con un icono de mensaje. Debajo de estos campos hay un botón verde con el texto 'Enviar' y un icono de flecha hacia la derecha.

En esta pagina el usuario es capaz de ingresar datos como la fecha y hora, el usuario, y el mensaje en cuestión, es importante resaltar que la fecha debe ser en el formato que se indica en el placeholder, y el usuario no debe contener espacios

Una vez enviado el mensaje el servidor analizara la petición y buscara cada una de las coincidencias, en caso de que el archivo sea factible lo guardara con todo el análisis correspondiente de los pesos, en un archivo



El aparato de filtrado es muy simple ya que permite mostrar una tabla diseñada en Graphviz con los datos de los mensajes y los perfiles correspondientes, en caso de que no se conozcan los usuarios cargados en sistema, el usuario puede escribir **todos**, esto automáticamente cargara una tabla con cada uno de los usuarios cargados en sistema y sus respectivos pesos

Datos Importantes:

Puerto Servidor Django: 8000

Puerto Servidor Flask: 5000

Capturas Varias:

Manejo de los mensajes

```
def _leerMensaje(self):
    raiz = ET.fromstring(self.mensaje)

    #Lee el mensaje xml en busqueda de sus elementos
    for listaMensaje in raiz.findall("./mensaje"):
        mensaje = listaMensaje.text
        self.contadorMensajes += 1
        mensaje = mensaje.lower()
        self._analizar(mensaje)

def _analizar(self, mensaje):

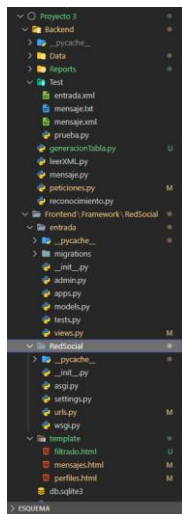
    opcion = 1
    usuario = ""
    textoAnalizar = ""
    ciudad = ""

    while opcion != 5:
        if opcion == 1:
            # Busca el subtítulo lugar y fecha
            match = re.search(r'\'lugar y fecha\'', mensaje)
            inicio = match.start()
            final = match.end()

            #print(f'Inicio: {inicio}, final:{final}')

            index = final+1
            while mensaje[index] != ',':
                ciudad += mensaje[index]
                index +=1
```

Sistema de Ficheros



Conclusiones

Se puede concluir que este proyecto fue una experiencia muy importante en mi carrera como estudiante ya que puse en práctica mis conocimientos básicos sobre la programación orientada a objetos además que pude crear nuevas soluciones en base a cada una de las necesidades del programa.

Para este proyecto fue una experiencia agradable la creación de solicitudes y endpoints ya que nos permite conocer a grandes rasgos como funcionan las estructuras a grandes escalas.

El manejo de frameworks para la creación de paginas web y servidores es cada vez mas necesaria por lo que este proyecto fue una introducción al manejo de este tipo de peticiones.

De mi parte creo que hubiera sido de mas agrado el hecho de poder explorar las herramientas que nos ofrece Django un poco más a fondo

Referencias bibliográficas

Arcila, D. (2020). Introducción a Tkinter: la interfaz gráfica de Python. Platzi. Recuperado el 7 de abril de 2023, de <https://platzi.com/blog/introduccion-a-tkinter-la-interfaz-grafica-de-python/>

García, A., & Pina, J. (2016). Tipos de datos abstractos (TDA): conceptos y aplicaciones. *Revista de Investigación Académica*, 32. Recuperado el 7 de abril de 2023, de <https://www.redalyc.org/articulo.oa?id=323545506005>

Herrera, J. (2017). Tutorial de XML: qué es, cómo funciona y cómo utilizarlo. Programación Fácil.

Recuperado el 7 de abril de 2023, de
<https://programacion-facil.com/es/xml/tutorial-xml-que-es-como-functiona-como-utilizarlo>

Pagella, M. A. (2018). Introducción a Graphviz: la herramienta para visualización de gráficos.
OpenWebinars. Recuperado el 7 de abril de 2023, de
<https://openwebinars.net/blog/introduccion-a-graphviz-la-herramienta-para-visualizacion-de-graficos/>

Mejía, J. C. (2021). Graphviz: una herramienta para visualizar gráficos. GeeksforGeeks. Recuperado el 7 de abril de 2023, de
<https://www.geeksforgeeks.org/graphviz-a-tool-to-visualize-graphs/>