Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

PROYECTO 3: Aplicación Cliente-servidor

201907636 – Carlos Roberto Rangel Castillo

Resumen

Esta investigación proyecto consiste en una aplicación de tipo cliente-servidor utilizando el Protocolo HTTP, donde el cliente sera desarrollado en Django en este se deberá cargar la información de un archivo XML y se podrá visualizar asi como tambien la informacion de salida, por otra parte el servidor sera una api desarrollada en el lenguaje de programación python donde se desarrollaran los métodos ‘GET’ y ‘POST’ Para la comunicación entre entre la api y el cliente.

Palabras clave
HTTP, Django, XML, GET, POST.

Abstract

This research project consists of a client-server type application using the HTTP Protocol, where the client will be developed in Django in which the information from an XML file must be loaded and it will be possible to view as well as the output information, on the other hand the server will be an api developed in the python programming language where the 'GET' and 'POST' methods will be developed for communication between the api and the client.

Keywords
HTTP, Django, XML, GET, POST.

Introducción

Este ensayo hablara acerca de el lenguaje de programación python sus características y los elementos de este mismo, asi tambien acerca del framework web de alto nivel Django y como por medio de este se puede crear el cliente de una aplicación, además se detallaran algunas características del micro framework flask.

Desarrollo del tema

Python: Python es un lenguaje de programación interpretado cuya filosofía hace énfasis en la legibilidad de su código. Hablamos de un lenguaje de programación multiparadigma, debido a que aguanta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación servible. Es un lenguaje interpretado, dinámico y multiplataforma.
Es administrado por la Python Programa Foundation. Tiene una licencia de código abierto, llamada Python Programa Foundation License.

Características y paradigmas: Python es un lenguaje de programación multiparadigma. Esto quiere decir que más que forzar a los programadores a adoptar un estilo especial de programación, posibilita diversos estilos: programación dirigida a objetos, programación imperativa y programación servible. Otros paradigmas permanecen soportados por medio de la utilización de extensiones.

Python usa tipado dinámico y conteo de referencias para la gestión de memoria.
Una característica fundamental de Python es la resolución dinámica de nombres; o sea, lo cual enlaza un procedimiento y un nombre de variable a lo largo de la ejecución del programa (también denominado enlace dinámico de métodos).
Otro objetivo del diseño del lenguaje es la facilidad de expansión. Tienen la posibilidad de redactar nuevos módulos de forma fácil en C o C++. Python puede incluirse en aplicaciones que requieren una interfaz programable.
Aun cuando la programación en Python podría considerarse en varias situaciones hostil a la programación servible clásico del Lisp, hay bastantes analogías entre Python y los idiomas minimalistas de el núcleo familiar Lisp como podría ser Scheme.

Modo interactivo: El intérprete de Python estándar incluye un modo interactivo en el que se escriben las indicaciones en una especie de intérprete de comandos: las expresiones tienen la posibilidad de ser introducidas una a una, logrando verse el resultado de su evaluación velozmente, lo cual da la probabilidad de probar cantidades de código en el modo interactivo previo a integrarlo como parte de un programa. Esto resulta eficaz como para los individuos que se permanecen familiarizando con el lenguaje como para los programadores más avanzados.

Hay otros programas, como por ejemplo IDLE, bpython o IPython,25 que agregan funciones extra al modo interactivo, como la compleción automática de código y el coloreado de la sintaxis del lenguaje.
Ejemplo del modo interactivo:

```
>>> 1 + 1
2
>>> a = range(10)
>>> print(list(a))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

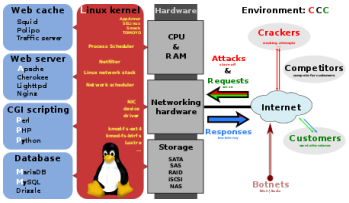


Figura 1. LAMP comprende Python

Elementos del lenguaje: Python ha sido pensado para ser leído con facilidad. Una de sus propiedades es la utilización de palabras donde otros idiomas usarían símbolos. Ejemplificando, los operadores lógicos !, || y &;&; en Python se escriben not, or y and, respectivamente. Curiosamente el lenguaje Pascal es junto con COBOL uno de los idiomas con bastante clara sintaxis y los dos son de la década del 70. La iniciativa del código claro y legible no es algo nuevo.

El contenido de los bloques de código (bucles, funcionalidades, clases, etcétera.) es delimitado por medio de espacios o tabuladores, conocidos como indentación, previo a cada línea de directivas pertenecientes al bloque. Python se diferencia de esta forma de otros idiomas de programación que mantienen

como costumbre publicar los bloques por medio de un grupo de letras y números, comúnmente entre llaves {}. Tienen la posibilidad de usar tanto espacios como tabuladores para sangrar el código, empero se sugiere no mezclarlos.

Variables: Las cambiantes se definen de manera dinámica, lo cual supone que no se tiene que especificar cuál es su tipo de antemano y puede tomar diversos valores en otro instante, inclusive de un tipo distinto al que poseía antes. Se utiliza el signo = para dedicar valores.

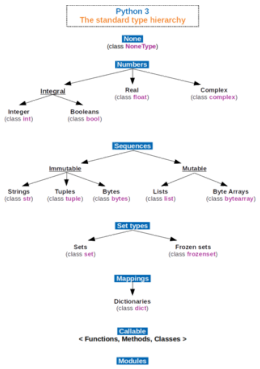


Figura 2. variables Python

Condicionales: Una sentencia condicional (if) hace su bloque de código interno solo si se cumple cierta condición. Se define utilizando el término clave if seguida de la condición, y el bloque de código. Condiciones extras, si las hay, se introducen utilizando elif seguida de la condición y su bloque de código. Cada una de las condiciones se evalúan secuencialmente hasta hallar la primera que sea verdadera, y su bloque de código asociado es el exclusivo que se realiza. Opcionalmente, puede haber un bloque final (la palabra clave else seguida de un bloque de código) que se hace solo una vez que cada una de las condiciones fueron erróneas.

Bucle for: El bucle for es semejante a foreach en otros idiomas. Recorre un objeto iterable, como una lista, una tupla o un generador, y por cada factor del iterable hace el bloque de código interno. Se define con el término clave for seguida de un nombre de variable, seguido de in, seguido del iterable, y al final el bloque de código interno. En cada iteración, el factor siguiente del iterable se asigna al nombre de variable detallado.

Bucle while: El bucle *while* evalúa una condición y, si es verdadera, ejecuta el bloque de código interno. Continúa evaluando y ejecutando mientras la condición sea verdadera. Se define con la palabra clave `while` seguida de la condición, y a continuación el bloque de código interno

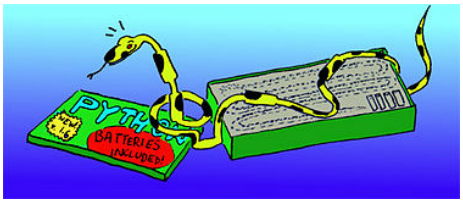


Figura 3. Python viene con "pilas incluidas"

Django: Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago.

Django es un software completo porque sigue la filosofía "Baterías incluidas" y provee casi todo lo que los desarrolladores quisieran que tenga "de fábrica". Porque todo lo que necesitas es parte de un único "producto", todo funciona a la perfección, sigue principios de diseño consistentes y tiene una amplia y actualizada documentación.

Django es un software versátil porque puede ser (y ha sido) usado para construir casi cualquier tipo de sitio web — desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc).
Internamente, mientras ofrece opciones para casi cualquier funcionalidad que desees (distintos motores de base de datos , motores de plantillas, etc.), también puede ser extendido para usar otros componentes si es necesario.

Django es seguro. Django ayuda a los desarrolladores evitar varios errores comunes de seguridad al proveer un framework que ha sido diseñado para "hacer lo correcto" para proteger el sitio web automáticamente. Por ejemplo, Django, proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando así errores comunes como colocar informaciones de sesión en cookies donde es vulnerable (en lugar de eso las cookies solo contienen una clave y los datos se almacenan en la base de datos) o se almacenan directamente las contraseñas en un hash de contraseñas.

Django es escalable. Django usa un componente basado en la arquitectura “shared-nothing” (cada parte de la arquitectura es independiente de las otras, y por lo tanto puede ser reemplazado o cambiado si es necesario). Teniendo en cuenta una clara separación entre las diferentes partes significa que puede escalar para aumentar el tráfico al agregar hardware en cualquier nivel: servidores de cache, servidores de bases de datos o servidores de aplicación. Algunos de los sitios más concurridos han escalado a Django para satisfacer sus demandas (por ejemplo, Instagram y Disqus, por nombrar solo dos).

También es mantenible. El código de Django está escrito usando principios y patrones de diseño para fomentar la creación de código mantenible y reutilizable. En particular, utiliza el principio No te repitas "Don't Repeat Yourself" (DRY) para que no exista una duplicación innecesaria, reduciendo la cantidad de código. Django también promueve la agrupación de la funcionalidad relacionada en "aplicaciones" reutilizables y en un nivel más bajo, agrupa código relacionado en módulos (siguiendo el patrón Model View Controller (MVC)).

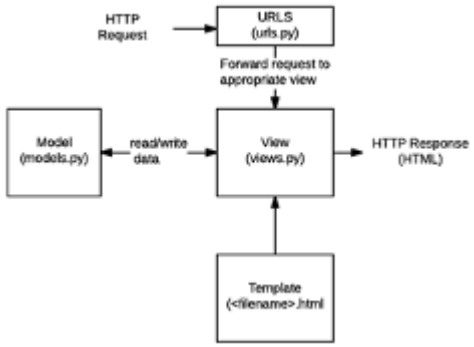
Finalmente Django es portable. Django está escrito en Python, el cual se ejecuta en muchas plataformas. Lo que significa que no está sujeto a ninguna plataforma en particular, y puede ejecutar sus aplicaciones en muchas distribuciones de Linux, Windows y Mac OS X. Además, Django cuenta con el respaldo de muchos proveedores de alojamiento web, y que a menudo proporcionan una infraestructura específica y documentación para el alojamiento de sitios de Django.

¿De donde vinó?:

Django fue desarrollado inicialmente entre 2003 y 2005 por un equipo que era responsable de crear y mantener sitios web de periódicos. Después de crear varios sitios, el equipo empezó a tener en cuenta y reutilizar muchos códigos y patrones de diseño comunes. Este código común se convirtió en un framework web genérico, que fue de código abierto, conocido como proyecto "Django" en julio de 2005.Django ha continuado creciendo y mejorando desde su primer hito, el lanzamiento de la versión (1.0) en septiembre de 2008, hasta el reciente lanzamiento de la versión 1.11 (2017). Cada lanzamiento ha añadido nuevas funcionalidades y solucionado errores, que van desde soporte de nuevos tipos de bases de datos, motores de plantillas, caching, hasta la adición de funciones genéricas y clases de visualización (que reducen la cantidad de código que los desarrolladores tiene que escribir para numerosas tareas de programación).

Django es ahora un próspero proyecto colaborativo de código abierto, con miles de usuarios y contribuidores. Mientras que todavía presenta algunas características que reflejan su origen, Django ha evolucionado en un framework versátil que es capaz de desarrollar cualquier tipo de sitio web. En un sitio web tradicional basado en datos, una aplicación web espera peticiones HTTP del explorador web (o de otro cliente). Cuando se recibe una petición la aplicación elabora lo que se necesita basándose en la URL y posiblemente en la información incluida en los datos POST o GET. Dependiendo de qué se necesita quizás pueda entonces leer o escribir información desde una base de datos o realizar otras tareas requeridas para satisfacer la petición.

Las aplicaciones web de Django normalmente agrupan el código que gestiona cada uno de estos pasos en ficheros separados:



Fuente: Anónimo Disponible vía web: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

URLs: Aunque es posible procesar peticiones de cada URL individual vía una función individual, es mucho más sostenible escribir una función de visualización separada para cada recurso. Se usa un mapeador URL para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición. El mapeador URL se usa para redirigir las peticiones HTTP a la vista apropiada basándose en la URL de la petición. El mapeador URL puede también emparejar patrones de cadenas o dígitos específicos que aparecen en una URL y los pasan a la función de visualización como datos.

Vista (View): Una vista es una función de gestión de peticiones que recibe peticiones HTTP y devuelve respuestas HTTP. Las vistas acceden a los datos que necesitan para satisfacer las peticiones por medio de *modelos*, y delegan el formateo de la respuesta a las *plantillas* (*"templates"*).

Modelos (Models): Los Modelos son objetos de Python que definen la estructura de los datos de una aplicación y proporcionan mecanismos para gestionar (añadir, modificar y borrar) y consultar registros en la base de datos.

Plantillas (Templates): una plantilla (template) es un fichero de texto que define la estructura o diagrama de otro fichero (tal como una página HTML), con marcadores de posición que se utilizan para representar el contenido real. Una *vista* puede crear dinámicamente una página usando una plantilla, rellenándola con datos de un *modelo*. Una plantilla se puede usar para definir la estructura de cualquier tipo de fichero

.

Flask: es un micro framework escrito en python y concebido para facilitar el desarrollo de aplicaciones Web bajo el patrón MVC utilizado para crear aplicaciones web.

¿Qué es un framework?: Podemos definir framework como un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. En general los framework están asociados a lenguajes de programación.
Ventajas de usar un framework:

- Proporciona una estructura del proyecto.
- Facilita la colaboración.
- Es fácil encontrar librerías adaptadas adaptas al framework.

¿Por qué usar Flask?

- Flask es un “micro” framework.
- Incluye un servidor web de desarrollo.
- Tiene un depurador y soporte integrado para pruebas unitarias.
- Es compatible con python 3.
- Es compatible wsgi.
- Buen manejo de rutas.
- Soporte el uso de cookies seguras.
- Se pueden usar sesiones.
- Flask no tiene ORMs.

Figura 4. Flask

Solucion del proyecto:

Se analizó un archivo XML con la librería regex separando cada una de sus partes realizando un conteo de de mensajes y errores, se filtró la información por el uso del método find para con ello devolver lo solicitado

por el cliente final con los resultados son mostrados por medio de gráficas.

```
            </ERROR>
        </ERRORES>
    </ESTADISTICA>\n'''
    escritura.append(nuevo)
myfile = open("estadisticas.xml", "w")
myfile.write('<ESTADISTICAS>' + "\n")
for r in escritura:
    myfile.write(_str(r))
myfile.write("\n"+'</ESTADISTICAS>')

def peticion_fechas(self, fecha):
    correos_grafica = []
    numero_n_veces = []
    global conteo_todo
    print(conteo_todo)
    for p in conteo_todo:
        j = str(p).find(str(fecha))
        if j >= 2:
            correos_grafica.append(p[1])
            numero_n_veces.append(p[2])

    colores = ['#025DE0', '#0BC1B9', '#ACDC0A', '#F6920B', '#F0FF00', '#E01002', '#3C3B3D']
    fechas = correos_grafica
    primas = numero_n_veces
    plt.autoscale()
    plt.bar(range(len(correos_grafica)), primas, edgecolor='black', color=colores)
    plt.xticks(range(len(correos_grafica)), fechas, rotation_='vertical')
    plt.margins(0.2)
    plt.subplots_adjust(bottom=0.45)
    plt.title(str(fecha))
    plt.ylim(min(primas) - 1, max(primas) + 1)

    plt.savefig("C:/Users/Carlos Rangel/Documents/GitHub/IPC2_Proyecto3_201907636/App/Aplicacion/static/Gr
```

Conclusiones

- El protocolo HTTP sirve para realizar las peticiones ‘GET’ y ‘POST’.

- El micro framework flask es utilizado para levantar servidores.

Referencias bibliográficas

- Flash Changelog. Flask (en inglés). Consultado el 21 de diciembre de 2016.
- Junio de 1999. Publicada la última versión de la especificación HTTP/1.1
- Django weblog. Extraído el 2 de Febrero de 2013.

C. J. Date, (1991). *An introduction to Database Systems*. Addison-Wesley Publishing Company, Inc.