



CREACION DE SERVICIOS WEB CON PYTHON

Software

Trabajo de Investigación

Autor:

Edwin Simba, Steeven Toasa, Alexander Pillajo, Katherine Yauli

Tutor/es:

Ing. Darwin Alulema (tutor)

28 de Mayo de 2019

CREACION DE SERVICIOS WEB CON PYTHON

Preguntas de Repaso (Deber N°2)

Autor

Edwin Simba, Steeven Toasa, Alexander Pillajo, Katherine Yauli

Tutor/es

Ing. Darwin Alulema (tutor)



Software



Sangolqui, 28 de Mayo de 2019

Índice general

1	PLANTEAMIENTO DEL PROBLEMA	1
2	OBJETIVOS	3
2.1	General	3
2.2	Específicos	3
3	ESTADO DEL ARTE	5
4	MARCO TEÓRICO	7
5	DIAGRAMAS	9
6	LISTA DE COMPONENTES	11
7	MAPA DE VARIABLES	13
8	EXPLICACIÓN DEL CÓDIGO FUENTE	15
9	DESCRIPCIÓN DE PRERREQUISITOS Y CONFIGURACIÓN	17
10	APORTACIONES	19
11	RECOMENDACIONES	21
12	CRONOGRAMA	23
13	BIBLIOGRAFÍA	25
14	ANEXOS	27
14.1	MANUAL DE USUARIO	29
14.2	HOJAS TÉCNICA	34
	Bibliografía	77

Índice de figuras

5.1	DIAGRAMAS.	9
7.1	MAPA DE VARIABLES.	13
12.1	CRONOGRAMA.	23
14.1	ANEXOS.	27
14.2	ANEXOS.	28
14.3	ANEXOS.	28
14.4	ANEXOS.	29
14.5	MANUAL DE USUARIO.	30
14.6	MANUAL DE USUARIO.	30
14.7	MANUAL DE USUARIO.	31
14.8	MANUAL DE USUARIO.	31
14.9	MANUAL DE USUARIO.	32
14.10	MANUAL DE USUARIO.	32
14.11	MANUAL DE USUARIO.	33
14.12	MANUAL DE USUARIO.	33
14.13	MANUAL DE USUARIO.	33

1 PLANTEAMIENTO DEL PROBLEMA

Actualmente las comunicaciones tienen una trascendencia muy alta en las organizaciones, desde las más pequeñas hasta las predominantes corporaciones. Por lo tanto, el poder acceder a los datos de una forma óptima, organizada y ordenada se vuelve imprescindible para poder mantener un dialogo entre distintas industrias. El desarrollo de servicios web en cualquier área es completamente posible con Python puesto que hay muchas librerías y frameworks. Python permite construir mucho más con menos líneas de código, por lo que se crean prototipos de forma más eficiente. Python es un lenguaje multiplataforma, esto significa que se puede ejecutar sobre servidores Linux, Windows, Sun, etc, tiene un excelente soporte para programación orientada a objetos (POO) y otros paradigmas de programación. Una de las especialidades de Python y que es donde más se esta utilizando actualmente es para el Big Data (Big data - Wikipedia, la enciclopedia libre), ya que Python tiene un excelente tratamiento para grandes cantidades de datos y operaciones complejas con éstos. El framework Django, proporcionado por Python, es una ventaja para todos los desarrolladores, ya que puede usarse para crear aplicaciones web dinámicas y muy seguras. Los usos y aplicaciones de Python se encuentran mucho más allá de los campos mencionados, desde el desarrollo de juegos hasta la visualización de datos, desde la creación de redes hasta el desarrollo de software en general. Las aplicaciones de Python son numerosas. Se necesita determinar los servicios web que se pueden crear mediante la implementación de Python, por lo que nos planteamos las siguientes preguntas. ¿Qué tipo de programas se puede realizar con Python? ¿Se puede hacer una aplicación Android con Python?

2 OBJETIVOS

2.1 General

- Determinar el proceso a seguir para la creación de servicios web mediante el uso de Python.

2.2 Especificos

- Fundamentarse de bases teóricas y científicas que permitan el desarrollo de la investigación.
- Analizar y preparar información para conocer las mejores soluciones en la creación de servicios web con Python.
- Orientarnos a crear y configurar una interfaz de programación de aplicaciones básica que permita a los usuarios acceder a los datos con mayor facilidad.

3 ESTADO DEL ARTE

Antonio Delgado en [1] señala que Google posee una de las redes de servidores más potentes del planeta. Cada vez que los usuarios acceden a un servicio del buscador, como las búsquedas o el correo electrónico, éste pone en marcha toda su maquinaria con el fin de localizar unos datos que están almacenados en miles de servidores repartidos por todo el mundo. De este modo, puede dar servicio de forma inmediata a millones de peticiones por segundo. Ahora Google pone su red a disposición de los desarrolladores de aplicaciones para que éstas ganen en eficacia y rapidez de ejecución. Cuando un servicio pasa de ser usado por unos pocos a ser utilizado por millones de usuarios se produce una sobrecarga en sus estructuras y una falta de capacidad de sus servidores (Delgado, 2008). En [2] los autores indican que, con el fin de ampliar el abanico de oportunidades de la educación virtual, es importante considerar dos aspectos tecnológicos relevantes: el gran potencial de la penetración de la televisión y el auge de servicios de la Web 2.0 en redes sociales y comunidades en Internet, espacios en los cuales los usuarios comparten y generan conocimiento alrededor de una temática. Así, se hace necesario definir la forma adecuada de implementar y desplegar dichos servicios interactivos en entornos de televisión, dadas las características particulares de este escenario. En [3] los autores presentan un marco metodológico para el análisis y la evaluación de las tecnologías y metodologías que soportarían el desarrollo, tanto de la plataforma como de las aplicaciones, en el área de Automatización Industrial. Dicho marco es requerido por la fase final de una metodología orientada por el contexto de realizar ejercicios de prospectiva tecnológica propuesta previamente. El producto de esa fase consiste en un plan de implantación que está constituido por una serie de etapas que involucran, entre otros aspectos, el análisis y la evaluación de las tecnologías y metodologías que dan soporte al desarrollo. El análisis y la evaluación derivan en la selección de una plataforma computacional, la arquitectura de la aplicación, y las tecnologías y metodologías requeridas para alcanzar un desarrollo funcionalmente exitoso y plenamente integrado. El marco metodológico es aplicado, para su validación, en el desarrollo de un sistema de Adquisición de Datos y Control Supervisorio (SCADA).

4 MARCO TEÓRICO

¿Qué es un servicio web?

Programa informático que permite la comunicación y el intercambio de datos entre aplicaciones y sistemas heterogéneos en entornos distribuidos. Los servicios web son por ende un conjunto de funcionalidad expuesta en una intranet o a través de Internet, por y para aplicaciones y computadoras sin la intervención humana.

Ventajas de los servicios web.

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

Inconvenientes de los servicios Web

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (Common Object Request Broker Architecture).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI (Remote Method Invocation), CORBA o DCOM (Distributed Component Object Model). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

¿Qué es Python?

Python es un lenguaje de programación interpretado de tipado dinámico cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas. Dicho de otro modo, Python es:

- Interpretado: Se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
 - Multiparadigma: Soporta programación funcional, programación imperativa y programación orientada a objetos.
 - Tipado dinámico: Las variables se comprueban en tiempo de ejecución. firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.
 - Multiplataforma: disponible para plataformas de Windows, Linux o MAC.
 - Gratuito: No dispone de licencia para programar.
-

5 DIAGRAMAS

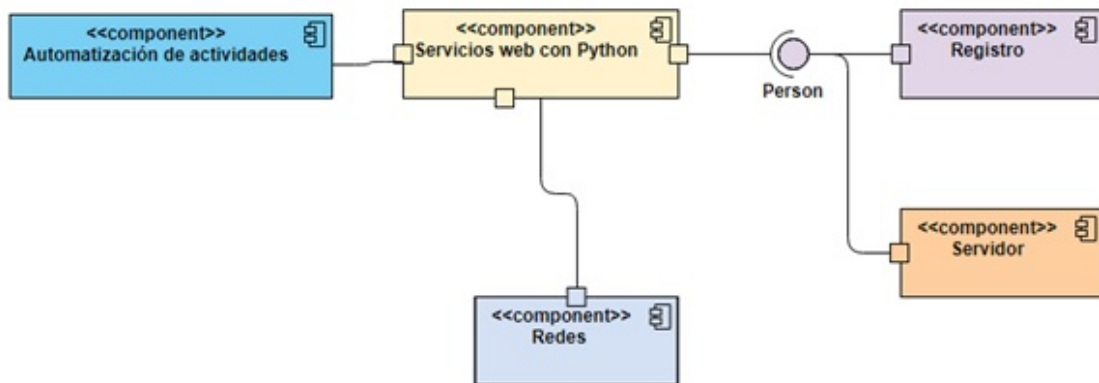


Figura 5.1: DIAGRAMAS.

6 LISTA DE COMPONENTES

En el cliente:

- Una aplicación cliente (proporcionada por el usuario) emite una petición de un servicio web.
- Un proxy de servicio web (también proporcionado por el usuario, pero que se puede generar automáticamente utilizando programas de utilidad del lado de cliente) incluye la solicitud de servicio en un sobre de solicitud SOAP y reenvía la solicitud a un URL definido como punto final del servicio web.
- La red transmite la solicitud al punto final del servicio web mediante HTTP, HTTPS, o JMS para SOAP/JMS.

En el servidor:

- La API de servicios web genérica recibe y decodifica la petición.
- La petición se atiende directamente mediante el componente genérico Business Flow Manager o Human Task Manager, o se remite al proceso BPEL o tarea de usuario especificado.
- Los datos devueltos se encapsulan en un envoltorio de respuesta SOAP.
- La red transmite la respuesta al entorno del lado del cliente mediante HTTP, HTTPS, o JMS para SOAP/JMS.
- Barra de comandos

De nuevo en el extremo cliente:

- La infraestructura de desarrollo del extremo cliente desencapsula el envoltorio de respuesta SOAP.
- El proxy de servicio web extrae los datos de la respuesta SOAP y los pasa a la aplicación cliente.
- La aplicación cliente procesa los datos devueltos como corresponda

7 MAPA DE VARIABLES

Variable	Definición operacional	Dimensión	Indicadores
Servicio web	Un “Web Service” o “Servicio Web”, en términos simples, es como su nombre lo indica, un servicio al que se puede acceder a través de la Web.	Intercambiador de información	<ul style="list-style-type: none">• Eficaz• Confiable• Practico
Python	Se trata de un lenguaje de programación multiparadigma	Enlace dinámico de métodos	<ul style="list-style-type: none">• Programación orientada a objetos,• Programación imperativa y• Programación funcional

Figura 7.1: MAPA DE VARIABLES.

8 EXPLICACIÓN DEL CÓDIGO FUENTE

Un servicio web es un sistema de software diseñado para admitir la interacción interoperable de máquina a máquina a través de una red. El término servicios web describe una forma estandarizada de integrar una aplicación basada en web utilizando XML, SOAP, Wsdl y UDDIopen Standard sobre un protocolo de Internet Back.

Funcionamiento:

- El Service Provider genera el WSDL describiendo el Web Service y registra el WSDL en el directorio UDDI o Service Registry.
- El Service Requestor o la aplicación del cliente requiere un Web Service y se pone en contacto con el UDDI para localizar el Web Service.
- El cliente, basándose en la descripción descrita por el WSDL, envía un request para un servicio particular al Web Service Listener, que se encarga de recibir y enviar los mensajes en formato SOAP.
- El Web Service analiza el mensaje SOAP del request e invoca una operación particular en la aplicación para procesar el request. El resultado se escribe de nuevo en SOAP en forma de respuesta y se envía al cliente. 1607superior)
- El cliente analiza el mensaje de respuesta SOAP y lo interpreta o genera un error si ha habido alguno.

9 DESCRIPCIÓN DE PRERREQUISITOS Y CONFIGURACIÓN

Componentes de los servidores en una aplicación Web Service:

- Web Service. Es el software o componente que realiza las operaciones. Si está escrito en Java, estas operaciones se realizarán en lenguaje Java. Los clientes invocarán estas operaciones enviando mensajes SOAP.
- SOAP Engine. El Web Service no sabe interpretar SOAP requests y crear SOAP responses. Para hacer esto hace falta un SOAP engine, un software que se encarga del manejo de estos mensajes. Apache Axis es un ejemplo.
- Application Server. Para funcionar como un servidor que puede recibir requests desde diferentes clientes, el SOAP engine normalmente funciona dentro de un application server. Este es otro software que proporciona un espacio libre para aplicaciones que han de ser accedidas por múltiples clientes. El SOAP engine funciona como una aplicación dentro del application server. Ejemplos son Apache Tomcat server, Java Servlet y JSP container.
- HTTP Server. Algunos application servers incluyen funcionalidades HTTP, por lo que se pueden tener Web Services funcionando instalando simplemente un SOAP engine y un application server. Sin embargo cuando un application server carece de funcionalidad HTTP es necesario también un HTTP server, más comúnmente llamado Web Server. Es un software que sabe cómo manejar mensajes HTTP. Los dos más populares en la actualidad son Apache HTTP Server y Nginx.

10 APORTACIONES

- Promueven la interoperabilidad: La interacción entre un proveedor y un solicitante de servicio está diseñada para que sea completamente independiente de la plataforma y el lenguaje. Esta interacción requiere un documento WSDL para definir la interfaz y describir el servicio, junto con un protocolo de red (generalmente HTTP).
- Permiten la integración “justo-a-tiempo”: El proceso de descubrimiento se ejecuta dinámicamente, a medida que los solicitantes de servicio utilizan a los agentes para encontrar proveedores de servicio. Una vez el solicitante y el proveedor de servicio se han ubicado, se utiliza el documento WSDL del proveedor para enlazar al solicitante con el servicio. Esto significa que los solicitantes, los proveedores y los agentes actúan en conjunto para crear sistemas que son auto-configurables, adaptativos y robustos.
- Reducen la complejidad por medio del encapsulamiento: Los solicitantes y los proveedores del servicio se preocupan por las interfaces necesarias para interactuar. Como resultado, un solicitante de servicio no sabe cómo fue implementado el servicio por parte del proveedor, y éste a su vez, no sabe cómo utiliza el cliente el servicio. Estos detalles se encapsulan en los solicitantes y proveedores. El encapsulamiento es crucial para reducir la complejidad.
- Dan una “nueva vida” a las aplicaciones de legado: Es relativamente correcto tomar una aplicación, generar un wrapper SOAP, luego generar un documento WSDL para moldear la aplicación como un servicio web.
- Abren la puerta a nuevas oportunidades de negocio: Los servicios web facilitan la interacción con socios de negocios, al poder compartir servicios internos con un alto grado de integración.
- Disminuyen el tiempo de desarrollo de las aplicaciones: Pues gracias a la filosofía de orientación a objetos utilizada, el desarrollo se convierte más bien en una labor de composición.

11 RECOMENDACIONES

En torno a la utilización de Python es importante tener conocimiento de su interfaz y mas que ello, los comandos que se llegan a emplear en el programa, ya que su uso a primera impresión puede ocasionar cierta confusión .

Otro punto importante es que se debe tener en cuenta que cada linea de programación debe estar estructurada para el entendimiento general de lo que se esta realizando.

12 CRONOGRAMA

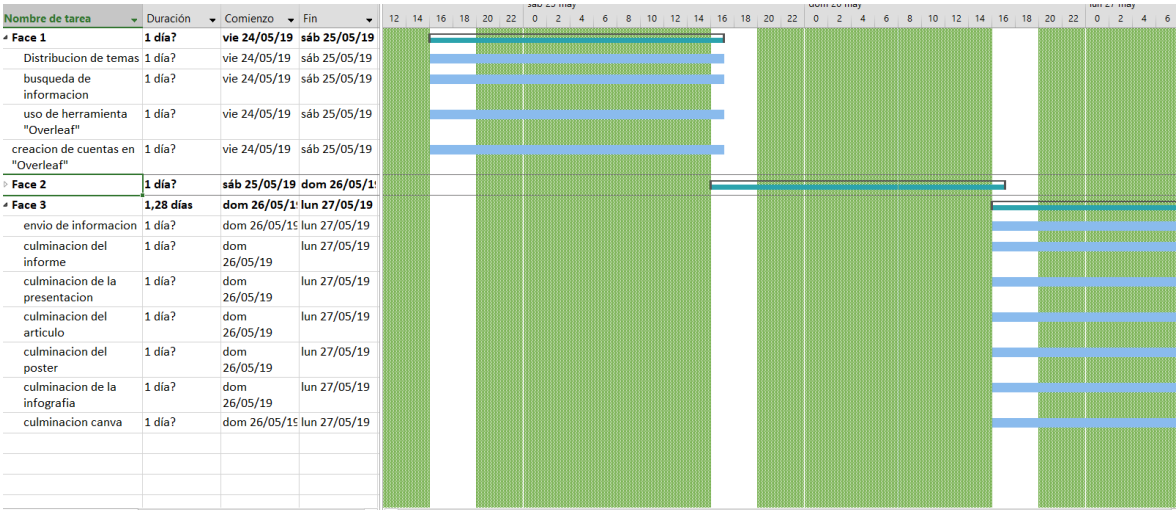


Figura 12.1: CRONOGRAMA.

14 ANEXOS

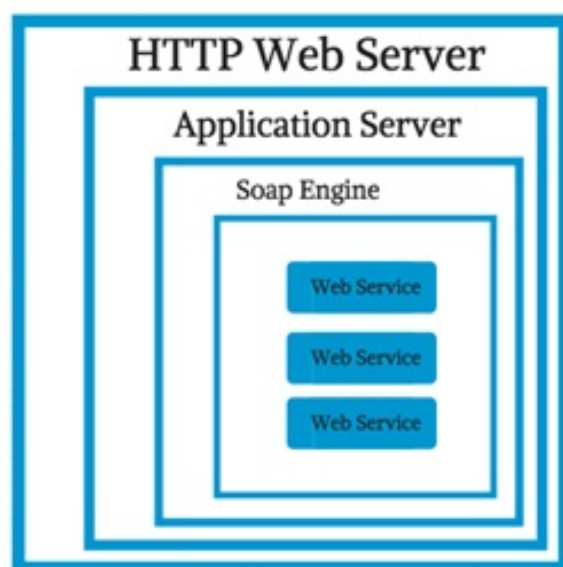
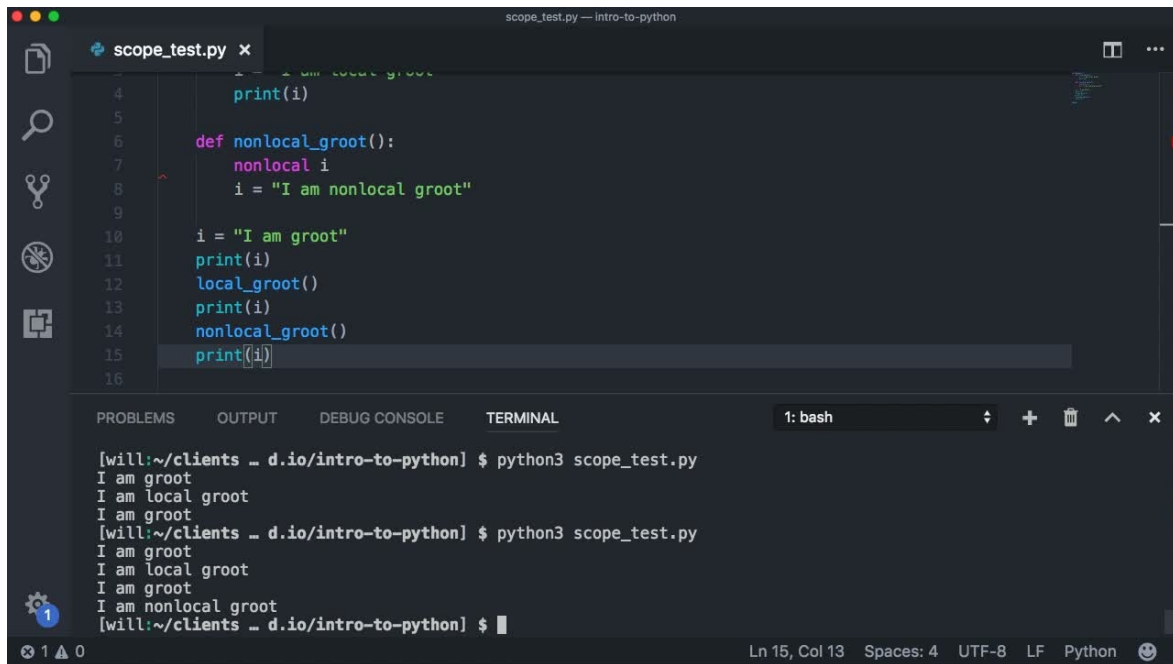


Figura 14.1: ANEXOS.



The screenshot shows a code editor with a file named `scope_test.py`. The code defines a function `nonlocal_groot()` that uses `nonlocal i` to modify a variable in the enclosing scope. The script prints the value of `i` at several points, including before and after calling `nonlocal_groot()`. The terminal output shows the execution results, demonstrating that the nonlocal variable is correctly updated.

```
scope_test.py x
4     print(i)
5
6     def nonlocal_groot():
7         nonlocal i
8         i = "I am nonlocal groot"
9
10    i = "I am groot"
11    print(i)
12    local_groot()
13    print(i)
14    nonlocal_groot()
15    print(i)
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
[will:~/clients - d.io/intro-to-python] $ python3 scope_test.py
I am groot
I am local groot
I am groot
[will:~/clients - d.io/intro-to-python] $ python3 scope_test.py
I am groot
I am local groot
I am groot
I am nonlocal groot
[will:~/clients - d.io/intro-to-python] $
```

Ln 15, Col 13 Spaces: 4 UTF-8 LF Python

Figura 14.2: ANEXOS.

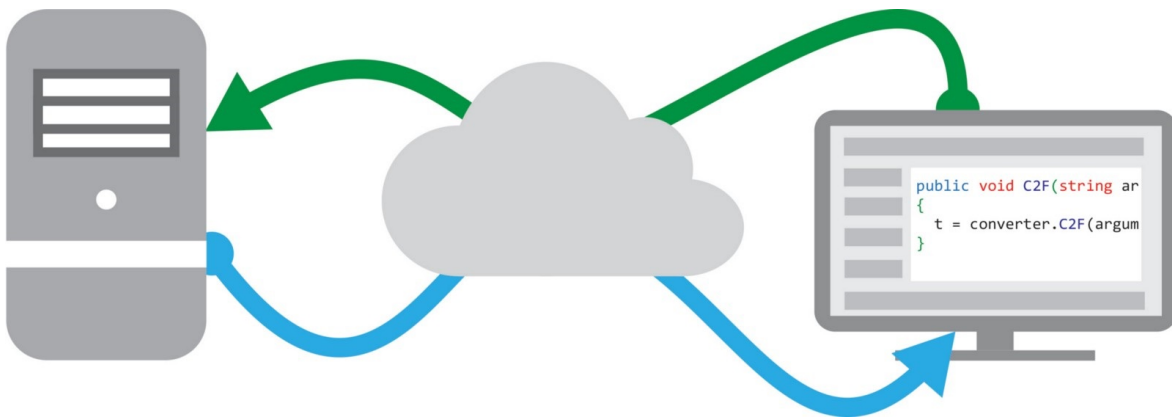


Figura 14.3: ANEXOS.

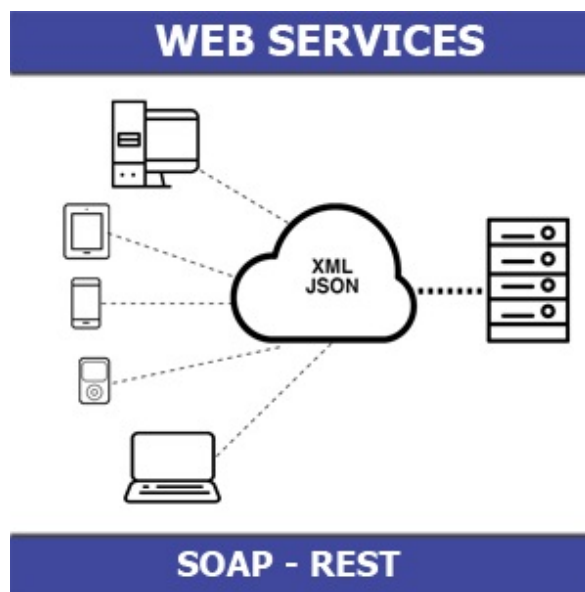


Figura 14.4: ANEXOS.

14.1 MANUAL DE USUARIO

Instalación Python en Windows

Por suerte (o por desgracia) todo el mundo dispone de una copia de Windows en casa, incluso yo; así que en esta instalación si os puedo guiar. Python no viene preinstalado en Windows, por lo que obligatoriamente deberás descargar este paquete

La instalación en Windows no tiene mayor complicación: basta con hacer clic en “Siguiente” repetidas veces hasta finalizar. Y con esto ya tenemos instalado Python en nuestro ordenador.

Python es un lenguaje de programación sencillo y práctico, que permite trabajar con mayor rapidez e integrar sistemas con mayor eficacia. Se puede aprender a usar Python y obtener beneficios casi inmediatos en la productividad y reducir los costes de mantenimiento. Además es gratis, por lo que se convierte en una de las alternativas a MATLAB más fuertes a la hora de remplazar a otros lenguajes de programación.

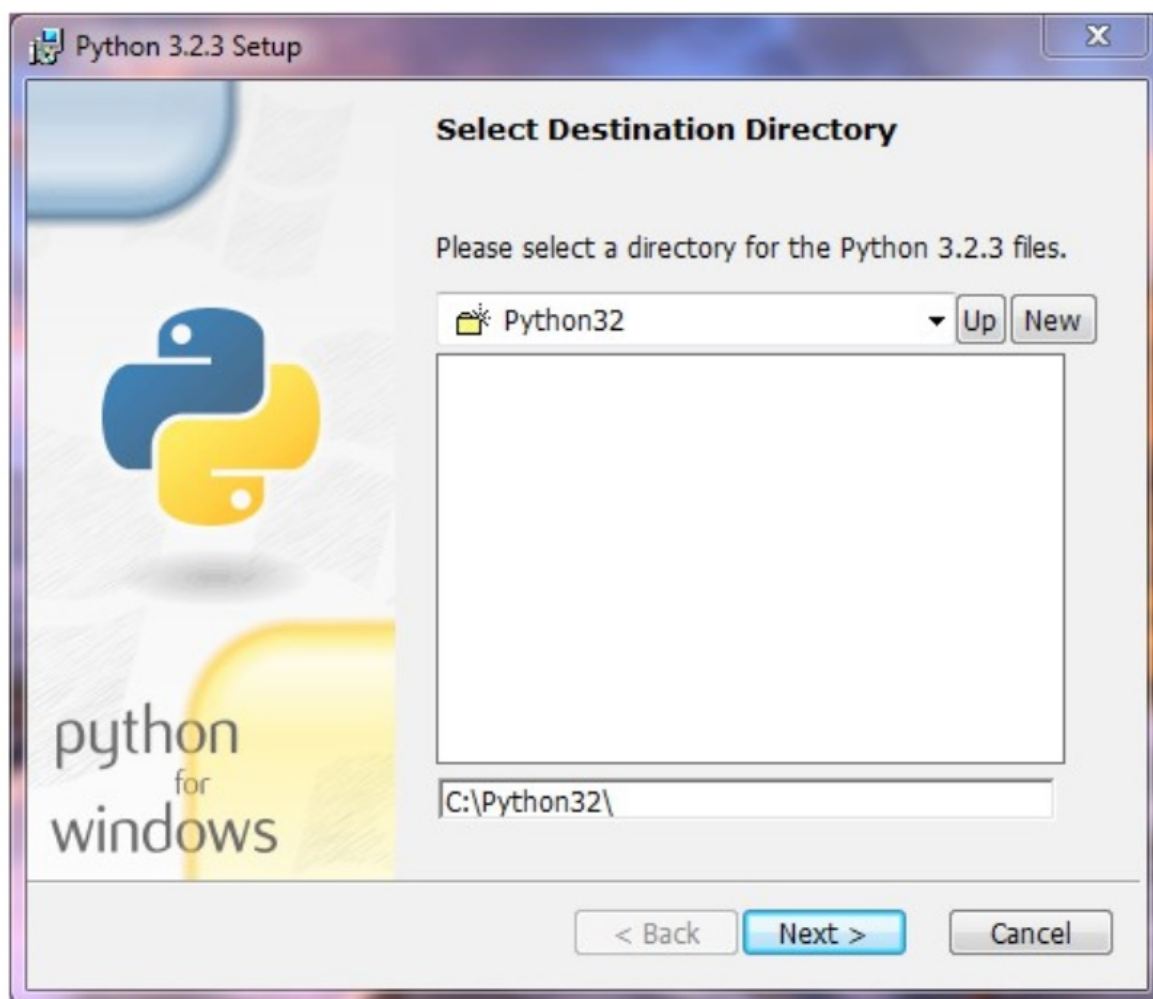


Figura 14.5: MANUAL DE USUARIO.

Crear servicios con Python

Primero vamos a instalar la biblioteca web.py con pip:

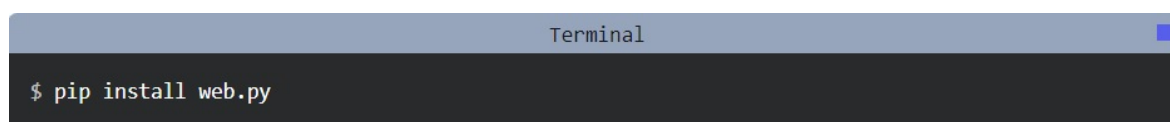


Figura 14.6: MANUAL DE USUARIO.

El ejemplo tendrá un listado de países con un código numérico, nombre y código ISO. El código se podría mejorar utilizando una base de datos en lugar de un diccionario, incluyendo el método PATCH, o creando una clase padre para unificar un poco el

código, pero en esta ocasión quiero mostrar el funcionamiento básico del servicio REST:

```
1 # coding=utf-8
2
3 import web
4 import json
5
6 urls = (
7     '/países(/.*)?', 'países',
8 )
9
10 application = web.application(urls, globals()).wsgifunc()
11
12 class países:
13     países = { 34: { 'España', "ES"}, 33: {"Francia", "FR"} }
14     codes = { 400 : '400 Bad Request',
15              404 : '404 Not Found',
16              405 : '405 Method Not Allowed',
```

Figura 14.7: MANUAL DE USUARIO.

```
17         409 : '409 Conflict'
18     }
19     def __init__(self):
20         web.header('Content-Type', 'application/json', unique=True)
21
22     def GET(self, pais=None):
23         try:
24             columns = [ 'codigo', 'nombre', 'iso' ]
25             if pais is None:
26                 output = []
27                 for i,v in self.países.items():
28                     output.append(dict(zip(columns, [i] + list(v))))
29             else:
30                 pais = int(pais[1:])
31                 if self.países[pais] is None:
32                     raise Exception('Pais no encontrado', 404)
```

Figura 14.8: MANUAL DE USUARIO.

```
        else:
            output = []
            output.append(dict(zip(columns, [pais] + list(self.países[pais]))))

        return json.dumps(output, ensure_ascii=False, encoding='utf8')
    except Exception, e:
        msg, code = e.args if len(e.args)==2 else (e.args, 404)
        raise web.HTTPError(self.codes[code], data="Error: " + str(msg) + "\n")

def POST(self, pais=None):
    try:
        if pais is not None:
            raise Exception('No permitido', 404)

        input = web.input(code=None, nombre=None, iso=None)
        if not input['code'] or not input['nombre'] or not input['iso']:
            raise Exception("Faltan datos de entrada", 400)
```

Figura 14.9: MANUAL DE USUARIO.

```
        raise Exception("Faltan datos de entrada", 400)

    pais = int(input['code'])
    if pais in self.países:
        raise Exception("Elemento existente", 409)

    self.países[pais] = { input['nombre'], input['iso'] }
    web.created()
    web.header('Location', '/países/'+str(pais))
    return ''

except Exception, e:
    msg, code = e.args if len(e.args)==2 else (e.args, 404)
    raise web.HTTPError(self.codes[code], data="Error: " + str(msg) + "\n")

def PUT(self, pais=None):
    try:
```

Figura 14.10: MANUAL DE USUARIO.


```
if pais is None or len(pais)==1:
    raise Exception('País no indicado', 405)

pais = int(pais[1:])
input = web.input(nombre=None, iso=None)
if not input['nombre'] or not input['iso']:
    raise Exception("Faltan datos de entrada", 400)

if pais not in self.países:
    raise Exception("Elemento no encontrado", 404)

self.países[pais] = { input['nombre'], input['iso'] }
return ''

except Exception, e:
    msg, code = e.args if len(e.args)==2 else (e.args, 404)
```

Figura 14.11: MANUAL DE USUARIO.

```
raise web.HTTPError(self.codes[code], data="Error: " + str(msg) + "\n")

def DELETE(self, pais=None):
    try:
        if pais is None or len(pais)==1:
            raise Exception('País no indicado', 405)

        pais = int(pais[1:])
        if pais not in self.países:
            raise Exception("Elemento no encontrado", 404)

        del self.países[pais]
        return ''

    except Exception, e:
        msg, code = e.args if len(e.args)==2 else (e.args, 404)
```

Figura 14.12: MANUAL DE USUARIO.

Para probar el código basta con ejecutar la aplicación:

```
Terminal
$ python webservice.py
http://0.0.0.0:8080/
```

Figura 14.13: MANUAL DE USUARIO.

14.2 HOJAS TÉCNICA

Bibliografía

- [1] Delgado. eroskiconsumer. 29 de Abril de 2008, abril 2012. URL <http://www.consumer.es/web/es/tecnologia/internet/2008/04/29/176214.php>.
- [2] Gabriel chanchi. J. a. scielo, 2015. URL <http://www.scielo.org.co/pdf/inge/v21n1/v21n1a05.pdf>.
- [3] Oswaldo Teran. Facultad de ingenieria. 29 de Marzo de 2009, marzo 2012. URL http://www.scielo.org.ve/scielo.php?script=sci_arttext&pid=S0798-40652009000100005.
- [4] Khosgoftaar. A case study in telecommunications, 2015. URL <http://www.scielo.org.co/pdf/inge/v21n1/v21n1a05.pdf>.
- [5] Object Management Group Inc. (omg)meta object facility (mof), 2015. URL <http://www.scielo.org.co/pdf/inge/v21n1/v21n1a05.pdf>.