TECNOLOGÍAS Y SERVICIOS DE RED

P

P4 Programación de una aplicación con servicios web

- Programación de aplicaciones vía web
- Interacción con servicios
- Almacenamiento y representación de datos
- Python + Servidor de aplicación Flask

INTRODUCCIÓN

A lo largo de esta práctica se pretende que el alumno realice el desarrollo de una aplicación con servicios web. Por sencillez, la aplicación no va a seguir la arquitectura de Web Services XML-SOAP vista en teoría. En su lugar, la arquitectura será parecida a REST, modelo cliente-servidor web. En el servidor residirá la aplicación a programar que seguirá una arquitectura de N-niveles, siendo Flask el servidor de aplicación, implementando el servidor web y la aplicación, tal y como se muestra en la siguiente figura.

Más concretamente, la aplicación a desarrollar será el módulo nombrado como "*lógica de aplicación*", mientras que el módulo "*gestión de recursos*" serán los datos que procesará la aplicación. Estos serán almacenados en una base de datos (BBDD) local y otra en Internet. El acceso a la base de datos de Internet se hará a través de otros servicios web.

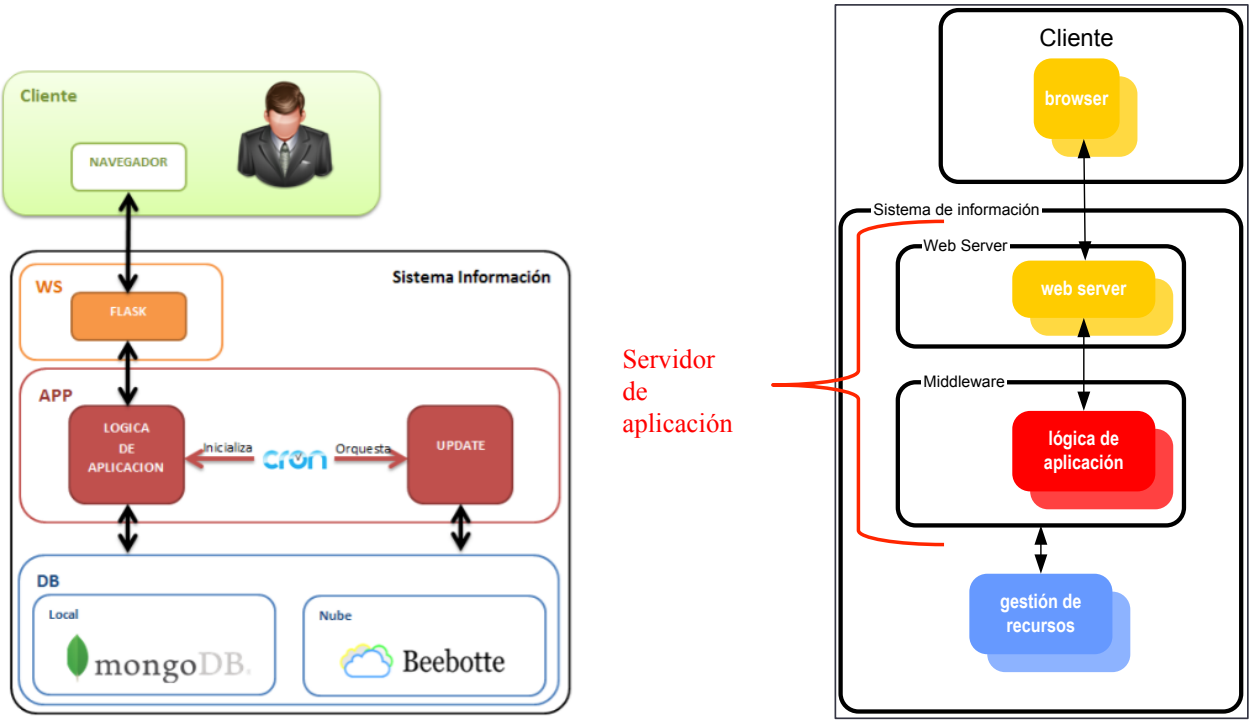

FIGURA 1: ARQUITECTURA DE SISTEMA DE INFORMACIÓN DE N-NIVELES

El objetivo principal de esta práctica es que el alumno muestre su capacidad de aprendizaje de nuevas tecnologías relacionadas con el escenario definido, de forma autónoma, con la guía del profesor.

Herramientas

Para realización de la aplicación, en la que se necesita un procesamiento de datos, se utilizará el lenguaje **Python** [1]. Python es un lenguaje fácil de aprender (para aquellos que no hayan trabajado antes con él), especialmente si partimos de la premisa de que todos los alumnos saben programar al menos en C. Para que los scripts que escribiremos con Python tengan visibilidad en la Web, se deberá hacer uso de un sencillo framework como es **Flask** [2]. Este framework permite a la aplicación el envío de información al cliente usando la capa de presentación estándar Web, así como la recogida de información del usuario final a través de formularios Web.

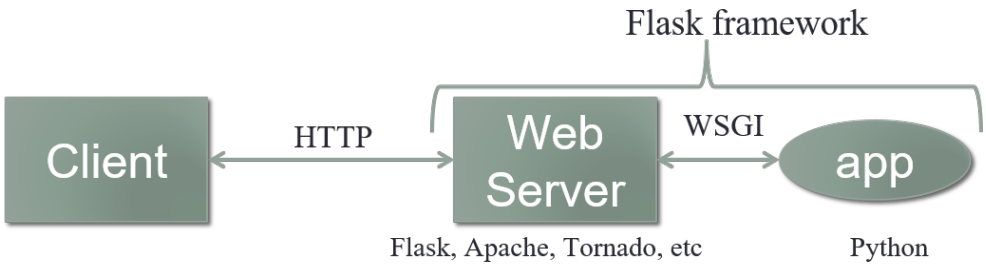

FIGURA 2: ARQUITECTURA IMPLEMENTADA CON FLASK Y PYTHON

Para escribir el código de la aplicación, utilizaremos un editor de código, que podría ser **Sublime Text** [3] o bien **Brackets** [4], o incluso **Notepad++** [5], o cualquier editor y subirlo con Fireftp para el Firefox. Una breve comparativa los describe en [6].

Para mostrar que la aplicación puede ejecutarse en otra máquina diferente a la del cliente, se ejecutará en una máquina virtual Linux que se deberá bajar desde el Aula Virtual y que puede ejecutarse con **VMware Player** [7] en cualquier ordenador.

Finalmente, para el procesado de datos, consideraremos que generalmente toda aplicación obtiene, almacena y muestra datos según una serie de criterios. Estos datos pueden tener una carga de proceso mayor o menor en función del tipo de datos que sea. Para esta práctica:

- Obtención:
  La obtención de datos se realizará de una fuente externa concreta y utilizando un recurso como son las **expresiones regulares (RegExp)** [8], que utiliza patrones de texto para filtrar datos concretos que queramos obtener de un conjunto más amplio.
  Para probar cómo utilizarlas, pueden probarse usando el editor Sublime Text, o de manera online mediante Regex101 [9] ó Pythex [10].

- Procesamiento:
  La parte de proceso de datos va a ser muy ligera, los datos se obtendrán, se almacenarán y se mostrarán en función de unos criterios.

- Almacenamiento:
  El almacenamiento de datos se realizará en dos bases de datos, para tener más fiabilidad y para que el alumno aprenda a trabajar con varios tipos de BBDD.
  - **La primera BBDD será local**.
    Habrá que instalar y trabajar con un Sistema Gestor de Bases de Datos (SGDB). Se valorará positivamente que sea de tipo **NoSQL** [11].
  - **La segunda estará localizada en Internet**, concretamente se utilizará alguno que se ofrezca de manera gratuita. Algunos ejemplos de estos servicios son: **Xiveli** [12], **ThingSpeak** [13], **GrobeStream** [14], **Beebotte** [15], etc.
    Este tipo de plataformas suelen permitir almacenamiento de datos y también implementa interfaces de usuario para representación de los mismos, por ejemplo mediante gráficas. Una aplicación típica que hace uso de esta arquitectura puede ser el control de la temperatura, luces o consumo energético de una casa, en la que una serie de sensores (IoT) se encargan de tomar los datos. En nuestro caso, los datos los obtendremos de manera sencilla mediante una fuente concreta, definida en detalle a continuación.

ESPECIFICACIÓN DE LA APLICACIÓN

Cuando un cliente se conecta a la aplicación a través de su URI asociada, la aplicación realizará los siguientes pasos:

1. **Obtención y filtrado de datos:** Concretamente, los datos se obtendrán de http://www.numeroalazar.com.ar/ , que genera números aleatorios entre 0 y 100 por defecto, tal y como se muestra en la captura de debajo. La aplicación a desarrollar extraerá de la web el primer número generado mediante expresiones regulares, y lo almacenará junto a la fecha y hora de obtención (tomados del servidor de nuestra aplicación). Esta acción se realizará cada 2 minutos.

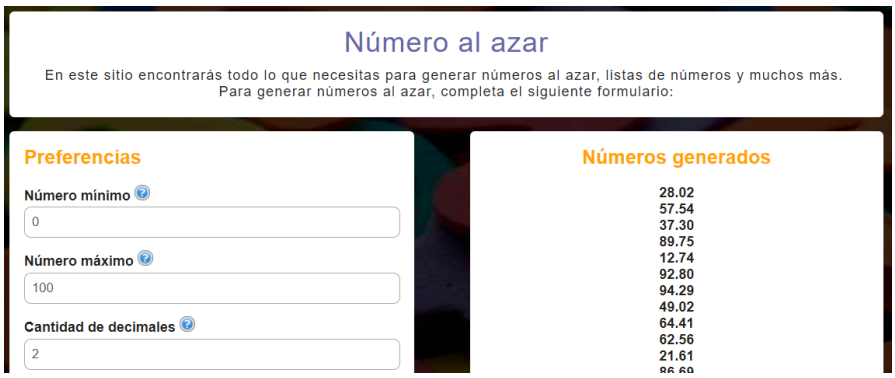

FIGURE 3: CAPTURA DE PANTALLA QUE MUESTRA DATOS OBTENIDOS DE NUMEROALAZAR.COM.AR

2. **Almacenamiento:** Guardará estos valores en la base de datos local elegida por el alumno y, por seguridad, también en uno de los servicios externos indicados anteriormente. Dado que las plataformas localizadas en Internet están centradas en datos recogidos por sensores (temperatura, humedad, etc.), el alumno deberá guardar la información en variables distintas, utilizando una de las opciones presentadas anteriormente como servicios adicionales, para que sean mostradas posteriormente en un modo gráfico.
3. **Presentación de la web:** Mandará la página web al cliente presentando los valores anteriores. Esta página también tendrá un formulario y varios botones, invitando al usuario a introducir un **umbral** como parámetro. Este parámetro será considerado por la aplicación, que deberá indicar cuándo (fecha y hora) fue la última vez que dicho umbral fue superado (revisando su historial de datos obtenidos).
4. **Umbral histórico - Petición del usuario (dato 1):** La aplicación esperará a que el usuario introduzca el umbral en el formulario y le dé al botón de envío. Entonces el cliente web mandará con un POST el formulario a la aplicación.
5. **Umbral histórico - Respuesta de la aplicación al usuario (dato 1):** La aplicación extraerá el valor del umbral y, si se ha superado en alguna de los valores máximos o mínimos almacenados, informará al cliente con un mensaje dando dichos valores con sus respectivas fechas. Este proceso se puede repetir cada vez que se pulse el botón que envía del umbral.
6. **Valor medio - Petición del usuario y respuesta de la aplicación (dato 2):** La página tendrá otro botón para solicitar a la aplicación el valor medio de los valores guardados, que será calculado alternativamente, leyendo los valores almacenados hasta el momento, en la base de datos externa e interna. Además de la media se indicará qué base de datos se ha usado para los cálculos.
7. **Interfaz gráfica de usuario de las plataformas externas (dato 3):** La página web del cliente también tendrá otro botón llamado "gráficas externas" que hará que la aplicación acceda a los servicios de gráficos que tienen las bases de datos externas y mandará dichos gráficos al cliente. Tras mostrar las gráficas, la acción de "volver a la página anterior" del navegador permitirá visualizar la página web del cliente de nuevo.

Especificación avanzada (voluntaria)

Esta práctica tiene una parte avanzada para la obtención de puntuación adicional. Para la implementación de esta parte práctica, se proponen varias ideas:

- Desarrollo de un **umbral actual (dato 4)**, que el usuario indicará y al que la aplicación responderá cuando alguno de los datos que se han capturado, desde que se mandó el

umbral, lo supere. La diferencia con el umbral anterior, histórico, es que la aplicación no revisará la base de datos, sino que continuará leyendo datos hasta que se cumpla la condición y mostrará (mediante una notificación SSE o *web pusher*[1]) una página web al cliente informando qué valor ha superado el umbral y en qué momento.

- Que la aplicación sea multiusuario, de manera que cada usuario pueda fijar el umbral de observación, además los usuarios serán autenticados mediante un sistema OAuth.
- Implementación del servidor de aplicaciones en uno o varios contenedores o *containers* (por ejemplo, basados en Docker [16]), en vez de en una máquina virtual.
- Que el alumno gestione el código de la aplicación realizando control de versiones del código. Para ello se puede hacer uso de servicios de Internet como GitHub [17].
- Otras ideas que el alumno puede proponer.

PRESENTACIÓN Y EVALUACIÓN DE LA PRÁCTICA

La práctica se dará por presentada al profesor encargado del grupo, mediante una **demostración en el laboratorio**[2] del resultado al que se ha llegado, en el día que se fije para ello. Además, antes del viernes 12 de enero, el alumno deberá subir una **memoria** de la práctica realizada (en formato *.pdf) y el **código** (comprimido en formato *.zip), a la plataforma de Aula Virtual de la asignatura.

La memoria deberá incluir:

- Una explicación detallada de todas las fases y pasos realizados (toma de decisiones, desarrollo, resultados, etc.).
- Una descripción de las distintas configuraciones que han sido necesarias para poder llevar a buen término tanto la aplicación realizada, así como los servicios utilizados.
- Capturas de la página web del usuario, así como de la navegación por la misma.

REFERENCIAS

[1] Python – Página oficial: https://www.python.org/
(tutorial en español: http://docs.python.org.ar/tutorial/)
[2] Flask – Web development, one drop at a time: http://flask.pocoo.org/
[3] Sublime Text: https://www.sublimetext.com/

[1] Para implementarlo se puede usar los servicios de alguna de estas páginas https://sendpulse.com/, https://www.izooto.com/ o https://goroost.com/, ver también w3c: https://www.w3.org/TR/push-api/

[2] La demostración podrá realizarse en el propio ordenador del alumno.

[4] Brackets: http://brackets.io/
[5] Notepad++: https://notepad-plus-plus.org
[6] SublimeText vs Brackets vs Notepad++: https://pruebas012015.wordpress.com/2015/06/02/sublime-text-vs-brackets-vs-notepad-son-grandiosos/
[7] VMware Workstation Player: https://www.vmware.com/products/workstation-player.html
[8] Guía de expresiones regulares en Python: https://platzi.com/blog/expresiones-regulares-python/
[9] Regex101 – Comprobador de expresiones regulares en Python online: https://regex101.com/
[10] Pythex – Comprobador de expresiones regulares en Python online: https://pythex.org/
[11] NoSQL: https://aws.amazon.com/es/nosql/
[12] Xively by LogMeIn: https://www.xively.com/
[13] ThingSpeak – Understand Your Things: https://thingspeak.com/
[14] GroveStreams – The platform to build your Internet of Things: https://grovestreams.com/
[15] Beebotte – Cloud Platform for Real Time Connected Objects: https://beebotte.com/
[16] Docker – Build, Ship, and Run any app, anywhere: https://www.docker.com/
[17] GitHub – The world's leading software development platform: https://github.com/

BIBLIOGRAFÍA RECOMENDADA

Además de las referencias que se han presentado previsamente, a continuación se listan una serie de enlaces web a manuales que pueden servir para hacer ciertas partes de la práctica.

- Python. Tutorial Oficial: http://docs.python.org.ar/tutorial/
- Python. Tutorial 2 de: http://www.tutorialspoint.com/python/index.htm
- Python (Coursera) – "*Programming for Everybody (Getting Started with Python)*": https://es.coursera.org/learn/python
- Python. Libro de Eugenia Bahit "*Curso: Python para principiantes*" (pdf): http://www.cursosdeprogramacionadistancia.com/static/pdf/material-sin-personalizar-python.pdf
- Python. Libro de Raúl González Duque "Python para todos" (pdf): http://www.ceibal.edu.uy/contenidos/areas_conocimiento/aportes/python_para_todos.pdf
- Python+RegExp. Expresiones regulares con Python desde *Python para impacientes*. http://python-para-impacientes.blogspot.com.es/2014/02/expresiones-regulares.html
- RegExp. Libro sobre Expresiones Regulares de Dan Nguyen "*The Bastards Book of Regular Expressions*" (pdf): http://regex.bastardsbook.com/files/bastards-regexes.pdf
- Flask. Documentación oficial de Flask. http://flask.pocoo.org/docs/0.10/
  - Flask. Tutorial rápido del uso de Flask. http://code.tutsplus.com/tutorials/an-introduction-to-pythons-flask-framework--net-28822