Seminario: Python+Django

+

Ingeniería del Software orientada al desarrollo web.

Autores (y emails):

Raúl Jiménez Ortega, Jesús Molina López, Ana Lallena Arquillo, Raúl López Campos y Jesús Javier Nuño García

Índice

¿Qué es Python?

Historia y propiedades

Por: Ana Lallena Arquillo

¿Qué es Python?

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90.

Es un lenguaje interpretado e interactivo, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos (luego explicaremos esto).

Permite:

- Programación OO
- Programación modular
- Programación declarativa
- Programación funcional

¿Qué es Python?

(Transparencia añadida con posterioridad al seminario)

Python es usado por la NASA

Y algunas de las entidades más conocidas (Google, Youtube, ...) también hacen reseñas acerca del lenguaje que puedes ver AQUÍ

¿Qué es Django?

Historia, propiedades y filosofía

¿Qué es Django? (I)

Django es un marco de desarrollo web sobre Python que permite desarrollar rápidamente aplicaciones web.

Proviene del mundo periodístico:

- World online en Kansas

La idea es poder montar sitios nuevos y añadir contenidos de forma muy (muy) rápida y dinámica

¿Qué es Django? (II)

Muy escalable (habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos)

Sigue el principio **DRY**

"Don't Repeat Yourself"

¡¡No te repitas!!

¿Por qué usar Python+Django?

¿Por qué usar Python+Django? (I)

El rápido avance de Internet a lo largo de los últimos años ha incentivado al desarrollo des-estructurado de la Web.

En un principio las páginas web no eran más que contenido estático (HTML) que ofrecía información estática.

Gracias a la aparición de un amplio número de tecnologías aplicables a la web y de la evolución y creación de nuevos navegadores se hace indispensable el seguimiento de:

- Estándares (W3C, Usabilidad y Accesibilidad)
- Técnicas de Ingeniería del Software.

¿Por qué usar Python+Django? (II)

Hoy en día se pueden crear aplicaciones Web de la envergadura de cualquier aplicación de escritorio (Google Docs, Amazon, EyeOS,...), y para esto el uso de lenguajes de programación y bases de datos es casi imprescindible.

¿Por qué usar Python+Django?

1. Obliga a usar la arquitectura "Modelo Vista Controlador"
2. Ofrece una interfaz transparente a la BD (ORM)
3. Una vez declaradas las clases en el modelo (models.py), Django puede crear la base de datos por nosotros

En definitiva: nos ahorra tiempo y nos facilita el desarrollo

Problemas de usar Python+Django

- Una vez sincronizada la BD e introducido contenido en ella no es posible re-sincronizarla si se ha cambiado el modelo (actualmente: Django 0.96)
- El proceso de instalación y configuración es bastante complejo: requiere configuración de variables de entorno, etc.

Breve introducción a Python

Por: Jesús Molina López

Breve introducción a Python (I)

El lenguaje interpretado Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado.

En Python, como en Java y muchos otros lenguajes, **el código fuente se traduce a un pseudo código máquina intermedio** llamado bytecode la primera vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones.

Breve introducción a Python (II)

Fuertemente tipado, pero con tipos dinámicos

Una variable puede cambiar de tipo sobre la marcha, pero una vez asignado un valor se tendrá en cuenta el tipo para realizar las operaciones.

No es necesario declarar las variables

Al utilizarlas serán declaradas de forma automática.

Breve introducción a Python (III)

Multiplataforma

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

Además de funcionar en practicamente cualquier plataforma (móviles nokia, PDAs, automatas, PCs empotrados, ...) también funciona en todas las plataformas que soportan JAVA (Jython) y .NET (IronPython)

Breve introducción a Python (IV)

- Los tipos primitivos, así como las definiciones de una clase, son objetos.
- Algunos tipos primitivos:

Tipo	Clase	Notas	Ejemplo
str	String	Inmutable	'Wikipedia'
unicode	String	Versión Unicode de str	u'Wikipedia'
list	Secuencia	Mutable, puede contener diversos tipos	[4.0, 'string', True]
tuple	Secuencia	Immutable	(4.0, 'string', True)
set	Conjunto	Mutable, sin orden, no contiene duplicados	set([4.0, 'string', True])
frozenset	Conjunto	Inmutable, sin orden, no contiene duplicados	frozenset([4.0, 'string', True])
dict	Mapping	Grupo de pares claves, valor	{'key1': 1.0, 'key2': False}
int	Número entero	Precisión fija	42
long	Número entero	Precisión arbitraria	42L ó 456966786151987643L
float	Número	Coma flotante	3.1415927

Breve introducción a Python (V)

Métodos de clase y de instancia:

- @classmethod
  def metodoclase(cls):
    - return "soy un metodo de clase"
- def metodoinstancia(self):
    - return "soy un metodo de instancia"

```
x = Ejemplo()
print Ejemplo.metodoclase()
print x.metodoinstancia()
```

Herencia:

```
class nombreClaseDerivada (nombreClaseBase):
   <sentencia1>...
```

Breve introducción a Python (VI)

Otras características:

- No utiliza llaves "{" "}". Se basa en la sangría para determinar si una instrucción pertenece a un bloque de instrucciones.
- Permite añadir y eliminar atributos en caliente.
- En Python, todo es un objeto (incluso las clases). Las clases, al ser objetos, son instancias de una metaclase. Python además soporta herencia múltiple y polimorfismo.

Breve introducción a Django

Por: Jesús Javier Nuño García

Breve introducción a Django (I)

- Django es un Framework (plataforma de ayuda a la programación) implementada en Python.

- Patrón de diseño MVT ~= MVC
  - Model > Modelo de datos
  - View > Vistas de los datos
  - Template > Plantillas de páginas
- Componentes principales:
  - Mapeador objeto relacional (**ORM**)
  - Gestor de URLs (usa URLs amigables)
  - Sistema de plantillas
  - Interfaz de administración automática

Breve Introducción a Django (II)

Django nos obliga a usar la Ingeniería del Software (MVC) y además nos ofrece ORM (Object Relational Mapping) = Interfaz de acceso a BD orientada a objetos en la que:

- Se trabaja con los datos de las tablas como objetos
- Genera SQL optimizado
- Se nos ofrece la posibilidad de dejar la creación de tablas de la BD a Django.
- Nos permite aplicar relaciones entre objetos a los datos de la BD: agregación, herencia, etc.

Breve introducción a Django (III)

Etapas del desarrollo de una aplicación usando Django:

- Crear el modelo de datos (models.py)
- Instalarlo en la base de datos (manage.py syncdb)
- Definir las URLs del sitio (urls.py)
- Crear las vistas necesarias para los usuarios finales (views.py)
- Diseñar las plantillas

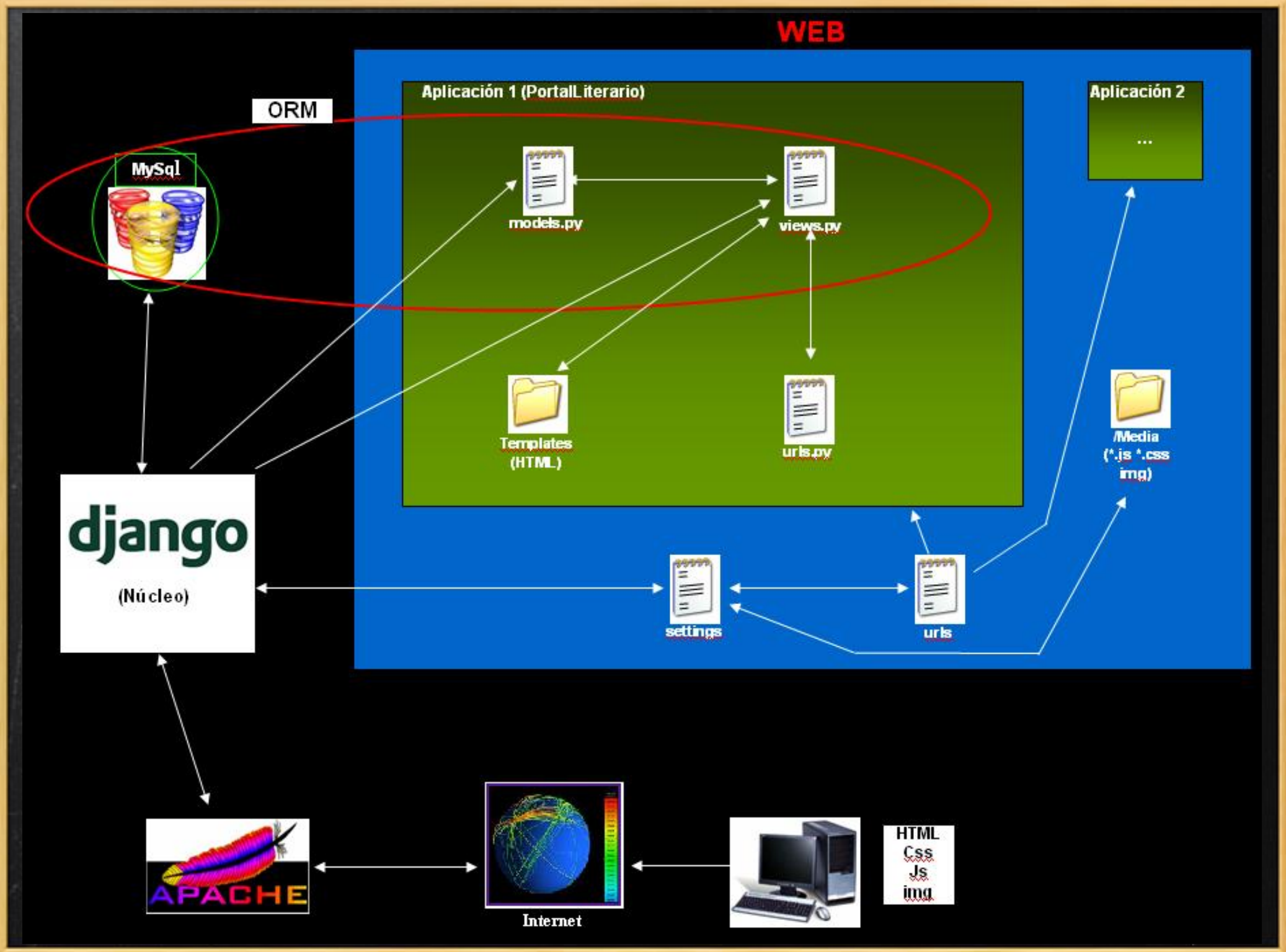
EJEMPLO:

Proceso de servicio de una página Web desarrollada utilizando Python + Django

Proceso de Servicio de una Página (I)

Los roles de Modelo Vista y Controlador son cubiertos de la siguiente manera:

- MODELO (Modelo Django) - En él se definen la jerarquía de clases de la aplicación
- VISTA (vistas Django + plantillas Django) - Su función es tratar los datos mediante objetos
- CONTROLADOR (el propio Django) - Establece las comunicaciones


WEB
ORM
MySql
Aplicación 1 (PortalLiterario)
models.py
views.py
Templates
(HTML)
urls.py
Aplicación 2
...
/Media
(*.js *.css
img)
django
(Núcleo)
settings
urls
APACHE
Internet
HTML
Css
Js
img

Proceso de Servicio de una Página (II)

1.- El USUARIO abre su cliente (Mozilla p. ej.) y abre una página web.

2.- El CLIENTE mediante el protocolo HTTP solicita al SERVIDOR (p. ej., un pc con Apache) la resolución de una URL (la página correspondiente a dicha URL).

3.- Apache detecta que la petición corresponde a nuestra aplicación y solicita al CONTROLADOR que tramite la petición.

4.- El controlador identifica la VISTA a la que corresponde la petición y delega en ella.

Proceso de Servicio de una Página (III)

5.- La VISTA utiliza el MODELO para conocer la estructura de clases con las que va a trabajar y realiza la gestión de datos contra la base de datos usando ORM.

6.- Al terminar la VISTA procesa los datos mediante una PLANTILLA que devolverán como respuesta resultante código [(X) HTML]+[CSS]+[Javascript] al SERVIDOR Apache.

7.- El SERVIDOR le enviará como respuesta a su petición HTTP el código generado dinámicamente.

Por qué funciona y cómo

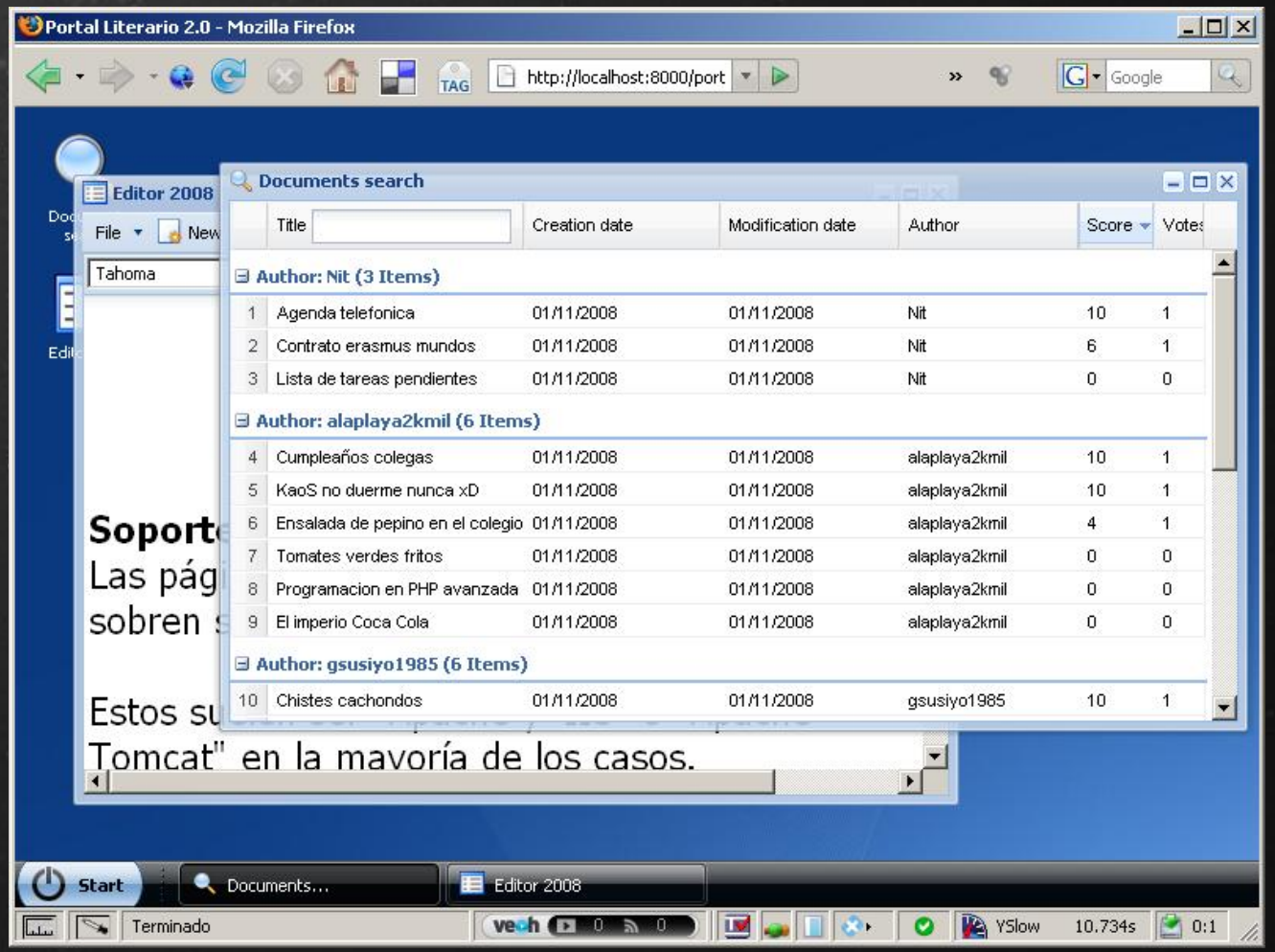
Un ejemplo práctico:

Web Portal Literario 2.0

Por:

Raúl Jiménez Ortega & Raúl López Campos

Portal Literario 2.0 - Ejemplo práctico



Esta es una aplicación Open Source. Puedes encontrar el código en: http://code.google.com/p/pdoweb

Orientación a objetos y el PortalLiterario

Las 3 siguientes transparencias han sido incluidas después de la presentación por lo que **no** caerán en el examen, pero para los que estén interesados voy a hacer otra reflexión acerca de la OO y esta aplicación PortalLiterario.

Las aplicaciones que forman una web son:

- La aplicación servidor: que corre en la máquina donde se almacena la web (normalmente distinta a la del cliente) . La forman Servidor web(encargado de recibir los mensajes) y la aplicación desarrollada.
- Y la aplicación cliente: que corre en la máquina del visitante (gracias al intérprete del navegador)

Y en **nuestro caso** se ejecutan del siguiente modo:

- El servidor web (por ej: IIS, Apache,..) se mantiene corriendo en la memoria RAM del servidor y a la espera de mensajes durante todo el tiempo (normalmente 24h al día, 365 días al año).
- La aplicación servidor **crea una instancia** de la **aplicación cliente*** como respuesta a la primera petición y esta **es devuelta al visitante**, para ser interpretada por en el navegador. Esta se mantiene viva en la RAM mientras el visitante tenga la página abierta (osea 2min, 10min, 3h, ..).

*Esta aplicación que es devuelta al visitante contiene la interfaz gráfica y el código de programación (Javascript) que permite no sólo ver sino también interactuar con la web. Javascript también se utiliza como soporte para la comunicación mediante mensajes con el servidor.

Orientación a objetos y el PortalLiterario

El protocolo de red utilizado para la comunicación mediante mensajes usado en la web es HTTP y una técnica que nos permite el envío de mensajes si forzar la recarga completa de la página es

AJAX (acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML) **.**

Y al igual que toda aplicación se necesita un soporte físico para almacenar los datos (datos del cliente, sus acciones, etc.). En la web se pueden utilizar tanto el cliente (mediante cookies) como el servidor (usando bases de datos, ficheros de texto plano, ficheros binarios, etc) para almacenar cierta información, aunque no es recomendable guardar información vital en el cliente por motivos de seguridad. y eficiencia.

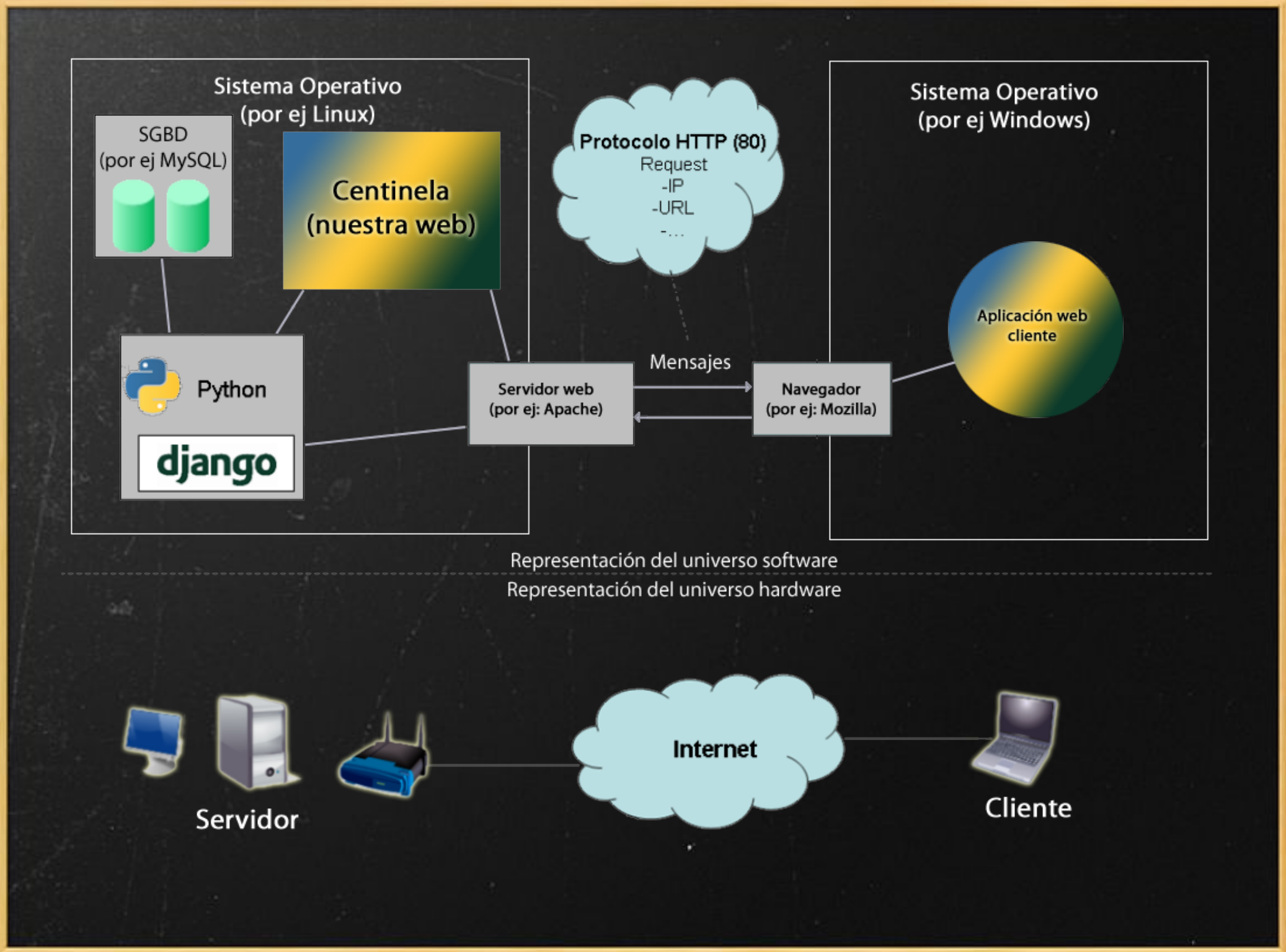
En cuanto al estado de los objetos, el estado del cliente se mantiene en la misma aplicación del cliente y el estado de la aplicación del servidor se tiene que almacenar en el soporte físico ya que los objetos creados en la petición mueren al devolver la respuesta al servidor web* (Apache/IIS) encargado de devolver la respuesta al cliente (en formato JSON).

*Existe en modo para usar Django para servir páginas pero es más recomendable no usarlo

Orientación a objetos y el PortalLiterario

Esto es todo, espero que haya aclarado un poco el comportamiento de esta aplicación que está dotada de una combinación de las últimas y más potentes tecnologías web actuales con el fin de crear una web más potente, rápida y estructurada:

- XHTML - Sirve como soporte para estructurar los contenidos y de puente entre los lenguajes
- CSS - Que permite establecer el diseño gráfico
- Javascript + Framework ExtJS - Dota de autonomía a la aplicación cliente
- Python + Framework Django - Ofrece un soporte potente y organizado para la resolución de peticiones


Sistema Operativo
(por ej Linux)
SGBD
(por ej MySQL)
Centinela
(nuestra web)
Python
django
Protocolo HTTP (80)
Request
-IP
-URL
-...
Mensajes
Servidor web
(por ej: Apache)
Navegador
(por ej: Mozilla)
Sistema Operativo
(por ej Windows)
Aplicación web
cliente
Representación del universo software
Representación del universo hardware
Servidor
Internet
Cliente

Bibliografía

Bibliografía

2ª edición del curso:

Desarrollo rápido de aplicaciones Web 2.0 con Python y Django

Impartido en Septiembre de 2007 por:

Francisco Javier Nievas Muñoz

Miguel Hernández Martos

José Carlos Calvo Tudela

Otras referencias

Algunos de los recursos han sido extraidos de las siguientes fuentes:

- Wikipedia
- Blog de GenBeta

Bibliografía complementaria

Python:

Tutoriales sobre Python (inglés) | Metaclases en Python (castellano)

Django:

http://www.djangoproject.com/documentation (inglés)

http://apress.com/book/view/1590597257 (libro)

Ext JS:

http://www.extjs-tutorial.com (Castellano)

Inglés: API , Documentación

Árbol sobre familia de tecnologías web