DESARROLLO WEB EN ENTORNO SERVIDOR

Desarrollo de Aplicaciones Web

UNIDAD 1: SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN WEB

Índice

Características de la programación web.

Seguro que ya sabes exactamente qué es una página web, e incluso conozcas cuáles son los pasos que se suceden para que, cuando visitas una web poniendo su dirección en el navegador, la página se descargue a tu equipo y se pueda mostrar. Sin embargo, este procedimiento que puede parecer sencillo, a veces no lo es tanto. Todo depende de cómo se haya hecho esa página.

Cuando una página web se descarga a tu ordenador, su contenido define qué se debe mostrar en pantalla. Este contenido está programado en un lenguaje de marcado, formado por etiquetas, que puede ser HTML o XHTML. Las etiquetas que componen la página indican el objetivo de cada una de las partes que la componen. Así, dentro de estos lenguajes hay etiquetas para indicar que un texto es un encabezado, que forma parte de una tabla, o que simplemente es un párrafo de texto.

Además, si la página está bien estructurada, la información que le indica al navegador el estilo con que se debe mostrar cada parte de la página estará almacenado en otro fichero, una hoja de estilos o CSS (*Abreviatura de "Hoja de estilos en cascada", del inglés Cascading Style Sheet (CSS). Es un lenguaje utilizado para definir las características de presentación de un documento escrito en lenguaje HTML, XHTML o XML*). La hoja de estilos se encuentra indicada en la página web y el navegador la descarga junto a ésta. En ella nos podemos encontrar, por ejemplo, estilos que indican que el encabezado debe ir con tipo de letra Arial y en color rojo, o que los párrafos deben ir alineados a la izquierda. Estos dos ficheros se descargan a tu ordenador desde un servidor web como respuesta a una petición. El proceso es el siguiente:

1. Tu ordenador solicita a un servidor web una página con extensión .htm, .html o .xhtml.
2. El servidor busca esa página en un almacén de páginas (cada una suele ser un fichero).
3. Si el servidor encuentra esa página, la recupera.
4. Y por último se la envía al navegador para que éste pueda mostrar su contenido.

Este es un ejemplo típico de una comunicación cliente-servidor. El cliente es el que hace la petición e inicia la comunicación, y el servidor es el que recibe la petición y la atiende. En nuestro caso, el navegador es el cliente web.

¿Podemos ver una página web sin que intervenga un servidor web?

Sí

No

Podemos ver páginas web con extensión .htm, .html o .xhtml que tengamos almacenadas en nuestro equipo simplemente

abriéndolas con

el navegador. En este caso la única utilidad del servidor web es enviar la página que solicitemos a nuestro equipo.

1.1 Páginas web estáticas y dinámicas.

Las páginas que viste en el ejemplo anterior se llaman páginas web estáticas. Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y su contenido no varía. Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen. La única forma en que pueden cambiar es si un programador la modifica y actualiza su contenido.

En contraposición a las páginas web estáticas, como ya te imaginarás, existen las páginas web dinámicas. Estas páginas, como su nombre indica, se caracterizan porque su contenido cambia en función de diversas variables, como puede ser el navegador que estás usando, el usuario con el que te has identificado, o las acciones que has efectuado con anterioridad.

Dentro de las páginas web dinámicas, es muy importante distinguir dos tipos:

- Aquellas que **incluyen código que ejecuta el navegador**. En estas páginas el código ejecutable, normalmente en **lenguaje JavaScript**, se incluye dentro del HTML (o XHTML) y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código que la acompaña. Este código puede incorporar múltiples funcionalidades que pueden ir desde mostrar animaciones hasta cambiar totalmente la apariencia y el contenido de la página. En este módulo no vamos a ver JavaScript, salvo cuando éste se relaciona con la programación web del lado del servidor.
- Como ya sabes, hay muchas páginas en Internet que no tienen extensión .htm, .html o .xhtml. Muchas de estas páginas tienen extensiones como .php, .asp, .jsp, .cgi o .aspx. En éstas, el contenido que se descarga al navegador es similar al de una página web estática: HTML (o XHTML). Lo que cambia es la forma en que se obtiene ese contenido. Al contrario de lo que vimos hasta ahora, esas páginas no están almacenadas en el servidor; más concretamente, el contenido que se almacena no es el mismo que después se envía al navegador. El HTML de estas páginas se forma como resultado de la ejecución de un programa, y esa ejecución tiene lugar en el servidor web (aunque no necesariamente por ese mismo servidor).

El esquema de funcionamiento de una página web dinámica es el siguiente:

1. El cliente web (navegador) de tu ordenador solicita a un servidor web una página web.
2. El servidor busca esa página y la recupera.
3. En el caso de que se trate de una página web dinámica, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el servidor web contacta con el módulo responsable de ejecutar el código y se lo envía.

4. Como parte del proceso de ejecución, puede ser necesario obtener información de algún repositorio (Cualquier almacén de información digital, normalmente una base de datos), como por ejemplo consultar registros almacenados en una base de datos.
5. El resultado de la ejecución será una página en formato HTML, similar a cualquier otra página web no dinámica.
6. El servidor web envía el resultado obtenido al navegador, que la procesa y muestra en pantalla.

Este procedimiento tiene lugar constantemente mientras consultamos páginas web. Por ejemplo, cuando consultas tu correo en GMail, HotMail, Yahoo o cualquier otro servicio de correo vía web, lo primero que tienes que hacer es introducir tu nombre de usuario y contraseña. A continuación, lo más habitual es que el servidor te muestre una pantalla con la bandeja de entrada, en la que aparecen los mensajes recibidos en tu cuenta. Esta pantalla es un claro ejemplo de una página web dinámica.Obviamente, el navegador no envía esa misma página a todos los usuarios, sino que la genera de forma dinámica en función de quién sea el usuario que se conecte. Para generarla ejecuta un programa que obtiene los datos de tu usuario (tus contactos, la lista de mensajes recibidos) y con ellos compone la página web que recibes desde el servidor web.

1.1.1 Ventajas e inconvenientes.

Aunque la utilización de páginas web dinámicas te parezca la mejor opción para construir un sitio web, no siempre lo es. Sin lugar a dudas, es la que más potencia y flexibilidad permite, pero las páginas web estáticas tienen también algunas ventajas:

- No es necesario saber programar para crear un sitio que utilice únicamente páginas web estáticas. Simplemente habría que conocer HTML/XHTML y CSS, e incluso esto no sería indispensable: se podría utilizar algún programa de diseño web para generarlas.
- La característica diferenciadora de las páginas web estáticas es que su contenido nunca varía, y esto en algunos casos también puede suponer una ventaja. Sucede, por ejemplo, cuando quieres almacenar un enlace a un contenido concreto del sitio web: si la página es dinámica, al volver a visitarla utilizando el enlace su contenido puede variar con respecto a cómo estaba con anterioridad. O cuando quieres dar de alta un sitio que has creado en un motor de búsqueda como Google.

Para que Google muestre un sitio web en sus resultados de búsqueda, previamente tiene que indexar su contenido. Es decir, un programa recorre las páginas del sitio consultando su contenido y clasificándolo. Si las páginas se generan de forma dinámica, puede ser que su contenido, en parte o por completo, no sea visible para el buscador y por tanto no quedará indexado. Esto nunca sucedería en un sitio que utilizase páginas web estáticas.

Como ya sabes, para que un servidor web pueda procesar una página web dinámica, necesita ejecutar un programa. Esta ejecución la realiza un módulo concreto, que puede estar integrado en el servidor o ser independiente.

Además, puede ser necesario consultar una base de datos como parte de la ejecución del programa. Es decir, la ejecución de una página web dinámica requiere una serie de recursos del lado del servidor. Estos recursos deben instalarse y mantenerse. Las páginas web estáticas sólo necesitan un servidor web que se comunique con tu navegador para enviártela. Y de hecho para ver una página estática almacenada en tu equipo no necesitas siquiera de un servidor web. Son archivos que pueden almacenarse en un soporte de almacenamiento como puede ser un disco óptico o una memoria USB y abrirse desde él directamente con un navegador web.

Pero si decides hacer un sitio web utilizando páginas estáticas, ten en cuenta que tienen limitaciones. La desventaja más importante ya la comentamos anteriormente: la actualización de su contenido debe hacerse de forma manual editando la página que almacena el servidor web. Esto implica un mantenimiento que puede ser prohibitivo en sitios web con gran cantidad de contenido.

¿Qué tipo de tecnología emplearías para crear una página web personal? ¿Sería necesario utilizar páginas dinámicas? ¿Qué tareas de actualización y mantenimiento tendrías que realizar en cada caso?

Las primeras páginas web que se crearon en Internet fueron páginas estáticas. A esta web compuesta por páginas estáticas se le considera la primera generación. La segunda generación de la web surgió gracias a las páginas web dinámicas. Tomando como base las web dinámicas, han ido surgiendo otras tecnologías que han hecho evolucionar Internet hasta llegar a lo que ahora conocemos.

Realizar de AC1 punto 1.

1.2 Aplicaciones web.

Las aplicaciones web emplean páginas web dinámicas para crear aplicaciones que se ejecuten en un servidor web y se muestren en un navegador. Puedes encontrar aplicaciones web para realizar múltiples tareas. Unas de las primeras en aparecer fueron las que viste antes, los clientes de correo, que te permiten consultar los mensajes de correo recibidos y enviar los tuyos propios utilizando un navegador.

Hoy en día existen aplicaciones web para multitud de tareas como procesadores de texto, gestión de tareas, o edición y almacenamiento de imágenes. Estas aplicaciones tienen ciertas ventajas e inconvenientes si las comparas con las aplicaciones tradicionales que se ejecutan sobre el sistema operativo de la propia máquina.

1.2.1 Ventajas de las aplicaciones web:

- No es necesario instalarlas en aquellos equipos en que se vayan a utilizar. Se instalan y se ejecutan solamente en un equipo, en el servidor, y esto es suficiente para que se puedan utilizar de forma simultánea desde muchos equipos.
- Como solo se encuentran instaladas en un equipo, es muy sencillo gestionarlas (hacer copias de seguridad de sus datos, corregir errores, actualizarlas).
- Se pueden utilizar en todos aquellos sistemas que dispongan de un navegador web, independientemente de sus características (no es necesario un equipo potente) o de su sistema operativo.
- Se pueden utilizar desde cualquier lugar en el que dispongamos de conexión con el servidor. En muchos casos esto hace posible que se pueda acceder a las aplicaciones desde sistemas no convencionales, como por ejemplo teléfonos móviles.

1.2.2 Inconvenientes de las aplicaciones web:

- El interface de usuario de las aplicaciones web es la página que se muestra en el navegador. Esto restringe las características del interface a aquellas de una página web.
- Dependemos de una conexión con el servidor para poder utilizarlas. Si nos falla la conexión, no podremos acceder a la aplicación web.
- La información que se muestra en el navegador debe transmitirse desde el servidor. Esto hace que cierto tipo de aplicaciones no sean adecuadas para su implementación como aplicación web (por ejemplo, las aplicaciones que manejan contenido multimedia, como las de edición de vídeo).

Hoy en día muchas aplicaciones web utilizan las ventajas que les ofrece la generación de páginas dinámicas. La gran mayoría de su contenido está almacenado en una base de datos. Aplicaciones como Drupal, Joomla! y otras muchas ofrecen dos partes bien diferenciadas:

Una parte externa o front-end, que es el conjunto de páginas que ven la gran mayoría de usuarios que las usan (usuarios externos).

Una parte interna o back-end, que es otro conjunto de páginas dinámicas que utilizan las personas que producen el contenido y las que administran la aplicación web (usuarios internos) para crear contenido, organizarlo, decidir la apariencia externa, etc.

¿Cuál de las siguientes no es una característica de una aplicación web?

Sólo es necesario instalarla una vez.

Se crea a partir de páginas web dinámicas.

Se puede utilizar en múltiples sistemas.

Sólo necesita un servidor web para ejecutarse.

Las aplicaciones web emplean páginas web dinámicas. Y como ya vimos, para ejecutar páginas web dinámicas se necesitan una serie de

recursos adicionales en el servidor, como un módulo que ejecute el código o un sistema gestor de bases de datos.

1.3 Ejecución de código en el servidor y en el cliente.

Como vimos, cuando tu navegador solicita a un servidor web una página, es posible que antes de enviártela haya tenido que ejecutar, por sí mismo o por delegación, algún programa para obtenerla. Ese programa es el que genera, en parte o en su totalidad, la página web que llega a tu equipo. En estos casos, el código se ejecuta en el entorno del servidor web.

Además, cuando una página web llega a tu navegador, es también posible que incluya algún programa o fragmentos de código que se deban ejecutar. Ese código, normalmente en lenguaje JavaScript, se ejecutará en tu navegador y, además de poder modificar el contenido de la página, también puede llevar a cabo acciones como la animación de textos u objetos de la página o la comprobación de los datos que introduces en un formulario.

Estas dos tecnologías se complementan una con otra. Así, volviendo al ejemplo del correo web, el programa que se encarga de obtener tus mensajes y su contenido de una base de datos se ejecuta en el entorno del servidor, mientras que tu navegador ejecuta, por ejemplo, el código encargado de avisarte cuando quieres enviar un mensaje y te has olvidado de poner un texto en el asunto. Esta división es así porque el código que se ejecuta en el cliente web (en tu navegador) no tiene, o mejor dicho tradicionalmente no tenía, acceso a los datos que se almacenan en el servidor. Es decir, cuando en tu navegador querías leer un nuevo correo, el código Javascript que se ejecutaba en el mismo no podía obtener de la base de datos el contenido de ese mensaje. La solución era crear una nueva página en el servidor con la información que se pedía y enviarla de nuevo al navegador.
Sin embargo, desde hace unos años existe una técnica de desarrollo web conocida como AJAX, que nos posibilita realizar programas en los que el código JavaScript que se ejecuta en el navegador pueda comunicarse con un servidor de Internet para obtener información con la que, por ejemplo, modificar la página web actual.

En nuestro ejemplo, cuando pulsas con el ratón encima de un correo que quieres leer, la página puede contener código Javascript que detecte la acción y, en ese instante, consultar a través de Internet el texto que contiene ese mismo correo y mostrarlo en la misma página, modificando su estructura en caso de que sea necesario. Es decir, sin salir de una página poder modificar su contenido en base a la información que se almacena en un servidor de Internet.

En este módulo vas a aprender a crear aplicaciones web que se ejecuten en el lado del servidor. Otro módulo de este mismo ciclo, Desarrollo Web en Entorno Cliente, enseña las características de la programación de código que se ejecute en el navegador web.

Muchas de las aplicaciones web actuales utilizan estas dos tecnologías: la ejecución de código en el servidor y en el cliente. Así, el código que se ejecuta en el servidor genera páginas web que ya incluyen código destinado a su ejecución en el navegador. Hacia el final de este módulo verás las técnicas que se usan para programar aplicaciones que incorporen esta funcionalidad.

Si quieres verificar que la contraseña introducida en una página web tenga una longitud mínima, ¿dónde sería preferible que se ejecutara el código de comprobación?

En el navegador web.

En el servidor web.

Obviamente; no es necesario enviar la contraseña al servidor web para comprobar su longitud, cuando esa tarea se puede hacer perfectamente en el navegador.

Realizar de AC1 puntos 2 y 3.

Tecnologías para programación web del lado del servidor.

Cuando programas una aplicación, utilizas un lenguaje de programación. Por ejemplo, utilizas el lenguaje Java para crear aplicaciones que se ejecuten en distintos sistemas operativos. Al programar cada aplicación utilizas ciertas herramientas como un entorno de desarrollo o librerías de código. Además, una vez acabado su desarrollo, esa aplicación necesitará ciertos componentes para su ejecución, como por ejemplo una máquina virtual de Java.

En este bloque vas a aprender las distintas tecnologías que se pueden utilizar para programar aplicaciones que se ejecuten en un servidor web, y cómo se relacionan unas con otras. Verás las ventajas e inconvenientes de utilizar cada una, y qué lenguajes de programación deberás aprender para utilizarlas.

Los componentes principales con los que debes contar para ejecutar aplicaciones web en un servidor son los siguientes:

- Un servidor web para recibir las peticiones de los clientes web (normalmente navegadores) y enviarles la página que solicitan (una vez generada puesto que hablamos de páginas web dinámicas). El servidor web debe conocer el procedimiento a seguir para generar la página web: qué módulo se encargará de la ejecución del código y cómo se debe comunicar con él.
- El módulo encargado de ejecutar el código o programa y generar la página web resultante. Este módulo debe integrarse de alguna forma con el servidor web, y dependerá del lenguaje y tecnología que utilicemos para programar la aplicación web.
- Una aplicación de base de datos, que normalmente también será un servidor. Este módulo no es estrictamente necesario pero en la práctica se utiliza en todas las aplicaciones web que utilizan grandes cantidades de datos para almacenarlos.
- El lenguaje de programación que utilizarás para desarrollar las aplicaciones.

2.1 Plataformas Web.

La primera elección que harás antes de comenzar a programar una aplicación web es la plataforma
que vas a utilizar. Hoy en día, puedes elegir entre:

- Java EE (Enterprise Edition), que antes también se conocía como J2EE. Es una plataforma orientada a la programación de aplicaciones en lenguaje Java. Puede funcionar con distintos gestores de bases de datos, e incluye varias librerías y

especificaciones para el desarrollo de aplicaciones de forma modular. Está apoyada por grandes empresas como Oracle, que mantiene Java, o IBM. Es una buena solución para el desarrollo de aplicaciones de tamaño mediano o grande. Una de sus principales ventajas es la multitud de librerías existentes en ese lenguaje y la gran base de programadores que lo conocen.
Para poder ejecutar aplicaciones Java EE en un servidor básicamente tenemos dos opciones: servidores de aplicaciones, que implementan todas las tecnologías disponibles en Java EE, y contenedores de servlets, que soportan solo parte de la especificación. Dependiendo de la magnitud de nuestra aplicación y de las tecnologías que utilice, tendremos que instalar una solución u otra.

- Existen varias implementaciones de servidores de aplicaciones Java EE certificados. Las dos soluciones comerciales más usadas son IBM Websphere y Weblogic de Oracle. También existen soluciones de código abierto como JBoss, Geronimo (de la fundación Apache) o Glassfish.

Puedes consultar una lista con los servidores de aplicaciones Java EE certificados por Sun en la Wikipedia.

http://es.wikipedia.org/wiki/Java_EE#Servidores_de_Aplicaciones_Java_EE_5_certificados

Sin embargo, en la mayoría de ocasiones no es necesario utilizar un servidor de aplicaciones completo, especialmente si no utilizamos EJB (Enterprise JavaBean, que proporcionan un modelo de componentes distribuido estándar del lado del servidor) en nuestras aplicaciones, sino que nos será suficiente un contenedor de servlets. En esta área, destaca Tomcat, la implementación por referencia de un contenedor de servlets, que además es de código abierto.

Una vez instalada la solución que hayamos escogido, tenemos que integrarla con el servidor web que utilicemos, de tal forma que reconozca las peticiones destinadas a servlets y páginas JSP y las redirija.

Otra opción es utilizar una única solución para páginas estáticas y páginas dinámicas. Por ejemplo, el contenedor de servlets Tomcat incluye un servidor HTTP propio que puede sustituir a Apache.

- AMP. Son las siglas de Apache, MySQL y PHP/Perl/Python. Las dos primeras siglas hacen referencia al servidor web (Apache) y al servidor de base de datos (MySQL). La última se corresponde con el lenguaje de programación utilizado, que puede ser PHP, Perl o Python, Dependiendo del sistema operativo que se utilice para el servidor, se utilizan las siglas LAMP (para Linux), WAMP (para Windows) o MAMP (para Mac). También es posible usar otros componentes, como el gestor de bases de datos

PostgreSQL en lugar de MySQL. Todos los componentes de esta arquitectura son de código libre (open source). Es una plataforma de programación que permite desarrollar aplicaciones de tamaño pequeño o mediano con un aprendizaje sencillo. Su gran ventaja es la gran comunidad que la soporta y la multitud de aplicaciones de código libre disponibles.

El módulo mod_php es la forma habitual que se utiliza para ejecutar guiones en PHP utilizando plataformas AMP, y su equivalente para el lenguaje Python es mod_python.

Existen paquetes software que incluyen en una única instalación una plataforma AMP completa. Algunos ni siquiera es necesario instalarlos, e incluso disponen de versiones para distintos sistemas operativos como Linux, Windows o Mac. Uno de los más conocidos es XAMPP.

http://www.apachefriends.org/es/xampp.html

- CGI/Perl. Es la combinación de dos componentes: Perl, un potente lenguaje de código libre creado originalmente para la administración de servidores, y CGI, un estándar para permitir al servidor web ejecutar programas genéricos, escritos en cualquier lenguaje (también se pueden utilizar por ejemplo C o Python), que devuelven páginas web (HTML) como resultado de su ejecución. Es la más primitiva de las arquitecturas que comparamos aquí.
  El principal inconveniente de esta combinación es que CGI es lento, aunque existen métodos para acelerarlo. Por otra parte, Perl es un lenguaje muy potente con una amplia comunidad de usuarios y mucho código libre disponible.

El servidor web para delega en un programa externo la generación de una página web. Esos programas externos se conocen como guiones CGI, independientemente del lenguaje en el que estén programados (aunque se suelen programar en lenguajes de guiones como Perl).

El principal problema de CGI es que cada vez que se ejecuta un guión CGI, el sistema operativo debe crear un nuevo proceso. Esto implica un mayor consumo de recursos y menor velocidad de ejecución. Existen algunas soluciones que aceleran la ejecución, como FastCGI, y también otros métodos para ejecutar guiones en el entorno de un servidor web, por ejemplo el módulo mod_perl para ejecutar en Apache guiones programados en Perl.

- ASP.Net es la plataforma comercial propuesta por Microsoft para el desarrollo de aplicaciones. Es la parte de la plataforma .Net destinada a la generación de páginas web dinámicas. Proviene de la evolución de la anterior tecnología de Microsoft, ASP. El lenguaje de programación puede ser Visual Basic.Net o C#. La arquitectura utiliza el servidor web de Microsoft, IIS, y puede obtener información de varios gestores de bases de datos entre los que se incluye, como no, Microsoft SQL Server.

Una de las mayores ventajas de la plataforma .Net es que incluye todo lo necesario para el desarrollo y el despliegue de aplicaciones. Por ejemplo, tiene su propio entorno de desarrollo, Visual Studio, aunque hay otras opciones disponibles. La mayor desventaja es que se trata de una plataforma comercial de código propietario.

La arquitectura ASP.Net utiliza el servidor IIS de Microsoft, que ya integra soporte en forma de módulos para manejar peticiones de páginas dinámicas ASP y ASP.Net. La utilidad de administración del servidor web incluye funciones de administración de las aplicaciones web instaladas en el mismo.

Realizar de AC1 puntos 4 y 5. Necesitarás también la información del punto 3 de este documento

¿Cuál de estas tecnologías permite la ejecución por el servidor web de programas escritos en cualquier lenguaje?

PHP.

Java EE.

AMP.

CGI.

Efectivamente es correcto, CGI es un estándar que permite al servidor web la ejecución de programas genéricos, escritos en cualquier lenguaje.

Lenguajes y herramientas.

3.1 Tipos de lenguajes

Una de las diferencias más notables entre un lenguaje de programación web y otro es la manera en que se ejecutan en el servidor web. Debes distinguir tres grandes grupos:

- Lenguajes de guiones (scripting). Son aquellos en los que los programas se ejecutan directamente a partir de su código fuente (Conjunto de instrucciones que componen un programa, y que no son ejecutables directamente, sino que deben traducirse utilizando un compilador, intérprete o similar antes de que pueda ser ejecutado por la máquina) original. Se almacenan normalmente en un fichero de texto plano. Cuando el servidor web necesita ejecutar código programado en un lenguaje de guiones, le pasa la petición a un intérprete, que procesa las líneas del programa y genera como resultado una página web.
  De los lenguajes que estudiaste anteriormente, pertenecen a este grupo Perl, Python, PHP y ASP(el precursor de ASP.Net).
- Lenguajes compilados a código nativo (Denominación habitual del lenguaje máquina. Código que puede ser ejecutado directamente por el procesador). Son aquellos en los que el código fuente se traduce a código binario, dependiente del procesador, antes de ser ejecutado. El servidor web almacena los programas en su modo binario, que ejecuta directamente cuando se les invoca.
  El método principal para ejecutar programas binarios desde un servidor web es CGI. Utilizando CGI podemos hacer que el servidor web ejecute código programado en cualquier lenguaje de propósito general como puede ser C.
- Lenguajes compilados a código intermedio. Son lenguajes en los que el código fuente original se traduce a un código intermedio, independiente del procesador, antes de ser ejecutado. Es la forma en la que se ejecutan por ejemplo las aplicaciones programadas en Java, y lo que hace que puedan ejecutarse en varias plataformas distintas. En la programación web, operan de esta forma los lenguajes de las arquitecturas Java EE (servlets y páginas JSP) y ASP.Net.

En la plataforma ASP.Net y en muchas implementaciones de Java EE, se utiliza un procedimiento de compilación JIT. Este término hace referencia a la forma en que se convierte el código intermedio a código binario para ser ejecutado por el procesador. Para acelerar la ejecución, el compilador puede traducir todo o parte del código intermedio a código nativo cuando se invoca a un programa. El código nativo obtenido suele almacenarse para ser utilizado de nuevo cuando sea necesario.

Cada una de estas formas de ejecución del código por el servidor web tiene sus ventajas e inconvenientes.

- Los lenguajes de guiones tienen la ventaja de que no es necesario traducir el código fuente original para ser ejecutados, lo que aumenta su portabilidad. Si se necesita realizar alguna modificación a un programa, se puede hacer en el momento. Por el contrario el proceso de interpretación ofrece un peor rendimiento que las otras alternativas.
- Los lenguajes compilados a código nativo son los de mayor velocidad de ejecución, pero tienen problemas en lo relativo a su integración con el servidor web. Son programas de propósito general que no están pensados para ejecutarse en el entorno de un servidor web. Por ejemplo, no se reutilizan los procesos para atender a varias peticiones: por cada petición que se haga al servidor web, se debe ejecutar un nuevo proceso. Además los programas no son portables entre distintas plataformas.
- Los lenguajes compilados a código intermedio ofrecen un equilibrio entre las dos opciones anteriores. Su rendimiento es muy bueno y pueden portarse entre distintas plataformas en las que exista una implementación de la arquitectura (como un contenedor de servlets o un servidor de aplicaciones Java EE).

Completar AC1 punto 4

3.2 Código que genera lenguaje html y código embebido en el lenguaje de marcas.

Cuando la web comenzó a evolucionar desde las páginas web estáticas a las dinámicas, una de las primeras tecnologías que se utilizaron fue la ejecución de código utilizando CGI. Los guiones CGI son programas estándar, que se ejecutan por el sistema operativo, pero que generan como salida el código HTML de una página web. Por tanto, los guiones CGI deben contener, mezcladas dentro de su código, sentencias encargadas de generar la página web.

Por ejemplo, si quieres generar una página web utilizando CGI a partir de un guión de sentencias en Linux, tienes que hacer algo como lo siguiente:

```
#!/bin/bash
echo "Content-type: text/html"
echo ''
echo 'CGI Bash Example'
echo `df -h / | grep -v Filesystem`
```

Esta es una de las principales formas de realizar páginas web dinámicas: integrar las etiquetas HTML en el código de los programas. Es decir, los programas como el guión anterior incluyen dentro de su código sentencias de salida (echo en el caso anterior) que son las que incluyen el código HTML de la página web que se obtendrá cuando se ejecuten. De esta forma se programan, por ejemplo, los guiones CGI y los servlets.

Un enfoque distinto consiste en integrar el código del programa en medio de las etiquetas HTML de la página web. De esta forma, el contenido que no varía de la página se puede

introducir directamente en HTML, y el lenguaje de programación se utilizará para todo aquello que pueda variar de forma dinámica.

Por ejemplo, puedes incluir dentro de una página HTML un pequeño código en lenguaje PHP que muestre el mensaje “Hola, Mundo”:

```
<html>
<head>
<title>Prueba de PHP</title>
</head>
<body>
<?php echo '<p>Hola, Mundo</p>';?>
</body>
</html>
```

Esta metodología de programación es la que se emplea en los lenguajes ASP, PHP y con las páginas JSP de Java EE.

Los servlets de Java EE se diferencian de las páginas JSP en que los primeros son programas Java compilados y almacenados en el contenedor de servlets. Sin embargo, las páginas JSP contienen código Java embebido en lenguaje HTML y se almacenan de forma individual en el servidor web. La primera vez que se necesita una página JSP, se convierte a un servlet y éste se guarda para ser utilizado en posteriores llamadas a la misma página.

En la arquitectura ASP.Net, cada página se divide en dos ficheros: uno contiene las etiquetas HTML y otro el código en el lenguaje de programación utilizado. De esta forma se logra cierta independencia entre el aspecto de la aplicación y la gestión del contenido dinámico. A partir de esos ficheros se obtiene un código intermedio (MSIL en la terminología de la plataforma) que es lo que almacena el servidor.

La relación entre la forma de ejecución de un lenguaje, y el método para integrarse con las etiquetas HTML de una página web es:

Si el lenguaje integra en su código etiquetas HTML, entonces se trata de un lenguaje de guiones.

Si las instrucciones del lenguaje se integran dentro de las etiquetas HTML de una página web, entonces se trata de un lenguaje de guiones.

Si las instrucciones del lenguaje se integran dentro de las etiquetas HTML de una página web, entonces se trata de un lenguaje compilado.

Es indistinto, no hay una relación directa.

No existe una relación directa entre la forma en que se ejecuta un lenguaje, y cómo se integra con las etiquetas HTML de una página web.

3.3 Herramientas de programación.

A la hora de ponerte a programar una aplicación web, debes tener en cuenta con que herramientas cuentas que te puedan ayudar de una forma u otra a realizar el trabajo. Además de las herramientas que se tengan que utilizar en el servidor una vez que la aplicación se ejecute, como por ejemplo el servidor de aplicaciones o el gestor de bases de datos, de las que ya conoces su objetivo, existen otras que resultan de gran ayuda en el proceso previo, en el desarrollo de la aplicación.

Desde hace tiempo, existen entornos integrados de desarrollo (IDE) que agrupan en un único programa muchas de estas herramientas. Algunos de estos entornos de desarrollo son específicos de una plataforma o de un lenguaje, como sucede por ejemplo con Visual Studio, el IDE de Microsoft para desarrollar aplicaciones en lenguaje C# o Visual Basic para la plataforma .Net. Otros como Eclipse o NetBeans te permiten personalizar el entorno para trabajar con diferentes lenguajes y plataformas, como Java EE o PHP.

No es imprescindible utilizar un IDE para programar. En muchas ocasiones puedes echar mano de un simple editor de texto para editar el código que necesites. Sin embargo, si tu objetivo es desarrollar una aplicación web, las características que te aporta un entorno de desarrollo son muy convenientes.Entre estas características se encuentran:

- Resaltado de texto. Muestra con distinto color o tipo de letra los diferentes elementos del lenguaje: sentencias, variables, comentarios, etc. También genera indentado automático para diferenciar de forma clara los distintos bloques de un programa.
- Completado automático. Detecta qué estás escribiendo y cuando es posible te muestra distintas opciones para completar el texto.
- Navegación en el código. Permite buscar de forma sencilla elementos dentro del texto, por ejemplo, definiciones de variables
- Comprobación de errores al editar. Reconoce la sintaxis del lenguaje y revisa el código en busca de errores mientras lo escribes.
- Generación automática de código. Ciertas estructuras, como la que se utiliza para las clases, se repiten varias veces en un programa. La generación automática de código puede encargarse de crear la estructura básica, para que sólo tengas que rellenarla.
- Ejecución y depuración. Esta característica es una de las más útiles. El IDE se puede encargar de ejecutar un programa para poder probar su funcionamiento. Además, cuando algo no funciona, te permite depurarlo con herramientas como la ejecución paso a paso, el establecimiento de puntos de ruptura o la inspección del valor que almacenan las variables.

- Gestión de versiones. En conjunción con un sistema de control de versiones, el entorno de desarrollo te puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo, para que si es necesario puedas revertir los cambios realizados.

Los dos IDE de código abierto más utilizados en la actualidad son Eclipse y NetBeans. Ambos permiten el desarrollo de aplicaciones informáticas en varios lenguajes de programación. Aunque en sus orígenes se centraron en la programación en lenguaje Java, hoy en día admiten directamente o a través de módulos, varios lenguajes entre los que se incluyen C, C++, PHP, Python y Ruby.

Ambos además ofrecen para la descarga versiones personalizadas del IDE que pueden ser usadas directamente para programar en un lenguaje determinado, sin necesidad de cambiar la configuración o instalar módulos.

3.3.1 Instalación de una plataforma XAMP en Windows.

Una vez instalado el entorno de desarrollo, es necesario instalar el resto de la plataforma de desarrollo. Si vas a programar en lenguaje PHP, necesitas todos los componentes de una arquitectura XAMP; recuerda:

- X de Windows, el sistema operativo.
- A de Apache, el servidor web.
- M de MySQL, el gestor de bases de datos.
- P del lenguaje de programación, que puede ser PHP, Perl o Python. Vamos a instalar el más común de estos tres, PHP.

Mirar Anexo I: Instalación y configuración de XAMPP

Realizar AP1

3.3.2 Instalación de Notepad++ y Aptana para Windows.

Notepad++ es un editor de texto que puede trabajar con distintos lenguajes de programación y nos servirá para realizar nuestros primeros programas en php. Realizar la instalación de Notepad ++

Aptana es un IDE de programación web que nos permitirá trabajar entre otros con el lenguaje php.

Mirar Anexo II: Instalación y configuración de Aptana
Realizar AP2/Realizar AC1 completar punto 5 y realizar el 6

3.3.3 Instalación de XAMPP y Aptana para Linux.

Instalación y configuración de estas herramientas bajo Ubuntu / Linux.

Anexo III. Instalación de Java, XAMPP y Aptana para Ubuntu

¿Cuál de los siguientes elementos no es necesario para programar aplicaciones web en

lenguaje PHP?

Un servidor web.

Un sistema operativo.

El lenguaje de programación PHP.

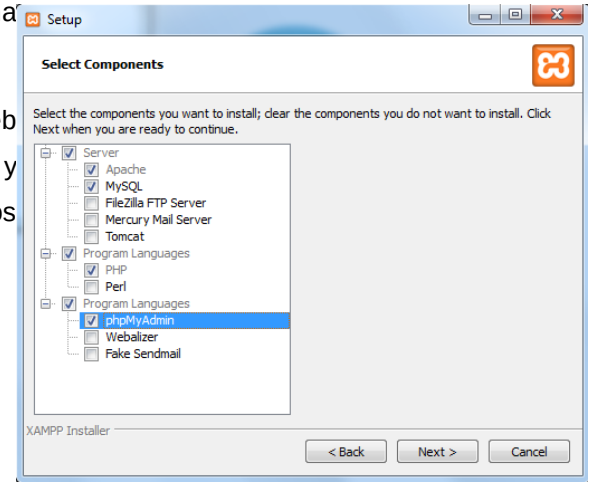
Un entorno integrado de desarrollo.

Efectivamente es correcto, no es imprescindible el uso de un entorno integrado de desarrollo (IDE) para programar, aunque sí muy útil.

Anexos

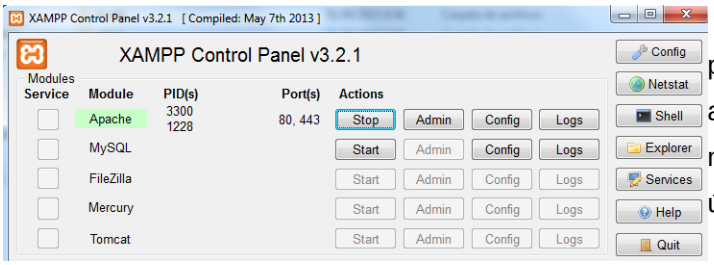
4.1 Anexo I. Instalación y configuración de XAMPP

XAMPP nos permite realizar la instalación de forma conjunta de los componentes anteriores.



Descargaremos la última versión del programa de la web oficial (https://www.apachefriends.org/es/index.html) y procederemos a la instalación. Seleccionaremos sólo los servicios que vamos a utilizar.

Después indicaremos el directorio donde queremos que se instale, que normalmente será C:\xampp, y permitiremos el acceso para que se pueda comunicar con las redes privadas cuando así nos los solicite el Firewall del sistema



Posteriormente abriremos el panel de control de Xampp y activaremos los servidores que necesitemos (de momento únicamente Apache).

Para comprobar que está funcionando correctamente teclea localhost en un navegador web y debería aparecernos la página de inicio(dashboard) de Xampp.

Una vez activado Apache, para poder visualizar nuestras páginas web las deberemos colocar en el directorio C:\xampp\htdocs, o en una subcarpeta del mismo. Si por ejemplo colocamos una página llamada prueba.html en C:\xampp\htdocs, la podremos visualizar en el navegador poniendo localhost/prueba.html. Si esta página estuviera situada en una subcarpeta, por ejemplo en C:\xampp\htdocs\proyecto1\prueba.html, para visualizarla en el navegador tenemos que poner localhost/proyecto1/prueba.html

También podemos tener situada nuestra carpeta de trabajo con nuestros proyectos web en otra unidad o carpeta distinta. Para poder visualizar los proyectos webs situados en estas otras ubicaciones crearemos un vínculo simbólico (symbolic link) que este ubicado en el árbol de C:\xampp\htdocs y que enlace con la otra ubicación.

Por ejemplo, imagina que tienes tus proyectos en D:\Proyectos\. Podemos crear un vínculo que esté situado en C:\xampp\htdocs llamado por ejemplo proyectos situándonos en C:\xampp\htdocs y tecleando en la consola de comando:

mklink /D proyectos D:\Proyectos

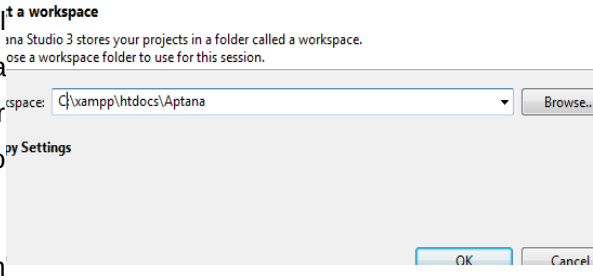
Y ya podríamos acceder a la carpeta proyectos desde el navegador tecleando localhost/proyectos, y añadiendo la página o la ruta adecuada.

4.2 Anexo II: Instalación y configuración de Aptana

Lo descargaremos e instalaremos de la web oficial: http://www.aptana.com/

Actualmente la versión para Windows es de 32 bits, y necesita para funcionar la versión de 32 bits del JDK de Java. Si ya tenemos instalada una versión de 64 bits de java y queremos mantenerla, podríamos tener las 2 e indicar en el fichero de configuración de Aptana (AptanaStudio3.ini) la ruta del compilador de java de 32 bits.

Una vez instalado arrancamos el programa y cambiaremos el espacio de trabajo (workspace) en File→Switch Workspace para que los proyectos web estén en la ruta adecuada para el servidor Apache de XAMPP. Otra posibilidad es crear un vínculo simbólico al workspace desde htdocs(ver Anexo I).



Como se ve en la imagen se ha creado un workspace en C:\xampp\htdocs\Aptana

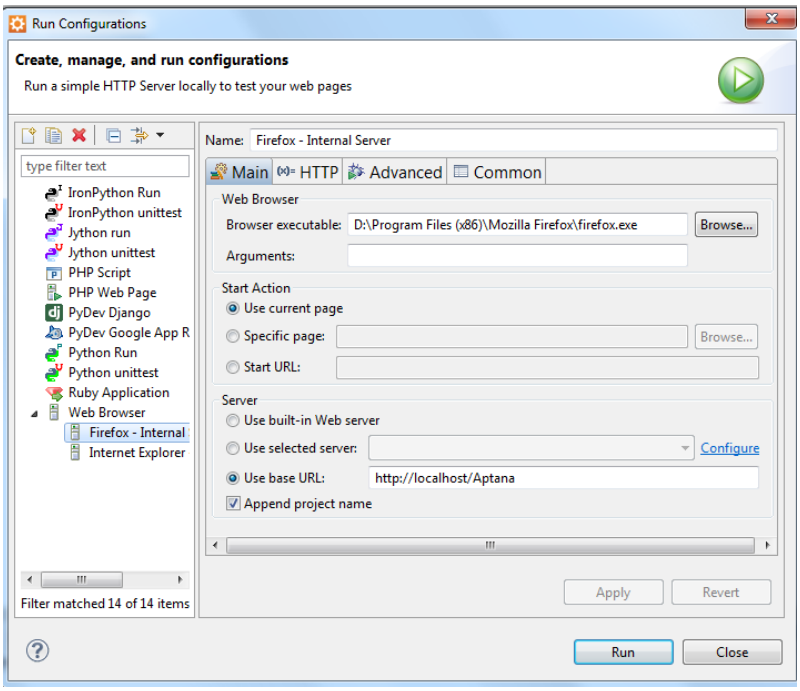
Después crearemos un nuevo proyecto de php (File→New→PHP Project) y le damos un nombre. Con ese nombre se creará una carpeta que situará en el workspace (en.

Dentro de esa carpeta crearemos nuestras carpetas y ficheros. Por ejemplo podemos crear un fichero de prueba de php:

```
<?php
    echo "Hola, mundo";
?>
```

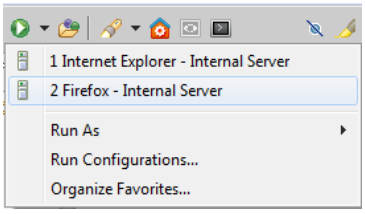
y comprobar si funciona de la siguiente forma:

Abrir el menú de Run->Run Configurations , seleccionar el explorador para el cual queremos hacer la configuración(p.e. Firefox) e indicar un Server y vamos a indicar la dirección URL base (Use base URL) que será la de la carpeta creada para el workspace.



Después ya podremos probar cómo se visualiza la página web pulsando en el icono verde con la flecha

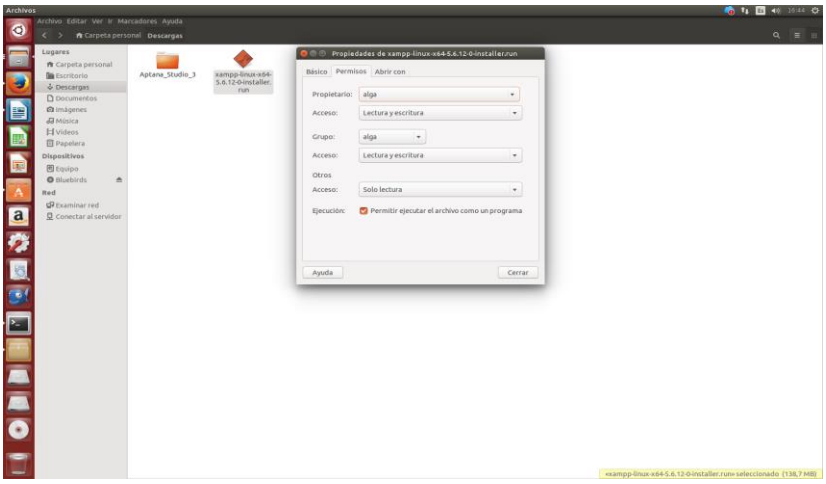
blanca y seleccionando el navegador configurado



4.3 Anexo III. Instalación de Java, XAMPP y Aptana para Ubuntu

1. Instalamos Java (OpenJDK Java runtime, using Hotspot JIT) desde el centro de software de Ubuntu
2. Descargamos los ejecutables de los programa desde sus respectivas páginas web, para XAMPP desde https://www.apachefriends.org/es/index.html y para Aptana desde http://www.aptana.com/

3. **XAMPP**
   a. Abrimos el explorador de archivos Nautilus mediante una terminal en modo root con el comando sudo nautilus y cambiamos el permiso del archivo descargado de xampp para que sea ejecutable y lo ejecutamos posteriormente haciendo doble click.



b. Para iniciar los servidores de xampp lo podemos hacer mediante comando o lanzando una herramienta gráfica mediante la siguiente instrucción desde un terminal (también podemos configurarlo para que se inicien al empezar linux)

```
sudo /opt/lamp/manager-linux-x64.run
```

Si queremos crear un acceso directo en el escritorio para lanzar el script lo podemos hacer de la siguiente manera:
Para crear un lanzador necesitas abrir una consola o terminal y escribir el siguiente comando

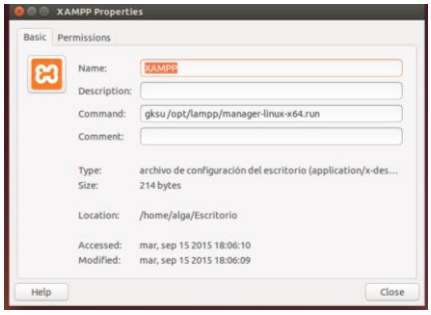
gnome-desktop-item-edit {ubicacion} - -create-new

En donde {ubicacion} es el directorio en donde crearas el lanzador, por ejemplo si lo quieres en el escritorio el comando sería el siguiente:

gnome-desktop-item-edit ~/Escritorio - -create-new

Si no tenemos gnome-panel nos pedirá hacer una instalación que podremos hacer mediante el siguiente comando:

sudo apt-get install gnome-panel



c. Las páginas web para que puedan ser localizadas por el servidor deberán estar en la carpeta /opt/lampp/htdocs o crear un enlace simbólico como hicimos en windows que en este caso se realiza así:

```
sudo ln –s /media/usuario/DATOS/ProyectosAptanaLinux proyectosAptana
```

d. En Ubuntu podemos tener problemas para acceder a los archivos guardados en la unidad de datos si está formateada en NTFS. Para evitar esto tenemos que montar la unidad de datos de la siguiente forma:

1. Desmontar la unidad que vayamos a utilizar (en nuestro caso la de datos)
2. Editar el archivo /etc/fstab en modo root

```
sudo gedit /etc/fstab
```

3. Identificar el UUID de la partición con el comando

```
sudo blkid
```

4. Añadir o modificar la línea en el fichero fstab

```
UUID=12102C02102CEB83 /media/usuario/DATOS ntfs-3g auto,users,permissions 0 0
```

5. Crear el punto de montaje en caso de que sea necesario

```
sudo mkdir /media/usuario/DATOS
```

6. Volver a montar la partición

```
mount /media/usuario/DATOS
```

4. APTANA

a. Lo vamos a instalar en el directo "/opt", que recordemos que es donde se deben de instalar todas las aplicaciones adicionales que no forman parte de la instalación estándar de Ubuntu o de cualquier GNU/Linux.
Entrar al navegador Nautilus (en concreto al directorio /opt) como root con el comando:

```
sudo nautilus
```

b. Crear un nuevo directorio o carpeta en /opt llamado "aptana" (Clic derecho - Crear una carpeta - llamarla aptana)
c. Navegar hasta donde lo hayamos descargado - Clic derecho en el "paquete.zip" y seleccionar "Extraer aquí"
d. Abrir la carpeta descomprimida y copiar todo su contenido, para pegarlo dentro de la nueva carpeta que hemos creado con anterioridad "/opt/aptana". (Ya podemos borrar el paquete.zip y la carpeta descomprimida de nuestro home).
e. Crear un Script para ejecutar Aptana con el comando:

```
sudo gedit /usr/bin/aptana
```

Y pegamos en la ventana que se nos abre, el siguiente código:

```
#!/bin/sh

#export MOZILLA_FIVE_HOME="/usr/lib/mozilla/"

export APTANA_HOME="/opt/aptana"

$APTANA_HOME/AptanaStudio3 $*
```

f. Pulsamos Guardar y cerramos el archivo. Dar los permisos adecuados para los diferentes archivos, ejecutando en una terminal los siguientes comandos uno a uno:

```
sudo chmod 755 /usr/bin/aptana

sudo chmod -R +r /opt/aptana

sudo chmod +x `sudo find /opt/aptana -type d`
```

g. Sólo queda crear un lanzador para que ejecutarlo, en el "menú principal": Para que nos aparezca la aplicación en el menú de "Aplicaciones - Programación" de Ubuntu 10.10 y anteriores y en el Dash Aplicaciones de Ubuntu 14.04 creamos un archivo.desktop en /usr/share/applications/, ejecutando en un terminal:

```
sudo gedit /usr/share/applications/aptana.desktop
```

Pegar el siguiente código en el archivo:

```
[Desktop Entry]

Encoding=UTF-8

Name=Aptana Studio 3

Comment=IDE for Rails, Python, PHP

Exec=/opt/aptana/AptanaStudio3

Icon=/opt/aptana/icon.xpm

Terminal=false
```

```
Type=Application

Categories=GNOME;Application;Development;

StartupNotify=true
```

Guardamos y cerramos el archivo y ya tendremos disponible Aptana desde el Dash de Unity, escribiendo Aptana....

h. Cuando arranquemos Aptana le indicaremos que utilice como workspace el indicado en el punto 3c (`/media/usuario/DATOS/ProyectosAptanaLinux`), al que tenemos apuntando el enlace simbólico proyectosAptana

   Por último configuraremos las opciones de ejecución de Aptana para poder probar las páginas desde el propio entorno. Lo haremos de forma similar a como lo hicimos para Windows, pero en este caso el workspace no está debajo del directorio de htdocs, sino que lo hemos creado en la unidad de datos de Windows a donde apunta el enlace simbólico (proyectosAptana) de htdocs.