FACENA
UNNE

Taller de Programación I

Teoría Tema 4

Facultad de Ciencias Exactas y Naturales y Agrimensura- UNNE

Desarrollo de Aplicaciones Web

Año 2014

Expto. Oscar Zalazar - Expto. Pedro L. Alfonzo - Lic. Yanina Medina - Osvaldo P. Quintana - Lic. Lucía Salazar
[2014]

Autores: Expto. Oscar Zalazar, Expto. Pedro L. Alfonzo, Lic. Yanina Medina, Osvaldo P. Quintana, Lic. Lucía Salazar

Tema 4: Desarrollo de aplicaciones Web

Arquitectura de las aplicaciones Web.

En la ingeniería de software se denomina **aplicación web** a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Las aplicaciones web son un tipo especial de aplicaciones cliente/ servidor. Las aplicaciones web (WebApp) utilizan la *arquitectura* cliente-servidor (Figura 1)

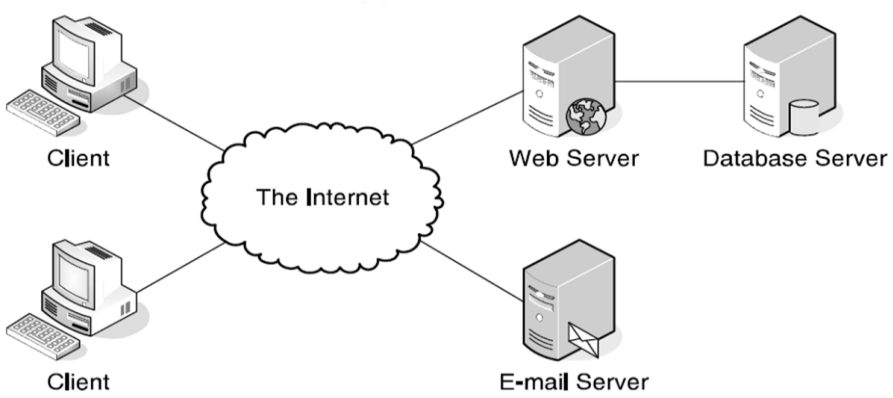


Figura 1. Arquitectura cliente-servidor

Cliente/servidor es una arquitectura de red[1] en la que cada ordenador o proceso en la red es cliente o servidor[2]. Normalmente los servidores son ordenadores potentes dedicados a gestionar unidades de disco (servidor de ficheros), impresoras (servidor de impresoras), tráfico de red (servidor de red), datos (servidor de base de datos), o incluso aplicaciones (servidor de aplicaciones), mientras que los clientes son máquinas menos potentes y usan los recursos que ofrecen los servidores.

Dentro de los clientes se suelen distinguir dos clases: los clientes inteligentes (rich client) y los clientes tontos (thin client). Los primeros son ordenadores completos, con todo el hardware y software necesarios para funcionar de forma independiente.

Los segundos son terminales que no pueden funcionar de forma independiente, ya que necesitan de un servidor para ser operativos.

Esta arquitectura implica la existencia de una relación entre procesos que solicitan servicios (clientes) y procesos que responden a estos servicios (servidores). Estos dos tipos de procesos pueden ejecutarse en el mismo servidor o en distintos.

La arquitectura cliente/ servidor[1] permite la creación de aplicaciones distribuidas.

La principal ventaja de esta arquitectura es que facilita la separación de las funciones según su servicio, permitiendo situar cada función en la plataforma más adecuada para su ejecución. Además presenta también las siguientes ventajas:

1. *Otro tipo de arquitectura de red es peer- to- peer (entre pares o de igual a igual), en la que cada ordenador de la red posee responsabilidades equivalentes.*
2. *Aunque un mismo ordenador puede ser cliente y servidor simultáneamente se establece una separación lógica según las funciones que realiza.*

a- Las redes de ordenadores permiten que múltiples ordenadores puedan ejecutar partes distribuidas de una misma aplicación, logrando concurrencia de procesos.

b- Existe la posibilidad de migrar aplicaciones de un procesador a otro con modificaciones mínimas en los programas.

c- Se obtiene una escalabilidad de la aplicación. Permite la ampliación horizontal o vertical de las aplicaciones. La escalabilidad horizontal se refiere a la capacidad de añadir o suprimir estaciones de trabajo que hagan uso de la aplicación (clientes), sin que afecte sustancialmente al rendimiento general. La estrategia más sencilla para escalar una aplicación web es simplemente comprar hardware mejor y más caro. Si nuestro hardware actual puede servir 100 peticiones por segundo, cuando alcancemos ese límite podemos actualizar la máquina a una con más potencia y así servir más peticiones. Esto se conoce como **escalabilidad vertical**.
La sabiduría popular nos dice que la escalabilidad vertical tiene un límite: llegados a cierto punto, simplemente no hay máquinas más caras, por lo que no se puede seguir escalando verticalmente. Además, el precio de las máquinas suele aumentar exponencialmente con su potencia, por lo que la escalabilidad vertical suele ser una opción cara.

La escalabilidad vertical se refiere a la capacidad de migrar hacia servidores de mayor capacidad o velocidad, o de un tipo distinto de arquitectura sin que afecte a los clientes. El mejor ejemplo de este paradigma es la arquitectura de Google. La plataforma en la que corre el famoso buscador está compuesta de decenas de miles de máquinas de hardware utilitario. Entre tanto hardware barato es normal que cada día decenas de máquinas dejen de funcionar por problemas de hardware, pero eso no supone un problema grave: el sistema es capaz de redirigir automáticamente el tráfico de un nodo caído a los nodos sanos y seguir funcionando como si tal cosa. Para aumentar su capacidad, Google sólo tiene que comprar más máquinas y ponerlas a funcionar. Posibilita el acceso a los datos independientemente de donde se encuentre el usuario.

Separación de funciones

La arquitectura cliente/servidor nos permite la separación de funciones en tres niveles:

La lógica de presentación: Se encarga de la entrada y salida de la aplicación con el usuario. Sus principales tareas son: obtener información del usuario, enviar la información del usuario a la lógica de negocio para su procesamiento, recibir los resultados del procesamiento de la lógica de negocio y presentar estos resultados al usuario.

Lógica de negocio (o aplicación): Se encarga de gestionar los datos a nivel de procesamiento. Actúa de puente entre el usuario y los datos. Sus principales tareas son: recibir la entrada del nivel de presentación, interactuar con la lógica de datos, para ejecutar las reglas de negocios (business rules) que tiene que cumplir la aplicación (facturación, cálculo de nóminas, control de inventarios, etc.) y enviar el resultado del procesamiento al nivel de presentación.

Lógica de datos: Se encarga de gestionar los datos a nivel de almacenamiento. Sus principales tareas son: almacenar los datos, recuperar los datos, mantener los datos y asegurar la integridad de los mismos.

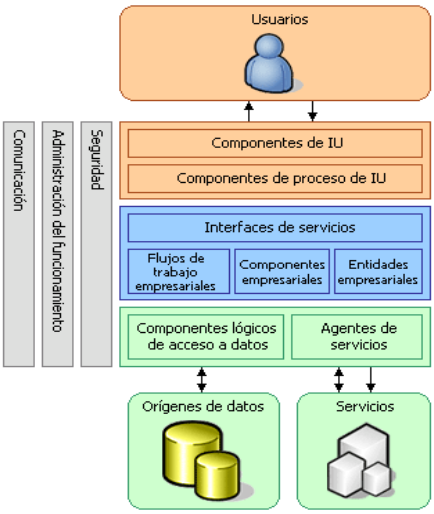
Si un sistema distribuido se diseña correctamente, los tres niveles anteriores pueden distribuirse y redistribuirse independientemente sin afectar al funcionamiento de la aplicación.

Modelos de distribución en aplicaciones cliente/servidor

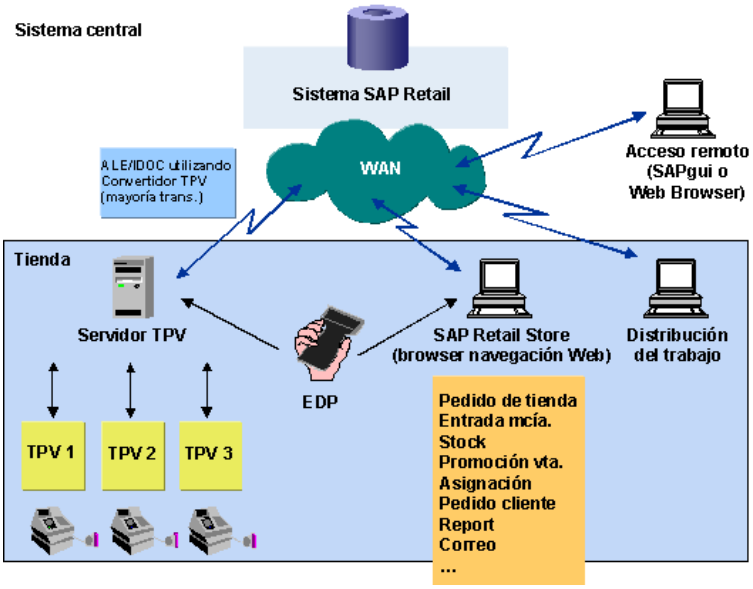
Según como se distribuyan las tres funciones básicas de una aplicación (presentación, negocio y datos) entre el cliente y el servidor, podemos contemplar tres modelos:

Presentación distribuida: El cliente solo mantiene la presentación, el resto de la aplicación se ejecuta remotamente. La presentación distribuida, en su forma más simple, es una interfaz gráfica de usuario a la que se le pueden acoplar controles de validación de datos, para evitar la validación de los mismos en el servidor.

Aplicación distribuida: Es el modelo que proporciona máxima flexibilidad, puesto que permite tanto a servidor como a cliente mantener la lógica de negocio realizando cada uno las funciones que le sean más propias, ya sea por organización, o por mejora en el rendimiento del sistema.



Datos distribuidos: Los datos son los que se distribuyen, por lo que la lógica de datos es lo que queda separada del resto de la aplicación. Se puede dar de dos formas: ficheros distribuidos o bases de datos distribuidas.

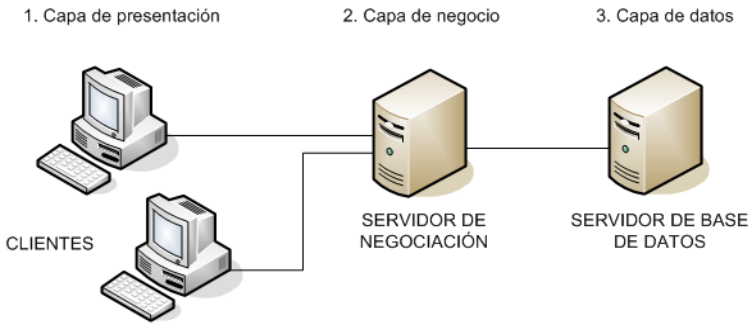


Arquitecturas de dos y tres niveles

La diferencia entre las arquitecturas de dos y tres niveles (capas) estriba en la forma de distribución de la aplicación entre el cliente y el servidor.

Aunque todos los modelos de distribución en aplicaciones cliente/servidor se basan en arquitecturas de dos capas, normalmente cuando se habla de aplicaciones de dos niveles se está haciendo referencia a una aplicación donde el cliente mantiene la lógica de presentación, de negocio y de acceso a datos., y el servidor únicamente gestiona los datos. Suelen ser aplicaciones cerradas que supeditan la lógica de los procesos cliente al gestor de base de datos que se está usando.

En las arquitecturas de tres niveles, la lógica de presentación, la lógica de negocios y la lógica de datos están separadas, de tal forma que mientras la lógica de presentación se ejecutará normalmente en la estación cliente, la lógica de negocio y la de datos pueden estar repartidas entre distintos procesadores. En este tipo de aplicaciones suelen existir dos servidores: uno contiene la lógica de negocio y otro la lógica de datos.



El objetivo de aumentar el número de niveles en una aplicación distribuida es lograr una mayor independencia entre un nivel y otro, lo que facilita la portabilidad en entornos heterogéneos y la escalabilidad en caso de incorporación de nuevos clientes.

Lenguajes de programación del lado del cliente.

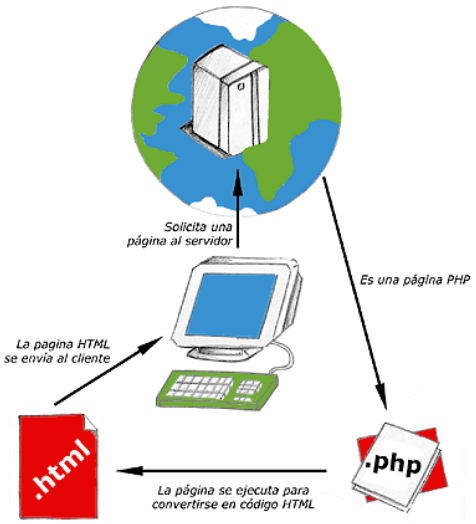
Estos programas residen junto a la página web en el servidor pero son transferidos al cliente para que este los ejecute.

Java, JavaScript, VBScript son lenguajes del lado del cliente.

Lenguajes de programación del lado del servidor.

Los lenguajes son ejecutados por el servidor y lo que se envía al cliente es la respuesta o el resultado de dicha ejecución.

Lenguajes como PHP o Perl pertenecen a esta categoría.



Ambientes para el desarrollo de aplicaciones Web.

Los IDE (ambientes integrados de desarrollo) para aplicaciones Web son muy numerosos. Algunos son específicos para lenguajes del lado del servidor. Por ejemplo, Visual Studio solo soporta ASP.NET del lado del servidor.

Existen IDE's en buena cantidad, libres y gratuitos de buena calidad. Algunos ejemplos de IDE para Web son:

- Microsoft Visual Studio.
- Microsoft Web Developer Express.
- Mono (para ASP.NET).
- NetBeans.
- Jbuilder.
- Eclipse.

Metodologías para el desarrollo de aplicaciones Web.

Las distintas metodologías se pueden dividir en tres generaciones, en base a su nivel de sofisticación, y en dos familias, las derivadas de modelos clásicos de datos (E/R) y las derivadas de modelos Orientados a Objetos (OMT y UML).

La primera generación: (primera mitad de los 90's). Sienta las bases de la Ingeniería Web al incluir conceptos como constructores de navegación, o promover la separación entre estructuras de navegación y el contenido durante el proceso de desarrollo.

La segunda generación: (segunda mitad de los 90's). Refina los primeros modelos e incluye conceptos como los soportes de funcionalidad básica, y los primeros esbozos de proceso donde se delimitan los modelos conceptual, lógico y físico.

La tercera generación: (2000-2002). Profundiza en el soporte para la funcionalidad, se enfatiza el artículo del usuario en los métodos, y se producen avances hacia la estandarización de notaciones, procesos y lenguajes textuales de especificación.

Debido al gran auge que ha tenido en el ámbito mundial el uso y navegación por Internet y la economía electrónica, los diseñadores de sitios Web y quienes los contratan para la construcción de sus portales, se han preocupado por las razones que implican que un usuario regrese o no al sitio; pues en el comercio electrónico, el usuario primero se enfrenta a la usabilidad y después hace el pago.

La evaluación o diagnóstico de usabilidad consiste en las metodologías que miden los aspectos de usabilidad de una interfaz utilizada por un sistema e identificar problemas específicos; siendo una parte importante del proceso total del diseño de la interfaz de usuario, que consiste en ciclos iterativos de diseñar, de prototipos y de la evaluación. No obstante la popularidad de UML, no se han contemplado la inclusión de características para el desarrollo en Web, tanto así que ha surgido una nueva rama de la ingeniería de software denominada "Ingeniería Web", en la cual se trata de cubrir los aspectos importantes de las aplicaciones enfocadas a la Web.

UWE (Ingeniería Web basada en UML)

La ingeniería Web basada en UML (UWE) fue presentada por Nora Koch en el 2000. Esta metodología utiliza un paradigma orientado a objetos, y está orientada al usuario. Está basada en los estándares UML y UP (Proceso Unificado), cubre todo el ciclo de vida de este tipo de aplicaciones centrando además su atención en aplicaciones personalizadas.

UWE propone una extensión de UML que se divide en 4 pasos:

1. Análisis de requisitos. Su objetivo es encontrar los requisitos funcionales de la aplicación Web para representarlos como casos de uso. Da lugar a un diagrama de casos de uso.

2. Diseño conceptual. Su objetivo es construir un modelo conceptual del dominio de la aplicación considerando los requisitos reflejados en los casos de uso. Da como resultado un diagrama de clases de dominio.

3. Diseño navegacional. Se obtienen el modelo de espacio de navegación y modelo de estructura de navegación, que muestra cómo navegar a través del

espacio de navegación. Se obtienen diagramas de clases que representan estos modelos.

4. Diseño de presentación. De este paso se obtienen una serie de vistas de interfaz de usuario que se presentan mediante diagramas de interacción UML. Los aspectos principales de esta metodología son:

- Uso de una notación estándar, como es la notación UML.
- Definición precisa del método, una serie de pasos para seguir la construcción de los modelos.

La especificación de restricciones, la metodología recomienda el uso de restricciones escritas en el Lenguaje de Restricciones de Objetos (OCL) para aumentar la precisión de los modelos.

WAE (Extensión de Aplicaciones Web para UML)

UML tiene definido un mecanismo para permitir cierto dominio para extender la semántica del modelo de elementos específicos, el mecanismo de extensión permite incluir nuevos atributos, diferente semántica y restricciones adicionales.

El conjunto de extensión de UML propuesto por Jim Conallen está formado por Valores etiquetados, estereotipos y restricciones (Tagged Values, Stereotypes and Constraints).

La parte del mecanismo de la extensión de UML es la habilidad de asignar iconos diferentes a las clases estereotipadas. El problema de una página Web es que tiene diferentes scripts y variables que se ejecutan en el servidor o del lado del cliente, este problema se puede resolver de dos formas:

El primero sería definir los estereotipos; método del servidor y método del cliente. En un objeto de la página un método que ejecuta en el servidor se estereotipará como «server method» y las funciones que corren en el cliente «client method». Esto resuelve el problema de distinguir los atributos y métodos de un objeto de la página, sin embargo todavía es un poco confuso. Una complicación extensa se levanta después cuando se hacen asociaciones a otros componentes en el modelo. No está claro que algunas de estas relaciones sólo son válidas en el contexto de los métodos y atributos del servidor o en los del cliente.

Metodologías Basadas en Hipermedia y Orientadas a Objetos.

¿Qué es Multimedia?

Multimedia: también denominada integración de medios digitales, consiste en un sistema que utiliza información almacenada o controlada digitalmente (texto, gráficos, animación, voz y vídeo) que se combinan en la computadora para formar una única presentación. La multimedia se pude definir, como una combinación de informaciones de naturaleza diversa, coordinada por la computadora y con la que el usuario puede interactuar. Se podrá emplear para realzar y optimizar el flujo de información, incrementando la eficacia de la comunicación entre el usuario final y la computadora.

La utilización de medios digitales de forma interactiva permitirá crear un entorno de comunicación más participativo, ya que combina información de diversos medios

en una única corriente de conocimiento, aumentando el impacto que se produciría en los usuarios si se emplea de manera separada, los medios digitales que se incluyen en una computadora son Animación, Gráficos, Sonido y Vídeo.

¿Qué es la Hipermedia?

La hipermedia es el resultado de combinación de hipertexto y la multimedia. Tradicionalmente, la idea de hipertexto se ha asociado con la documentación puramente textual, o en todo caso gráfico, por lo que la inclusión de otros tipos de información (vídeo, música, etc.) suele recogerse con el nombre de hipermedia. Por una parte, el hipertexto permite representar una estructura asociativa en la que nodos o conceptos pueden enlazarse automáticamente. Las aplicaciones multimedia permiten integrar diferentes medios bajo una presentación interactiva, para lo que pueden ofrecer dos tipos de accesos a la información:

- El mecanismo mayoritariamente utilizado, consiste en un control de usuario similar al de un reproductor de vídeo o sonido que avanza siguiendo el eje de coordenadas temporal.

- En algunos casos, se permite realizar saltos a un determinado instante dentro de una presentación, como suele hacerse en muchos reproductores de discos compactos. Esta facilidad tan sólo permite al usuario que, indicando el momento exacto en que debería aparecer un contenido, se pueda llevar a cabo un rápido movimiento hasta alcanzar dicho punto. A diferencia de los enlaces hipertextuales, estos saltos no dan lugar a ningún tipo de estructura concreta en la que navegar a través de la información.

Las ventajas de la hipermedia

En primer lugar, la hipermedia ofrece un medio adecuado para representar aquella información poco o nada estructurada que no puede ajustarse a los rígidos esquemas de las bases de datos tradicionales. Además, permite estructurar la información, jerárquicamente o no, de tal modo que también resulta útil en sistemas de documentación de textos tradicionales que poseen una marcada organización.

Esta tecnología, que se caracteriza por sus ergonómicos interfaces de usuario, muy intuitivos, pues imitan el funcionamiento de la memoria humana, hace que el usuario no tenga que realizar grandes esfuerzos para conseguir resultados rápidamente.

Estructura de los sistemas Hipermedia

Propiedades fundamentales en un sistema hipermedia hoy en día son:

1. El proceso de ligado entre dos ítems, dando una elevación al paradigma básico de nodo y liga.

2. La selección por asociación, dando una elevación al soporte de ligas que permite el recorrido de una red de nodos en una manera directa. Un sistema hipermedia esta hecho de nodos (conceptos) y ligas (relaciones). Los nodos están conectados a otros nodos por ligas. El nodo del cual una liga origina es llamado referencia y el nodo en el cual la liga termina es llamado referente. Ellos son también referidos como anclas. El contenido de un nodo es desplegado por ligas

activas. Aparte de esta arquitectura nodo-liga, los sistemas hipermedia tienen una estructura irregular.

Balasubramanian lista los componentes básicos de un sistema hipermedia como sigue:

1. Una interfaz de usuario gráfica con navegadores y diagramas generales para ayudar a los usuarios a navegar a través de una gran cantidad de información dispersa, posiblemente interconectada, unida por ligas activas.

2. Un sistema con herramientas para crear y manejar nodos y ligas.

3. Mecanismos de recuperación de información tradicional (tales como búsquedas por palabra clave, búsqueda por autor, etc.)

4. Una ingeniería hipermedia para manejar información acerca de nodos y ligas.

5. Un sistema de almacenaje el cual puede ser un sistema de archivos o una base de conocimiento o un tradicional DBMS.

Diseñando Sistemas Hipermedia

El proceso de desarrollo de un sistema hipermedia puede ser modelado a un cierto grado sobre los procesos de ingeniería de software. Por ejemplo, diseñar el modelo de datos conceptual y el modelo abstracto de navegación puede beneficiarse directamente de las propuestas de ingeniería de software. Las técnicas de diseño formal perfeccionan la consistencia de los documentos hipermedia y proveen líneas de guía. Sin embargo, para los procesos de ingeniería de software, no se requiere tomar en cuenta aspectos estéticos y cognoscitivos, los cuales son dos aspectos importantes del desarrollo hipermedia. Esto implica que las propuestas de ingeniería de software tradicional no son completamente adecuadas para el diseño de sistemas hipermedia. Nanard y Nanard sugiere que las aplicaciones hipermedia pueden ser mejor desarrolladas no siguiendo los pasos secuénciales de las metodologías formales tan correctamente.

SOHDM (Metodología de Diseño Hipermedia Orientado a Objetos y basada en escenarios)

Esta propuesta presenta la necesidad de disponer de un proceso que permita capturar las necesidades del sistema. Para ello, propone el uso de escenarios. El proceso de definición de requisitos parte de la realización de un diagrama de contexto tal y como se propone en diagrama de flujos de datos (DFD) de Yourdon. En este diagrama de contexto se identifican las entidades externas que se comunican con el sistema, así como los eventos que provocan esa comunicación. Las lista de eventos es una tabla que indica en qué eventos puede participar cada entidad. Por cada evento diferente SOHDM propone elaborar un escenario. Éstos son representados gráficamente mediante los denominados SACs (Scenario Activity Chart). Cada escenario describe el proceso de interacción entre el usuario y el sistema cuando se produce un evento determinado especificando el flujo de actividades, los objetos involucrados y las transacciones realizadas. SOHDM propone un proceso para conseguir a partir de estos escenarios el modelo conceptual del sistema que es representado mediante un diagrama de clases. El proceso de SOHDM continúa reagrupando estas clases para conseguir un modelo de clases navegacionales del sistema.

OOHDM (Método de Diseño Hipermedia Orientado a Objetos)

OOHDM propone el desarrollo de aplicaciones hipermedia a través de un proceso compuesto por cuatro etapas:

1. Diseño conceptual

2. Diseño navegacional

3. Diseño de interfaces abstractas

4. Implementación

Diseño conceptual: durante esta actividad se construye un esquema conceptual representado por los objetos del dominio, las relaciones y colaboraciones existentes establecidas entre ellos. En las aplicaciones hipermedia convencionales, cuyos componentes de hipermedia no son modificados durante la ejecución, se podría usar un modelo de datos semántica estructural (como el modelo E/R). De este modo, en los casos en que la información base pueda cambiar dinámicamente o se intente ejecutar cálculos complejos, se necesitará enriquecer el comportamiento del modelo de objetos.

En OOHDM el esquema conceptual está construido por clases, relaciones y subsistemas. Las clases son descritas como en los modelos orientados a objetos tradicionales. Sin embargo, los atributos pueden ser de múltiples tipos para representar perspectivas diferentes de las mismas entidades del mundo real. Se usa una notación similar a UML y tarjetas de clases y relaciones similares a las tarjetas CRC (Clase Responsabilidad Colaboración).

El esquema de las clases consiste en un conjunto de clases conectadas por relaciones.

Los objetos son instancias de las clases. Las clases son usadas durante el diseño navegacional para derivar nodos, y relaciones que son usadas para construir enlaces.

Diseño navegacional: en OOHDM la navegación es considerada un paso crítico en el diseño de aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual, admitiendo la construcción de modelos diferentes de acuerdo con los diferentes perfiles de usuarios. Cada modelo navegacional provee una vista subjetiva del diseño conceptual.

En OOHDM existen un conjunto de tipo predefinidos de clases navegacionales: nodos, enlaces y estructuras de acceso. La semántica de los nodos y los enlaces son las tradicionales aplicaciones hipermedia, y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos. Los nodos son enriquecidos con un conjunto de clases especiales que permiten de un nodo observar y presentar atributos (incluidos los links), así como métodos cuando se navega en un particular contexto.

Diseño de interfaz abstracta: en OOHDM se utiliza el diseño de interfaz abstracta para describir la interfaz de usuario de aplicación de hipermedia. El modelo de interfaz ADVs (Vista de Datos Abstracta) especifica la organización y comportamiento de la interfaz, pero la apariencia física real o de los atributos, y la disposición de las propiedades de las ADVs. en la pantalla real son hechas en fase de implementación.

Implementación: en esta fase, el diseñador debe implementar el diseño. Hasta ahora, todos los modelos fueron construidos en forma independiente de la plataforma de implementación; en esta fase es tenido en cuenta el entorno particular en el cual se va a correr la aplicación.

Al llegar a esta fase, el primer paso que debe realizar el diseñador es definir los ítems de información que son parte del dominio del problema. Debe identificar también, cómo son organizados los ítems de acuerdo con el perfil del usuario y su tarea; decidir qué interfaz debería ver y como debería comportarse. A fin de implementar todo en un entorno Web, el diseñador debe decidir además que información debe ser almacenada.

W2000

Supone una propuesta que amplía la notación UML con conceptos para modelar elementos de multimedia heredados de la propuesta HDM (Método de Diseño Hipermedia). El proceso de desarrollo de W2000 se divide en tres etapas:

1. Análisis de requisitos.

2. Diseño de hipermedia.

3. Diseño funcional.

La especificación de requisitos en W2000 se divide en dos subactividades: análisis de requisitos funcionales y análisis de requisitos navegacionales. La especificación de requisitos comienza haciendo un estudio de los diferentes roles de usuario que van a interactuar con el sistema. Cada actor potencialmente distinto tendrá su modelo de requisitos de navegación y de requisitos funcionales.

El modelo de requisitos funcionales es representado como un modelo de casos de uso tal y como se propone en UML. En él se representa la funcionalidad principal asociada a cada rol y las interacciones que se producen entre el sistema y cada rol. El segundo modelo consiste en otro diagrama de casos de uso pero que no representa funcionalidad sino posibilidades de navegación de cada actor. La representación gráfica es realizada con una extensión de UML.

EORM (Metodología de Relaciones de Objetos Mejorada)

En esta metodología, propuesta por D. Lange, el proceso de desarrollo de un Sistema de Información Hipermedial (Hiperdocumento) comprendería una primera fase de Análisis Orientada a Objetos del sistema, sin considerar los aspectos hipermediales del mismo, obteniendo un Modelo de Objetos con la misma notación utilizada en OMT, que refleje la estructura de la información (mediante clases de objetos con atributos y relaciones entre las clases) y el comportamiento del sistema (a través de los métodos asociados a las clases de objetos).

La idea fundamental de esta metodología es considerar una segunda fase, de Diseño, durante la cual se proceda a modificar el modelo de objetos obtenido durante el análisis añadiendo la semántica apropiada a las relaciones entre las de objetos para convertirlas en enlaces hipermedia, obteniendo finalmente un modelo enriquecido, que su autor denomina EORM (Enhanced Object-Relationship Model), en el que se refleje tanto la estructura de la información (modelo abstracto hipermedial compuesto de nodos y enlaces) como las posibilidades de navegación ofrecidas por el sistema. Sobre dicha estructura, para lo cual existirá un repositorio o librería de clases de enlaces, donde se especifican las posibles operaciones asociadas a cada enlace de un hiperdocumento, que serán de tipo crear, eliminar, atravesar, siguiente, previo etc., así como sus posibles atributos (fecha de creación del enlace, estilo de presentación en pantalla, restricciones de acceso, etc.).

El desarrollo del Sistema concluiría con una última fase de Construcción, durante la que se generaría el código fuente (por ejemplo en C++) correspondiente a cada clase, y se prepararía la Interfaz Gráfica de Usuario. La adopción del enfoque orientado a objetos OMT garantiza todas las ventajas reconocidas para esta técnica de modelado, como la flexibilidad (posible existencia de múltiples formas de relaciones entre nodos) y la reutilización, por la existencia de una librería de clases de enlaces que pueden ser reutilizados en diferentes proyectos de desarrollo hipermedial.

Para automatizar la aplicación de la metodología EORM, su autor ha desarrollado, en los laboratorios de investigación de IBM, una herramienta denominada ODMTool que, junto a un generador comercial de Interfaces Gráficas de Usuario denominado ONTOS Studio y un Sistema de Gestión de Base de Datos Orientado a Objetos (SGBDOO), permite el diseño interactivo de esquemas EORM y la generación de código fuente, inicialmente en C++, de las clases incluidas en estos esquemas. El SGBDOO ofrece un repositorio de objetos que permite compartir la información de los esquemas entre las herramientas (ODMTool, ONTOS Studio) y las aplicaciones hipermediales desarrolladas.

Aspectos de seguridad.

Introducción a la Seguridad en Sistemas Web

Un efecto secundario del crecimiento exponencial que ha tenido el Internet es la privacidad de información tanto personal como profesional. En Internet encontramos funcionando a tiendas en línea, negocios que mueven grandes cantidades de dinero, redes de los servicios que habilitan el comercio a nivel internacional así como sitios de redes sociales que contienen información muy delicada de la vida privada de sus miembros.

Mientras más se conecta el mundo, la necesidad de seguridad en los procedimientos usados para compartir la información se vuelve más importante. Desde muchos puntos de vista, podemos creer sin dudar que el punto más crítico de la seguridad del Internet, lo tienen las piezas que intervienen de forma directa con las masas de usuarios, los servidores web.

Respecto a los servidores web, es común escuchar sobre fallas en los sistemas de protección de los servidores más frecuentemente utilizados (Apache, IIS, etc.), o en los lenguajes de programación en los que son escritas las aplicaciones que son ejecutadas por estos servidores. Pero es un hecho, que la mayoría de los problemas detectados en servicios web no son provocados por fallas intrínsecas

de ninguna de estas partes, ya que una gran cantidad de los problemas se generan por malos usos por parte de los programadores.

Ahora que sabemos que la mayoría de los problemas de seguridad en los sitios web se encuentran a nivel aplicación y que son el resultado de escritura defectuosa de código, debemos entender que programar aplicaciones web seguras no es una tarea fácil, ya que requiere por parte del programador, no únicamente mostrar atención en cumplir con el objetivo funcional básico de la aplicación, sino una concepción general de los riesgos que puede correr la información contenida, solicitada y recibida por el sistema. En la actualidad, aunque existen muchas publicaciones que permiten formar un criterio sobre el tema, no existen acuerdos básicos sobre lo que se debe o no se debe hacer, y lo que en algunas publicaciones se recomienda, en otras es atacado.

Problemas principales en la Programación de Sistemas Web

Una gran parte de los problemas de seguridad en las aplicaciones web son causados por la falta de seguimiento por parte del programador en los siguientes aspectos:

- Entradas al sistema
- Salidas del sistema

Quizás uno de los consejos de seguridad en PHP más conocido en cualquier foro de Internet es el uso del parámetro register_globals que es considerado a priori por muchos administradores como un defecto en la configuración y muy probablemente sin entender con cabalidad que es lo que implica esta configuración. El tener habilitado este parámetro oculta el origen de los datos. Si se encuentra habilitado, no podemos saber cómo es que una variable entró al sistema (si lo hizo por medio de una petición GET o POST por ejemplo) y contribuye a la pérdida del control por parte del programador sobre los procesos a los que se ha sometido cada variable para librarla de riesgos potenciales para la aplicación.

Otro aspecto importante además de los procesos de verificación que se deben de tener para con las entradas y salidas del sistema lo representa la fuga de información útil para un posible ataque sobre nuestro sistema. En este punto, los mensajes de error enviados por el servidor, que suelen ser de gran utilidad durante el proceso de desarrollo de la aplicación, se vuelven contra nosotros cuando siguen apareciendo en una aplicación que se encuentra en la etapa de producción, por lo que es necesario deshabitar todos estos mensajes y editar algunos otros (como los que se envían cuando el servidor no encuentra algún archivo en particular) que también pueden ser utilizados por los atacantes para obtener información sobre nuestro sistema.

Practicas básicas de Seguridad Web

Balancear Riesgo y Usabilidad: La recomendación inicial sería tratar de usar medidas de seguridad que sean transparentes a los usuarios. Por ejemplo, la solicitud de un nombre de usuario y una contraseña para registrarse en un sistema son procedimientos esperados y lógicos por parte del usuario.

Rastrear el paso de los Datos: Particularmente para *PHP* existen arreglos superglobales como *$_GET*, *$_POST* y *$_COOKIE* entre otros que sirven para

identificar de forma clara las entradas enviadas por el usuario. Si esto lo combinamos con una convención estricta para el nombrado de las variables podemos así tener un control sobre el origen de los datos usados en el código.

Además de entender los orígenes de la información, tiene también igual importancia entender cuáles son las salidas que tiene ésta de la aplicación.

Filtrar Entradas: El filtrado es una de las piedras angulares de la seguridad en aplicaciones web. Es el proceso por el cual se prueba la validez de los datos. Si nos aseguramos que los datos son filtrados apropiadamente al entrar, podemos eliminar el riesgo de que datos contaminados y que reciben confianza indebida sean usados para provocar funcionamientos no deseados en la aplicación.

Escapar salidas: El proceso de escapado debe estar compuesto a su vez por los siguientes pasos: Identificar las salidas, escapar las salidas y distinguir entre datos escapados y no escapados. Para escapar las salidas, primero debemos identificarlas. En ***PHP*** una forma de identificar salidas hacia el cliente es buscar por líneas como: echo, print, printf, <?=

Clasificación de Ataques

Ataques URL de tipo Semántico: Este tipo de ataques involucran a un usuario modificando la URL a modo de descubrir acciones a realizar originalmente no planeadas para él. Los parámetros que son enviados directamente desde la URL son enviados con el método GET y aunque los parámetros que son enviados con este método sólo son un poco más fáciles de modificar que los enviados en forma oculta al usuario en el navegador, esta exposición adicional de los parámetros tiene consecuencias, como cuando queda registrada la URL con todo y estos parámetros quizás privados en buscadores como Google.

Ataques al subir archivos: Existen algunos ataques que aprovechan la posibilidad de la aplicación de subir archivos al servidor. Generalmente PHP almacena los archivos subidos en un carpeta temporal, sin embargo es común en las aplicaciones cambiar la localización del archivo subido a una carpeta permanente y leerlo en la memoria. Al hacer este tipo de procedimientos debemos revisar el parámetro que hará referencia al nombre del archivo, ya que puede ser truqueado a modo de apuntar a archivos de configuración del sistema (como ***/etc/passwd*** en sistemas Unix).

Ataques de Cross−Site Scripting: XSS es un tipo de vulnerabilidad de seguridad informática típicamente encontrada en aplicaciones web que permiten la inyección de código por usuarios maliciosos en páginas web vistas por otros usuarios. Los atacantes típicamente se valen de código HTML y de scripts ejecutados en el cliente. Desde la liberación del lenguaje JavaScript, se previeron los riesgos de permitir a un servidor Web enviar código ejecutable al navegador. Un problema se presenta cuando los usuarios tienen abiertos varias ventanas de navegador, en algunos casos un script de una página podría acceder datos en otra página u objeto, observando el peligro de que un sitio malicioso intentara acceder datos sensibles de esta forma. Por ello se introdujo la política same−origin. Esencialmente esta política permite la interacción entre objetos y páginas, mientras estos objetos provengan del mismo dominio y en el mismo protocolo. Evitando así que un sitio malicioso tenga acceso a datos sensibles en otra ventana del navegador vía JavaScript.

Cross−Site Request Forgeries: Este tipo de ataque permite al atacante enviar peticiones HTTP a voluntad desde la máquina de la víctima. Por la naturaleza de este tipo de ataques, es difícil determinar cuándo una petición HTML se ha originado por un ataque de este tipo.

Envío de Formas falsificadas: Falsificar una forma es casi tan fácil como manipular una URL. Un atacante podría copiar el código fuente de una página, salvarla en su equipo, y modificar el atributo de la acción que realizará la forma incluyendo ahora la ruta absoluta de la página originalmente deseada. Ahora el atacante puede quitar restricciones originales que se hubieran ejecutado en el cliente, como el tamaño máximo de un archivo adjunto, desactivar la validación de datos en el lado del cliente, alterar los elementos ocultos o los tipos de datos de los elementos de la forma. Operando de esta forma podemos enviar datos arbitrarios al servidor, de una manera sencilla y sin el uso de herramientas sofisticadas. Este tipo de falsificaciones es algo que no podemos prevenir, pero es algo que debemos tomar en cuenta. Mientras tengamos el poder de filtrar las entradas, los usuarios deben jugar con nuestras reglas.

Peticiones HTTP Falsificadas: Un ataque más sofisticado que el anterior es enviar peticiones falsas empleando herramientas especiales para este propósito. La existencia de este tipo de ataques es una prueba determinante de que los datos enviados por los usuarios no son dignos de ninguna confianza.

Los dos últimos tipos de ataques mencionados son una muestra de que el usuario no está obligado a enviar la información siguiendo las reglas que han sido definidas en la aplicación. En realidad un atacante puede confeccionar a gusto sus peticiones HTTP, la fortaleza de nuestro sistema será medible por su capacidad de detectar que peticiones recibidas deben ser escuchadas y procesadas de acuerdo a los parámetros y valores de éstos que son esperables recibir.

Como conclusión general podemos decir que la seguridad en aplicaciones Web involucra principalmente al desarrollador, aunque con gran frecuencia se encuentran defectos que pueden ser aprovechados por atacantes en las tecnologías en que se basan los sistemas web (Sistemas Operativos, Servidores Web, Servidor Base de Datos, etc.) la atención principal debe dirigirse a los defectos propios al desarrollo nuestras aplicaciones. A menudo, los desarrolladores desconocen a detalle el funcionamiento de los sistemas web y no consideran todas las posibilidades de uso o mal uso al que un sistema web puede someterse cuando se conoce con mayor detalle el protocolo HTTP y las herramientas que permiten aprovecharlo de otra manera, es decir, los programadores con frecuencia desconocen que las aplicaciones pueden ser accedidas con herramientas diferentes al puro navegador web, o incluso la existencia de aditamentos a los navegadores que potencializan su uso de manera diferente al navegador común.

Entendiendo lo anterior, todo programador debe estar consciente de que de él depende el rechazar o filtrar las peticiones recibidas en que los datos o variables recibidas no cumplan con las características esperadas o predefinidas. Ninguna entrada al sistema debe ser digna de una confianza plena, todas de preferencia deben pasar por el filtrado de los datos contenidos para confirmar su usabilidad.

Otro aspecto importante a considerar son los procesos de salida de la información del sistema. Es importante siempre considerar el significado que pueda tener la información enviada en su nuevo contexto, y en el caso de poder crear problemas de interpretación de las salidas escapar las salidas para preservarlas. Al igual que

en el proceso de filtrado, es importante mantener un control sobre la codificación que tienen los datos antes de enviarlos a su nuevo contexto.

Otro tipo de problemas, como los procesos de autenticación tienen consideraciones propias que debemos implementar para hacer más robusta y confiable nuestra aplicación.[2]

[2] *Aspectos Básicos de la Seguridad en Aplicaciones Web.*
http://www.seguridad.unam.mx