Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Diseño Orientado a Objetos

Erick Alejandro Campos Rivero

1506449

Cliente- servidor, seguridad de aplicación.

Prof.: Miguel Ángel Salazar Santillán

¿Qué es una aplicación web?

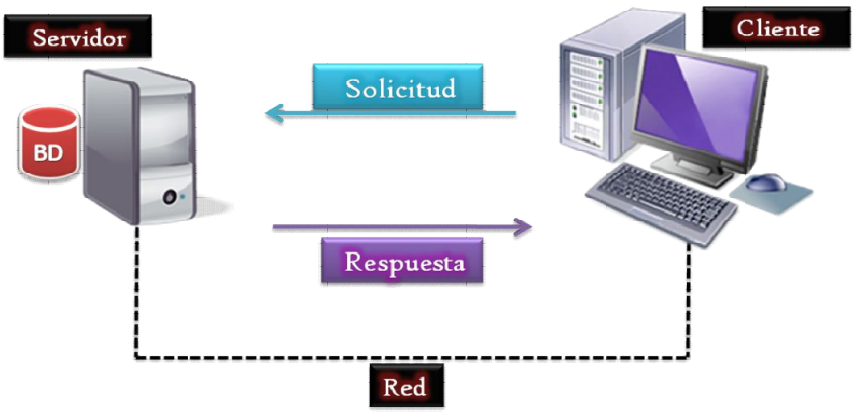
Las aplicaciones web reciben este nombre porque se ejecutan en la internet. Es decir que los datos o los archivos en los que trabajas son procesados y almacenados dentro de la web. Estas aplicaciones, por lo general, no necesitan ser instaladas en tu computador.

El concepto de aplicaciones web está relacionado con el almacenamiento en la nube. Toda la información se guarda de forma permanente en grandes servidores de internet y nos envian a nuestros dispositivos o equipos los datos que requerimos en ese momento, quedando una copia temporal dentro de nuestro equipo.

Características:

- Aplicaciones cliente/servidor que utilizan el protocolo HTTP para interactuar con los usuarios u otros sistemas
- El cliente utilizado por los usuarios es habitualmente un navegador
- Los problemas de seguridad pueden provenir de los programas web en los que se apoyan, aunque en su mayor parte son consecuencia de fallos en la lógica y el diseño de la propia aplicación

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta.



En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

¿Qué es el cliente?

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término front-end.
El Cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de una red.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

¿Qué es el servidor?

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término back-end [15].
El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

El desarrollo de una aplicación web requiere de una serie de **herramientas**: servidores web, servidores de aplicaciones, servidores de bases de datos, lenguajes de servidor, etc.

Estas herramientas pueden plantear **problemas** que comprometan a la aplicación:

- Vulnerabilidades debidas al uso de versiones no actualizadas
- Configuraciones por defecto inadecuadas
- Activación de cuentas por defecto

Las herramientas deben estar **actualizadas y bien configuradas** para impedir ataques a la aplicación.

Seguridad en la Aplicación

Control de acceso

Un aspecto muy importante de una aplicación web es el control de acceso de los usuarios a zonas restringidas de la aplicación. Aquí intervienen dos conceptos:

- **Autentificación**
- **Autorización**

Autentificación

Es el proceso de determinar si un usuario es quien dice ser Esto se puede hacer de varias maneras. Algunas de ellas son:

- **Autentificación HTTP básica**
- **Autentificación basada en la aplicación**

Autentificación HTTP básica

- **Ventajas**:
  - Es muy simple de implementar
  - Se pueden fijar restricciones de acceso por usuario y contraseña o por otros conceptos como por ejemplo el dominio o dirección IP de la máquina
- **Inconvenientes**:
  - Los datos viajan por la red sin encriptar
  - No se puede hacer logout, la única forma es cerrar el navegador
  - No hay control sobre el diseño de la ventana de diálogo.

Autentificación basada en la aplicación

- La propia aplicación puede implementar un mecanismo de autentificación que implica la presentación de un formulario para que el usuario introduzca sus credenciales y el uso de una base de datos para verificar la corrección de éstas
- Es **más costosa pero más flexible** ya que permite establecer diferentes permisos y niveles de acceso en función del usuario que solicita la autentificación

- **Passwords**

  Siempre que se utilizan passwords para autentificar usuarios hay que seguir unas **recomendaciones**:

  - Restringir los valores para los nombres de usuarios. Los que representan nombres reales suponen dar pistas a los atacantes.
  - Almacenar los passwords de forma segura, protegiendo el acceso a la base de datos.
  - Seguir reglas de seguridad para su elección.
  - Bloquear una cuenta cuando se detecta un número determinado de intentos de acceso incorrectos.
  - Actualizar los passwords periódicamente y mantener un histórico para evitar repeticiones.

- **Sesiones**

- Una vez que el usuario se ha autentificado introduciendo su nombre de usuario y su clave, es preciso **mantener esta autentificación** en cada conexión subsiguiente
- Para evitar tener que mostrar nuevamente la ventana de autentificación se recurre habitualmente al uso de sesiones, un mecanismo que permite mantener el estado entre diferentes peticiones HTTP
- Una vez autentificado, al usuario se le asigna un identificador de sesión
- Este identificador acompañará invisiblemente a cada petición del usuario, con lo cual se garantizará que la petición proviene de un usuario previamente autentificado
- El identificador de sesión se suele almacenar en la propia máquina del cliente, mediante una *cookie*
- Sólo se debe almacenar el identificador de la sesión; cualquier otro dato del usuario se almacenará en el servidor

-

La **gestión de las sesiones** es responsabilidad del programador. Normalmente los lenguajes de servidor disponen de funciones diseñadas específicamente para ello.

Un buen sistema de gestión de sesiones debe:

- Establecer un tiempo límite de vida para la sesión.
- Regenerar el identificador de sesión cada cierto tiempo.
- Detectar intentos de ataque de fuerza bruta con identificadores de sesión.
- Requerir una nueva autentificación del usuario cuando vaya a realizar una operación importante.
- Proteger los identificadores de sesión durante su transmisión.
- Destruir la *cookie* al finalizar la sesión para evitar el acceso de otro usuario en un entorno público.

- **Autorización**

- Es el acto de comprobar si un usuario tiene el permiso adecuado para acceder a un cierto fichero o realizar una determinada acción, una vez que ha sido autentificado

- **Modelos para el control de acceso:**

- **Control de Acceso Discrecional**: se basa en la identidad de los usuarios o su pertenencia a ciertos grupos. El propietario de una información puede cambiar sus permisos a su discreción
- **Control de Acceso Obligatorio**: cada pieza de información tiene un nivel de seguridad y cada usuario un nivel de acceso, lo cual permite determinar los permisos de acceso de cada usuario a cada pieza de información
- **Control de Acceso Basado en Roles**: cada usuario tiene un rol dentro de la organización y en función de él unos permisos de acceso

- **Validación de datos de entrada**

- El problema más frecuente que presentan las aplicaciones web es **no validar correctamente los datos de entrada**
- Esto da lugar a algunas de las vulnerabilidades más importantes de las aplicaciones, como la Inyección SQL, el *Cross-Site Scripting* y el *Buffer Overflow*

- **Fuentes de entrada – cadenas URL**

- La aplicación debe chequear el valor recibido **aunque proceda de una lista desplegable** con unos valores predefinidos, ya que el usuario ha podido modificar manualmente la URL
- Este problema se da también en los hipervínculos que incluyen parámetros
- Siempre que se envíen datos sensibles hay que acompañarlos de un identificador de sesión y comprobar que el usuario asociado a la sesión tiene acceso a la información requerida

- **Fuentes de entrada - *cookies***

- Es un método habitual de mantener el estado o almacenar preferencias del usuario.
- Pueden ser **modificadas por el cliente** para engañar al servidor. El peligro dependerá de lo que se almacene en la cookie. Por ejemplo, la *cookie*
  Cookie: lang=en-us; ADMIN=no; y=1; time=10:30GMT;
  puede ser modificada fácilmente por:
  Cookie: lang=en-us; ADMIN=yes; y=1; time=12:30GMT;
- Lo mejor es **almacenar en la *cookie* únicamente el identificador de sesión**, manteniendo la información relevante en el servidor.

-
    - **Fuentes de entrada - formularios**
    - Los formularios pueden ser modificados para enviar lo que el usuario desee. Basta con guardar la página, modificar el código y recargarlo en el navegador. **Las limitaciones impuestas en el propio formulario se pueden saltar** perfectamente. Ejemplo:

<input type=”text” name="titulo" maxlength="100">

- Este elemento podría modificarse así:

<input type=”text” name="titulo" maxlength="100000">

- con el consiguiente riesgo para la aplicación si el valor no se chequea adecuadamente
    - **Gestión de errores**
- Los mensajes de error son una fuente de información muy importante para los atacantes. Pueden proporcionar información sensible que les permita refinar sus ataques. En un entorno de producción debe evitarse la aparición de mensajes de aviso o error.
    - **Protección de información**
- Toda información sensible debe almacenarse por separado del programa que la utiliza y preferentemente en un directorio situado fuera del árbol de directorios de la Web para evitar que pueda ser accedida por su URL
    - **Protección de información**
- Debe evitarse utilizar en el código comentarios que den demasiados detalles acerca del funcionamiento del programa. Puede ser conveniente eliminarlos en la versión de producción de la aplicación
- Hay que suprimir las órdenes de depuración colocadas en el código durante su desarrollo
- Deben protegerse los ficheros que tengan el acceso restringido. Para ello no basta con suprimir los enlaces al fichero; alguien podría dar con su nombre y obtenerlo directamente escribiendo su URL.

Así existen aún más formas de asegurar la seguridad de una aplicación, pero estas son tan solo algunas de las sugerencias y de las que hemos visto en clase.