- Nunca se debe confiar en los datos introducidos por los usuarios
- Hay que garantizar la robustez de las aplicaciones
- Existen vulnerabilidades asociadas a los datos introducidos por los usuarios:
  - Vulnerabilidad de la integridad de los datos
  - Violación del formato de los datos
  - Incumplimiento de las reglas de negocio

Controles de Validación en Aplicaciones Web

- ¿Cómo evitar las amenazas asociadas a las vulnerabilidades?
  - Aplicando técnicas de codificación que mejoren la seguridad de las aplicaciones
  - Evitar los ataques por inyección con la validación de entradas

Controles de Validación en Aplicaciones Web (2)

- Permite avisarle al usuario que los datos que está ingresando no cumplen con reglas previamente definidas para los campos
- Corresponden a un control preventivo y detectivo
- Mejora la satisfacción del usuario con la aplicación y ayuda a reducir la carga de la aplicación en el servidor
- La validación puede contemplar tres aspectos:
  - Uso de expresiones regulares
  - Uso de campos ocultos
  - Uso de campos requeridos

Validación del lado del cliente

- Expresiones regulares
  - Son modelos que describen las combinaciones de caracteres en el texto
  - Se podrían definir como una serie de caracteres que forman un patrón
  - Pueden utilizarse en múltiples lenguajes de programación
  - Ejemplos

```
function objeto() {
  var m = document.getElementById("matricula").value;
  var expreg = new RegExp("^[A-Z]{1,2}\\s\\d{4}\\s([B-D]|[F-H]|[J-N]|[P-T]|[V-Z]){3}$");

  if(expreg.test(m))
    alert("La matrícula es correcta");
  else
    alert("La matrícula NO es correcta");
}
```

Validación del lado del cliente (2)

Carácter	Texto buscado
^	Principio de entrada o línea.
$	Fin de entrada o línea.
*	El carácter anterior 0 o más veces.
+	El carácter anterior 1 o más veces.
?	El carácter anterior una vez como máximo (es decir, indica que el carácter anterior es opcional).
.	Cualquier carácter individual, salvo el de salto de línea.
x|y	x o y.
{n}	Exactamente n apariciones del carácter anterior.
{n,m}	Como mínimo n y como máximo m apariciones del carácter anterior.
[abc]	Cualquiera de los caracteres entre corchetes. Especifique un rango de caracteres con un guión (por ejemplo, [a-f] es equivalente a [abcdef]).
[^abc]	Cualquier carácter que no esté entre corchetes. Especifique un rango de caracteres con un guión (por ejemplo, [^a-f] es equivalente a [^abcdef]).
\b	Límite de palabra (como un espacio o un retorno de carro).
\B	Cualquiera que no sea un límite de palabra.
\d	Cualquier carácter de dígito. Equivalente a [0-9].

Validación del lado del cliente (3)

Carácter	Texto buscado
\D	Cualquier carácter que no sea de dígito. Equivalente a [^0-9].
\f	Salto de página.
\n	Salto de línea.
\r	Retorno de carro.
\s	Cualquier carácter individual de espacio en blanco (espacios, tabulaciones, saltos de página o saltos de línea).
\S	Cualquier carácter individual que no sea un espacio en blanco.
\t	Tabulación.
\w	Cualquier carácter alfanumérico, incluido el de subrayado. Equivalente a [A-Za-z0-9_].
\W	Cualquier carácter que no sea alfanumérico. Equivalente a [^A-Za-z0-9_].

Guía y pruebas http://regexpre.orgfree.com/

Validación del lado del cliente (4)

Validación del lado del cliente (5)

- Campos ocultos
  - Útiles para realizar comprobaciones pero no se deben utilizar para almacenar información sensible
  - El uso de campos ocultos para almacenar información sensible expone el funcionamiento interno de la aplicación así como los datos
  - En caso de utilizarlos es recomendable cifrar el contenido o buscar la manera de que no sean accesibles a los usuarios
  - Ejemplo:
    - Uso de identificadores en campos ocultos
    - Uso de tokens en CSRF

Validación del lado del cliente (6)

- Campos requeridos
  - Se trata de forzar al usuario a introducir un valor en un cuadro de texto en los que sea obligatorio.
  - Para que se de por completado un campo de texto obligatorio, se debe comprobar además que el contenido del campo sea válido
    - Ejemplo: que no lleve espacios en blanco
  - Ejemplo

```
<form action="" method="" id="" name="" onsubmit="return validacion()">
  ...
</form>
```

- Campos requeridos
  - Ejemplo

```
function validacion() {
  if (condicion que debe cumplir el primer campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  else if (condicion que debe cumplir el segundo campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  ...
  else if (condicion que debe cumplir el último campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }

  // Si el script ha llegado a este punto, todas las condiciones
  // se han cumplido, por lo que se devuelve el valor true
  return true;
}
```

Validación del lado del cliente (7)

- Son más seguras pues los datos ya se han transmitido
- Funcionan correctamente con todo tipo de navegadores
- Al utilizarlas se garantiza el buen funcionamiento de la aplicación y evitar los errores
- Se evita la introducción de código malicioso
- Evita revelar información confidencial
- PHP cuenta con funciones para realizar las validaciones
  - Ejemplo:
    - preg_match → expresiones regulares
    - isset() → comprobar que una variable se haya definido

Validación del lado del servidor

- Pasos comunes:
  1. Comprobar que todas las variables $_POST o $_GET existen
  2. Comprobar que cada variable concuerda con su expresión regular.
  3. Si es necesario, comprobar el tipo de la variable (string, int, float…)
  4. En el caso de que se requiera una doble contraseña (habitual en formularios de registro), comprobar que nos han llegado las dos y que son iguales

Validación del lado del servidor (2)

- Ejemplo

```
if( isset($_POST['email']) && isset($_POST['password1']) && isset($_POST['password2']) ){
  $email = $_POST['email'];
  $password1 = $_POST['password1'];
  $password2 = $_POST['password2'];

  if( !preg_match(,$email) ){
    exit('Error: se ha recibido un email inválido');
  }else if( !preg_match(,$password1) ){
    exit('Error: la contraseña debe tener entre 6 y 25 caracteres, incluyendo al menos una letra mayúscula, una minúscula, un número
y un caracter especial');
  }else if( $password1 != $password2 ){
    exit('Error: las contraseñas introducidas no coinciden');
  }

  //Consultas a la base de datos

}else{
  exit('Error: no se han recibido todas las variables necesarias');
}
```

Validación del lado del servidor (3)

- Manejo de errores
  - La directiva error_reporting determina que niveles de errores son reportados por PHP (http://php.net/manual/es/errorfunc.constants.php)
  - Los directivas display_errors y log_errors se pueden utilizar para determinar cómo se informará de los errores.
    - display_errors = yes → errores se generan a la salida del script.
    - log_errors = yes → errores se escriban en el registro de errores del servidor web.

Validación del lado del servidor (2)

- Manejo de errores
  - En PHP se puede realizar definiendo una función que se encargue de manejar los errores que ocurran en la aplicación
  - Ejemplo:
    - http://michelletorres.mx/manejo-de-errores-en-php/

Validación del lado del servidor (3)

Enunciado práctica/tarea semana 4

- Implemente un formulario web con al menos los siguiente campos:
  - Campos
    - Nombre de usuario: <<nombre.apellido>>
    - Contraseña: <<mayusculas, minusculas, símbolos, numeros>>
    - Email: <<formato contraseña>>
    - Fecha de nacimiento: <<dd/mm/yyyy>>
    - Campo oculto: <<verificar que no esté lleno>>
  - Implemente validaciones del lado del cliente y del lado del servidor
- Cree una página web con la funcionalidad de al menos 4 controles web.

Enunciado práctica / semana 4