PHP (Hipertext preprocessor)

Para poder realizar proyectos PHP se necesitan tres tecnologías de soporte:

- Apache (Servidor páginas web).
- MySQL (Gestor de base de datos).
- Intérprete de PHP (Instalado en el servidor Apache)

HTTP (Hypertext Transfer Protocol – Protocolo de transferencia de hipertexto):

HTTP es un protocolo de solicitud y respuesta a través de TCP (Protocolo de control de transmisión, que permite la seguridad de transmisión de datos de forma segura).

Es el protocolo que se usa para la tranferencia/transporte de datos a Internet.

- Fue creado por el Consorcio de la World Wide Web.

Apache (Servidor de páginas web).

- Servidor HTTP más utilizado.
- Lidera el mercado en un 66%.
- Es un servidor configurable.
- Es gratis y de código abierto (se pueden implementar mejoras desarrolladas por la comunidad).
- Puede trabajar con gran cantidad de lenguajes de programación (PHP, Perl…).
- Fue diseñado para sistemas Unix (como Linux).
- Desde su versión 2.0 puede trabajar con Windows.

MySQL (Gestor de base de datos).

Gestor de base de datos relacionales rápido y sólido.

- Una base de datos estructural guarda los datos en tablas separadas en vez de guardarlo todo en un almacén. Con esto se consigue más velocidad y flexibilidad.

Con MySQL se pueden hacer búsquedas, ordenación y control el acceso simultaneo de usuarios en la base de datos. Pudiendo restringir la entrada de los usuarios que se deseen.

- Usa SQL (Lenguaje de consulta/Query estructurado), lenguaje de consulta de base de datos más utilizado.
- Se creó en 1986 por IBM.

Características de MySQL:

- Escalable (Puede trabajar con base de datos sencillas de 1Mb hasta con bases de datos de varios TB).
- Alto rendimiento (Puede trabajar con webs que tienen millones de consultas diarias).
- Protección de usuario (Mecanismo para autorizar acceso del usuario, posibilidad de backup y recuperación de datos).
- Posibilidad de desarrollar con distintos lenguajes de programación.

---

*píldoras informáticas:

MySQL es un gestor de BBDD relacional, multihilo y multiusuario.

- **Relacional:** Suele tener varias tablas y la información está relacionada entre sí.

- **Multihilo**: Soporta varios procesos a la vez (se pueden hacer varias peticiones simultáneamente).

- **Multiusuario**: Puede haber varios usuarios accediendo y modificando la bbdd.

Base de datos:

PHP (PHP Hipertext preprocessor).

“Hipertext: Es una herramienta que, pese a su nombre, no se limita a información de los documentos Hipertext solo pueden abrirse y leerse con un navegador web”

El hipertexto, a pesar de su nombre, no se limita a información escrita o textual, sino que puede enlazar con imágenes, sonidos, documentos audiovisuales, páginas Web enteras o cualquier otra forma de acción digital (enviar un e-mail, descargar un archivo, etc.). A la convivencia de este tipo de formatos se la suele denominar hipermedia.

Los documentos de hipertexto deben abrirse y leerse con un navegador o browser (también “visualizador” o “cliente”), que es un software informático diseñado para ello, y a la acción de saltar de un hipertexto a otros disponibles en la Web se la denomina navegar en Internet.

Funciones PHP para conexión SQL:

- **Conectar con base de datos (utilizando PROCEDIMIENTOS):**
  - **msqli_connect()**

```php
$db_host= "localhost"; //Dirección base de datos
$db_nombre= "pruebas"; //Nombre base de datos
$db_usuarios= "root"; //Usuario para acceder a la bbdd
$db_contra= ""; //Contraseña para acceder a la bbdd

$conexion= mysqli_connect($db_host, $db_usuarios, $db_contra, $db_nombre);
```

- **Guardar datos generados por una consulta (también llamado RESULSET):**
  Se genera una tabla (que no vemos) con todos los datos que se han generado cuando hemos hecho una consulta a la base de datos.
  - **mysqli_query()**

```php
$conexion=mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);

$consulta="SELECT * FROM DATOSPERSONALES";

$resultados=mysqli_query($conexion, $consulta);
```

- **Mirar información dentro del RESULSET (mysqli_query) con array <u>indexada</u>:**
  Mira fila a fila lo que hay almacenado en la tabla virtual generada con el resulset y lo guarda en un array.
  - **mysqli_fetch_row()**

```php
$fila=mysql_fetch_row($resultados);

echo $fila[0];
```

- **Mirar información dentro del RESULSET (mysqli_query) con array <u>asociativa</u>:**
  Mira fila a fila lo que hay almacenado en la tabla virtual generada con el resulset y lo guarda en un array.
  - **mysqli_fetch_array()**

```php
$fila=mysqli_fetch_array($resultados, MYSQL_ASSOC);
echo $fila['NOMBREARTÍCULO'];
```

- **Generar mensaje de error si no se produce la conexión a la base de datos** *(incluir justo después de la llamada a conexión)*:
  - **mysqli_connect_errno()**

```php
if(mysqli_connect_errno()){
    echo "Fallo al conectar con la BBDD";
    exit();
}
```

- **Incluir caracteres latinos** *(incluir justo después de la llamada a conexión):*
  - **mysqli_set_charset()**

```php
mysqli_set_charset($conexion, "utf8");
```

- **Cerrar conexión a la/las bbdd** (incluir al final del código):
  - **mysqli_close()**

```php
mysqli_close($conexion);
```

- **Evitar caracteres que no sean de tipo numerico o de tipo string** (forma para evitar la inyección SQL):
  - **mysqli_real_escape_string()**

```php
$usuario = mysqli_real_escape_string($conexion, $_GET["usu"]);
$contra = mysqli_real_escape_string($conexion, $_GET["con"]);
```

- **Comprobar, después de hacer una consulta, si ha habido registros afectados** (sirve para consultas de tipo SELECT, INSERT, DELETE, UPDATE):
  - **mysqli_affected_rows()**

```php
if(mysqli_affected_rows($conexion)>0){
    echo "Baja procesada";
}else{
    echo "No se ha encontrado usuario";
}
```

Consultas preparadas en mysqli (método procedimental):

- Evitan inyección SQL.
- Son más rápidas y eficientes.

1. **Creamos sentencia SQL sustituyendo el valor del criterio por: '?'.**

```
$sql= "SELECT CÓDIGOARTÍCULO, SECCIÓN, PRECIO, PAÍSDEORIGEN FROM PRODUCTOS WHERE
    PAÍSDEORIGEN = ?";
```

2. **Preparamos la consulta con la función mysqli_prepare().**

   - Nos devuelve un objeto de tipo mysqli_stmt() o RESULSET.

```
$resultado = mysqli_prepare($conexion, $sql);
```

3. **Unimos los parámetros de la sentencia SQL, es decir, con el uso de la sentencia mysqli_stmt_bind_param(), le pasamos el criterio que el usuario escribe y lo sustituimos por el '?' de la sentencia SQL. También le especificamos el tipo de dato que se utilizará como criterio (de tipo texto, numero, etc.).**

   - Nos devuelve true (si se ejecuta correctamente) o false (si no lo hace).

```
$ok = mysqli_stmt_bind_param($resultado, "s", $pais);
```

4. **Ejecutar la consulta con la función mysqli_stmt_execute().**

   - Nos devuelve true (si se ejecuta correctamente) o false (si no lo hace).

```
$ok = mysqli_stmt_execute($resultado);
```

5. **El resultado a las consultas se las guarda en distintas variables (tantas variables como campos encontrados), haciendo uso de la función: mysqli_stmt_bind_result().**

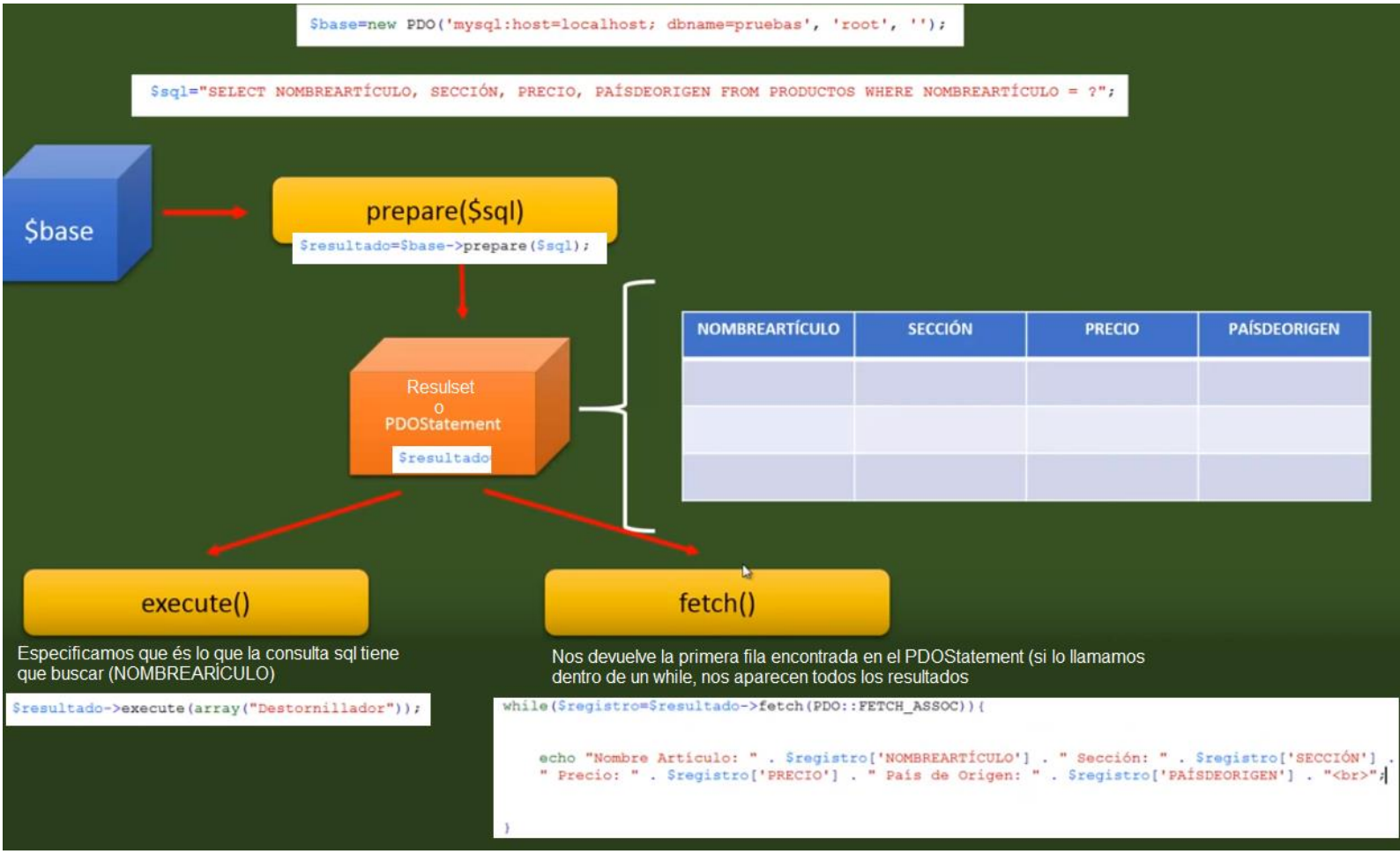
```php
if($ok==false){

    echo "Error al ejecutar la consulta";

}else{

    $ok=mysqli_stmt_bind_result($resultado, $codigo, $seccion, $precio, $pais);

    echo "Artículos encontrados: <br><br>";

    while(mysqli_stmt_fetch($resultado)){

        echo $codigo . " " . $seccion . " " . $precio . " " . $pais . "<br>";
    }

    mysqli_stmt_close($resultado);
}
```

6. **Recorrer lo que ha devuelto la consulta con mysqli_stmt_fetch().**

```php
mysqli_stmt_fetch($resultado)
```

PDO (PHP Data Object): Librería de PHP POO.

Consultas preparadas en PDO (método procedimental):



Encriptar password

Cuando un usuario introduce una contraseña en un formulario y ese formulario envía la información a una base de datos, esa contraseña se tiene que **encriptar**.

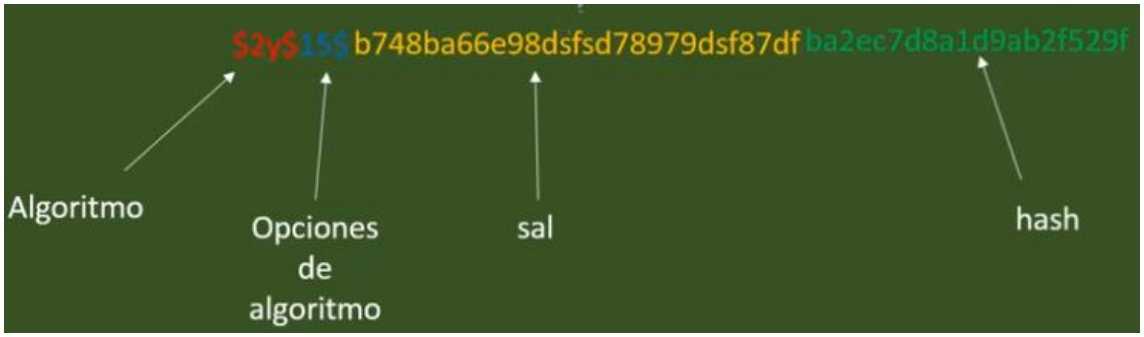
- **Encriptar:** Coger la contraseña del usuario, aplicarle un algoritmo de encriptación, para transformar la misma contraseña en un código indescifrable.
    - Al código resultante se le denomina código **hash**.

Algoritmos disponibles:

MD5	md5()
SHA1	sha1()
SHA256	hash()
BLOWFISH	crypt() / password_hash()

MD5, SHA1 y SHA256 no se recomiendan, ya que existen tablas que se denominan rainbow table que recogen los códigos y el resultado en símbolos/letras de la encriptación (ej. Jhk34 = "A").

BLOWFISH:



Algoritmo: Muestra el algoritmo utilizado ($2y$ en BLOWFISH).

Opciones de algoritmo: Fuerza a la que quieres que actúe el algoritmo. A más fuerza, más va a tardar la página web o el servidor en aplicar el algoritmo para guardar la contraseña, pero más difícil hará que la contraseña se pueda crackear.

Salt: Código aleatorio que se crea en el mismo código. Cambia independientemente de que dos usuarios creen la misma contraseña.

- *La salt se tiene que crear aleatoriamente, nunca manualmente.*

```php
$usuario= $_POST["usu"];
$contrasenia= $_POST["contra"];

$pass_cifrado=password_hash($contrasenia, PASSWORD_DEFAULT);
```

- **PASSWORD_DEFAULT** para que la función aplique la salt automáticamente.

Luego se ejecuta el resulset con la contraseña encriptada:

```php
$resultado->execute(array(":usu"=>$usuario, ":contra"=>$pass_cifrado));
```

Anotaciones

- **Include y require.**

Si se utiliza, en ese momento, el programa va a ir a buscar el archivo que hemos incluido, en el caso de que no se encuentre ese archivo, el programa seguirá. Con require, en cambio, el programa si no encuentra el archivo, no seguirá el flujo de ejecución y se parará.

Incluide_once y require_once se utiliza cuando solo queremos incluir el archivo una única vez. Va bien cuando estamos trabajando en mvc y no se quieren redeclarar objetos una y otra vez (si se utiliza, por ejemplo, require, al volver a esa página en cuestión, si ya se ha hecho una declaración de un objeto, ese objeto se mantendrá ahí y no podremos declarar otro objeto nuevo).