Ejemplo de aplicación PHP: Tienda

Aplicaciones Web / Sistemas Web

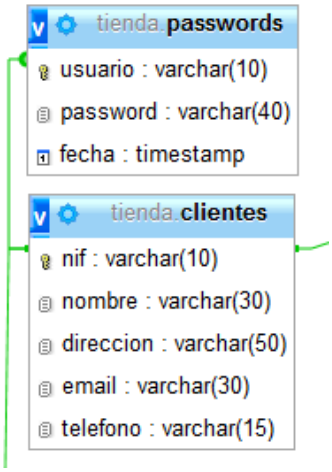
Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

Ejemplo de tienda

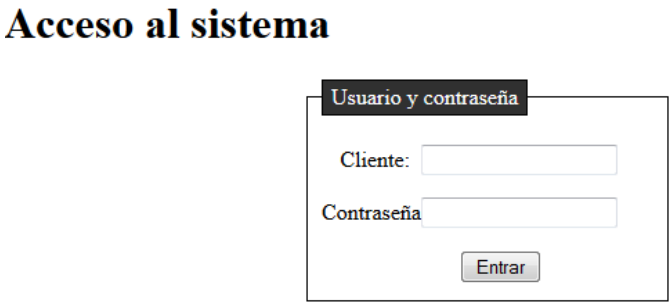
- Temas a tratar
  - Autenticación y control de sesión
  - Carrito de la compra
  - Medios de pago

Login

- Para identificar los usuarios se guarda el password codificado
  - Los algoritmos de hashing como SHA o MD5, pueden ser crackeados
    - Pero para la práctica pueden valer
  - Hay algoritmos más sofisticados: bcrypt encryption o sha-256/512 con refuerzo de claves (*key stretching*)
    - Se puede encontrar en http://blackbe.lt/php-secure-sessions/



Página de login




```
<form action="procesarLogin.php" method="POST">
<fieldset>
<legend>Usuario y contrase&ntilde;a</legend>

<p><label>Cliente:</label> <input type="text" name="usuario" maxlength="10"/></p>

<p><label>Contrase&ntilde;a:</label> <input type="password" name="password"
maxlength="40"/></p>

<button type="submit">Entrar</button>
</fieldset>
</form>
```

Variables de sesión

- Una sesión tiene las variables que determinan un contexto.
- En este ejemplo:
  - login: true
  - usuario: id del usuario en la BD
  - nombre: nombre del usuario en la BD
  - carrito: asignado al usuario
- Para saber si una sesión está activa
  if (isset($_SESSION["login"]))
- Al acabar la sesión (logout) se eliminarán todas las variables de sesión
  - Se produce al hacer
    session_destroy();
  - También se pueden eliminar todas con
    unset($_SESSION);
    - Esto puede ser útil en algunos casos para reiniciar la sesión

Página de proceso de login

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
   $uname = limpia_sql(htmlspecialchars($_POST['usuario']));
   $pword = limpia_sql(htmlspecialchars($_POST['password']));

   // Conexión a la BD tienda
   $bd = conectaBD();
   // Comprueba si el registro (usuario,password) está en la BD:
   $query = "SELECT * FROM passwords WHERE usuario = $uname AND password =md5($pword)";
   $resultado = $bd->query($query);
   $numregistros = $resultado->num_rows;

   if ($resultado) {
         if ($numregistros == 1) {   // El registro del usuario y password en la BD
              $_SESSION["login"] = true;
              $_SESSION["usuario"] = $uname;
              // Mira en la BD cual es el nombre del usuario que se ha logeado
              $resultado->free();
              $query = "SELECT * FROM clientes WHERE nif = $uname";
              $resultado = $bd->query($query);
              $registro = $resultado->fetch_assoc();
              $_SESSION["nombreusuario"] = $registro["nombre"];
              // Crea un carrito para este usuario en la sesión
              $_SESSION["carrito"] = new carrito($usuario);
       }
   }
   $bd->close();
}  ?>
```

Ataques de inyección SQL

- Es importante preparar la entrada cuando se vaya a utilizar para acceder a la BD
- Por ejemplo (del manual de PHP http://www.php.net/manual/es/function.mysql-real-escape-string.php)

```
<?php
// No hemos comprobado $_POST['password']
// Si lo que viene es lo siguiente
$_POST['username'] = 'aidan';
$_POST['password'] = "' OR ''='";
// Consultar la base de datos para comprobar si existe algún usuario que coincida
$consulta =
   "SELECT * FROM users WHERE user='{$_POST['username']}' AND password='{$_POST['password']}'";
mysql_query($consulta);

// Esto significa que la consulta enviada a MySQL sería:
echo $consulta;
?>
```

- La consulta enviada a MySQL será:

  SELECT * FROM users WHERE user='aidan' AND password='' OR ''=''
  - Esto permitiría a alguien acceder a una sesión sin una contraseña válida.

Ataques de inyección SQL

```
function limpia_sql($texto) {
    if (get_magic_quotes_gpc()) {
        $texto = stripslashes($texto); // quita barras \ de un string
    }
    if (!is_numeric($texto)) {  // quita secuencias de escape peligrosas
        $texto = "'" . mysql_real_escape_string($texto) . "'";
    }
    return $texto;
}
```

Página de proceso de login

```
<?php
if (isset($_SESSION["login"])) {
        echo "<h1>Bienvenido ". $_SESSION['nombreusuario'] . "</h1>";
        echo "<p>Use el men&uacute; de la izquierda para navegar.</p>";
} else {
        echo "<h1>ERROR</h1>";
        echo "<p>El usuario o contrase&ntilde;a no son v&aacute;lidos.</p>";
}
?>
```


Registro

Registro de nuevo usuario

Datos de nuevo cliente

NIF:
Nombre:
Dirección:
Email:
Teléfono:
Contraseña
Registrarse

```
<form action="procesarRegistro.php" method="POST">
<fieldset>
<legend>Datos de nuevo cliente</legend>
<p><label>NIF:</label> <input type="text" name="nif" maxlength="10"/></p>
<p><label>Nombre:</label> <input type="text" name="username" maxlength="30"/></p>
<p><label>Direcci&oacute;n:</label> <input type="text" name="direccion"
maxlength="50"/></p>
<p><label>Email:</label> <input type="text" name="email" maxlength="30"/></p>
<p><label>Tel&eacute;fono:</label> <input type="text" name="telefono"
maxlength="15"/></p>
<p><label>Contrase&ntilde;a:</label> <input type="password" name="password"
maxlength="40"/></p>
<button type="submit">Registrarse</button>
</fieldset>
</form>
```

Página de proceso de registro

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
   $uname = limpia_sql(htmlspecialchars($_POST['nif']));
   $pword = limpia_sql(htmlspecialchars($_POST['password']));
   $pLength = strlen($pword);
   if ($pLength >= 8 && $pLength <= 20) {
        $error = "";
   }
   else {
        $error = $error . "El password debe tener entre 8 1 20 caracteres";
   }

   $nombre = limpia_sql(htmlspecialchars( $_POST['username']));
   $direccion = limpia_sql(htmlspecialchars($_POST['direccion']));
   $email = limpia_sql(htmlspecialchars($_POST['email']));
   $telefono = limpia_sql(htmlspecialchars($_POST['telefono']));
```


Página de proceso de registro

```
if ($error == "") {
   $bd = conectaBD();

   // Se comprobará si ya existe el usuario
   $query = "SELECT * FROM passwords WHERE usuario = $uname";
   $resultado = $bd->query($query);
   $numregistros = $resultado->num_rows;
   if ($numregistros == 1) { // Ya existe un usuario con ese uname (nif)
        $error = $error . "Ya existe este usuario";
   }
   else { // Se procede a registrar el usuario
        // 1) En la tabla de clientes
        $query = "INSERT INTO clientes (nif, nombre, direccion, email, telefono)
VALUES ($uname, $nombre, $direccion, $email, $telefono)";
        $resultado = $bd->query($query);

        // 2) En la tabla de passwords
        $pword = md5($pword);       // El password se guardará codificado
        $query = "INSERT INTO passwords (usuario, password, fecha) VALUES ($uname,
'$pword', CURRENT_TIMESTAMP)";
        $resultado = $bd->query($query)
        // Y se deja al usuario dentro de la sesión
        $_SESSION["login"] = true;
        $_SESSION["usuario"] = $uname;
        $_SESSION["nombreusuario"] = $nombre;
   }
   $bd->close();
}
```

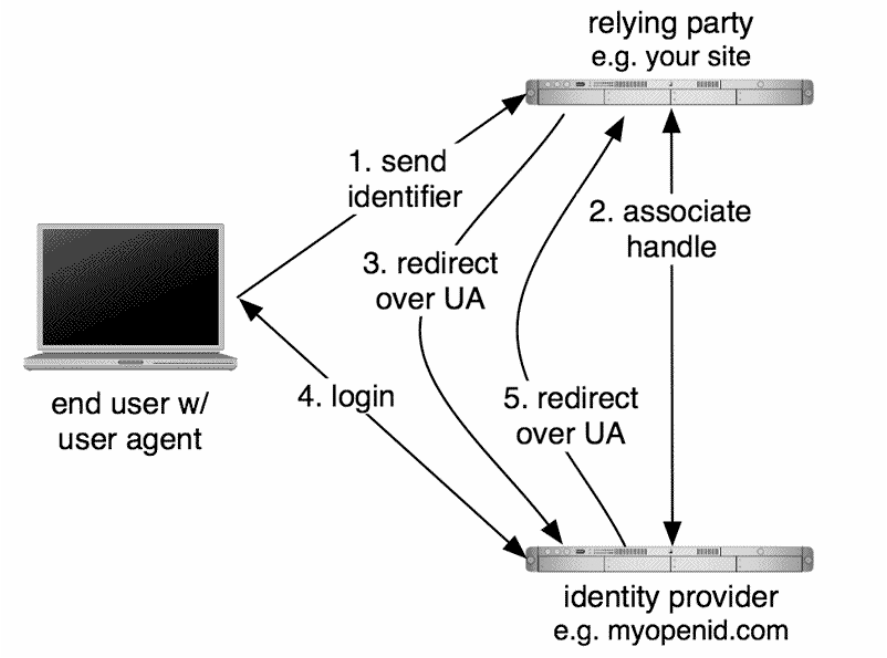
Ejercicios

- Desarrollar la página de logout
- Codificar el password antes de enviarlo
  - Usando javascript
- Enviar un email de confirmación
  - No se crea una sesión con el usuario si antes no ha validado su email
- Comprobar siempre si los usuarios ya están logeados
  - Si no, podrían logearse repetidamente sin cerrar el navegador
- Guardar información sobre la actividad del usuario
  - Horas de login y logout, historial de acciones, su dirección IP, etc.
- Implementa una página para recordar el password enviándoselo a su email
- Investigar cómo hacer la autenticación con OpenID
  - http://openid.net/developers/libraries/

OpenID

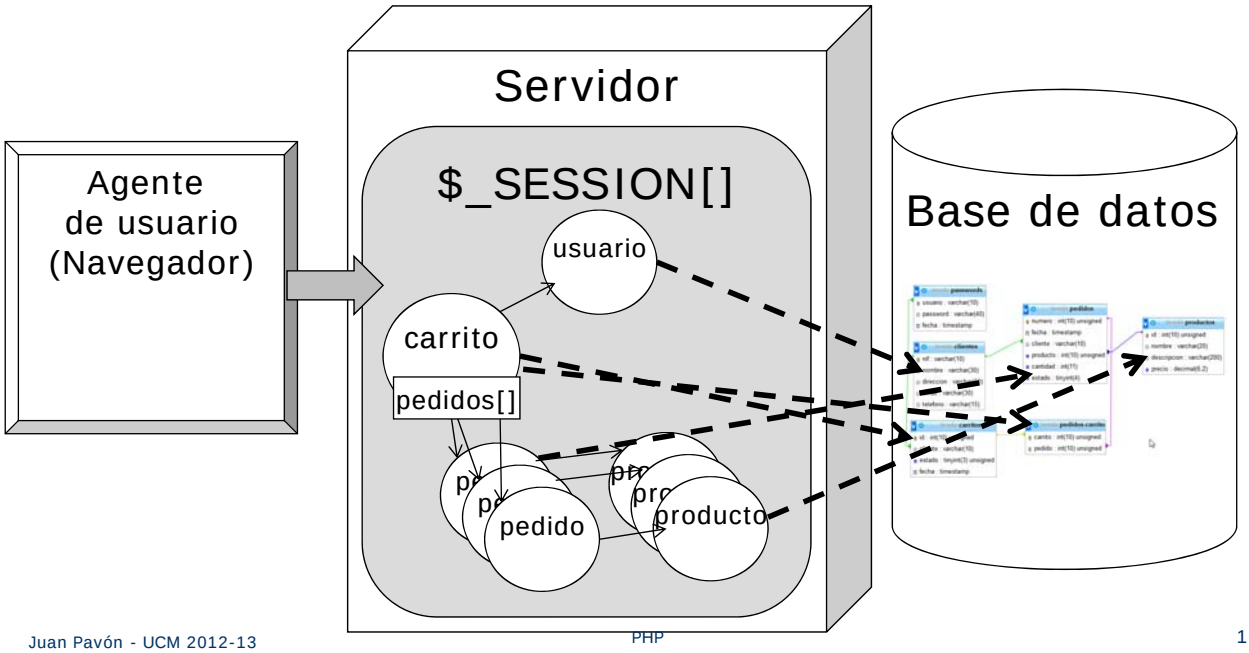
- Solución para evitar tener múltiples nombres de usuario/passwd
- Permite distribuir la gestión de los perfiles de usuario
  - Y que el usuario decida quién gestiona su perfil y autenticación
- En OpenID se consideran tres partes:
  - El usuario (cliente)
  - El proveedor de identidad
    - Un servidor web donde el usuario ya esté registrado previamente
    - Ejemplos: Google+, Yahoo!, WordPress, PayPal, etc.
  - El sitio web (relying party)
    - Confiará en la identidad gestionada por el proveedor de identidad
- Librerías OpenID en casi todos los lenguajes
  - Java, PHP, C/C++, C#, Python

OpenID - Funcionamiento



Carrito de la compra

- Persistencia: MySQL
- Gestión en servidor: PHP
- Pre-proceso en cliente: JavaScript

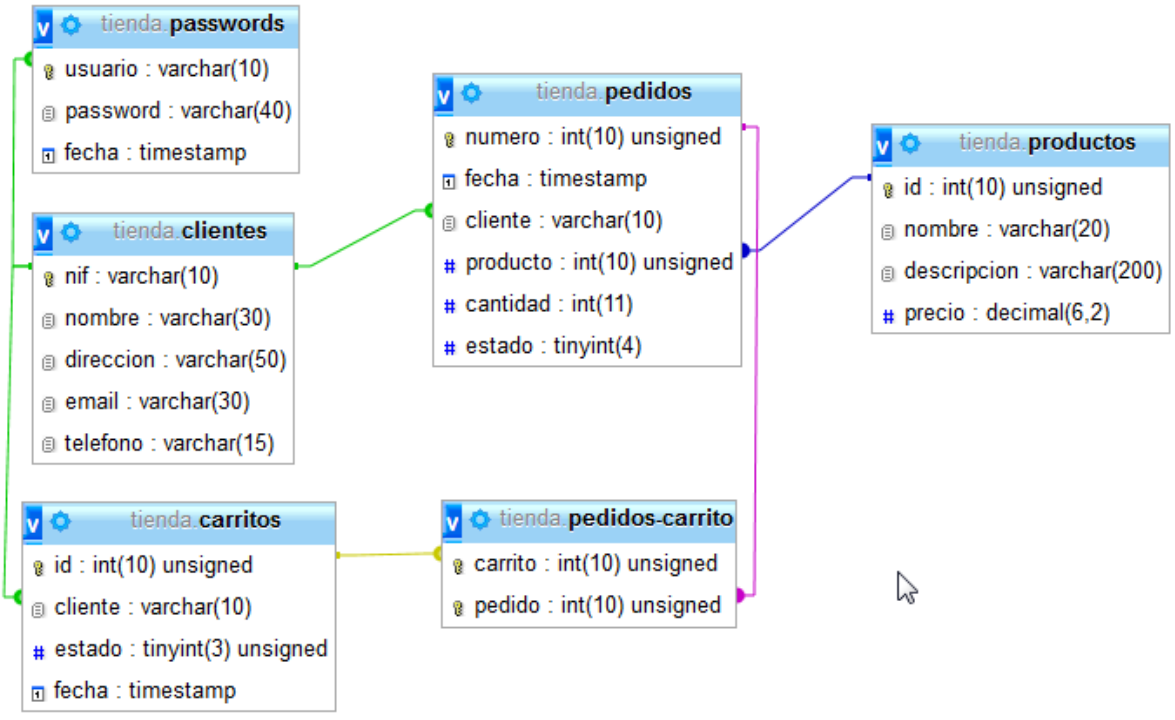


Operaciones y atributos de un carrito

- Operaciones
  - Inicializar (reset)
  - Poner producto
  - Sacar producto
  - Ver productos
  - Calcular gastos envío
  - Tramitar pedido
  - Pagar
- Atributos
  - Lista de productos
  - Estado
    - Comprando
    - Guardado
    - Tramitando
    - Pagando
    - Finalizado

Carrito de la compra: Base de datos

- Añadir el carrito como una tabla más de la BD



Implementación del carrito

- Clases del modelo de datos:
  - Clase cliente
  - Clase carrito
  - Clase producto
  - Clase pedido
- Páginas para recibir las peticiones del cliente
  - login.php, procesaLogin.php
  - logout.php
  - registro.php, procesaRegistro.php
  - ver_catalogo.php
  - ver_producto.php
  - ver_carrito.php
  - comprar_producto.php
  - etc.

Pagos

- Las soluciones de pago ofrecen sus propias APIs
- Ejemplo: PayPal
  - PayPal Instant Payment Notification Guide:
    https://developer.paypal.com/webapps/developer/docs/classic/ipn/integration-guide/IPNIntro/
  - http://www.micahcarrick.com/paypal-ipn-with-php.html

Pago con PayPal

- Sistema PayPal Instant Payment Notification (IPN)
  - Un servicio de mensajes que notifica eventos relacionados con transacciones PayPal
    - Tiene en cuenta la seguridad de las transacciones
  - Los mensajes IPN permiten automatizar funciones administrativas como:
    - Completar órdenes de pago
    - Seguimiento de clientes
    - Gestionar el estado de una transacción
  - Sistema de mensajes asíncrono
    - No garantiza una fiabilidad 100%
      - Se pueden perder o retrasar mensajes IPN
      - PayPal reenvía los mensajes IPN hasta 15 veces durante 4 días
  - El propietario del sitio web tiene que implementar un listener
    - Disponible todo el tiempo posible

Instant Payment Notification (IPN)

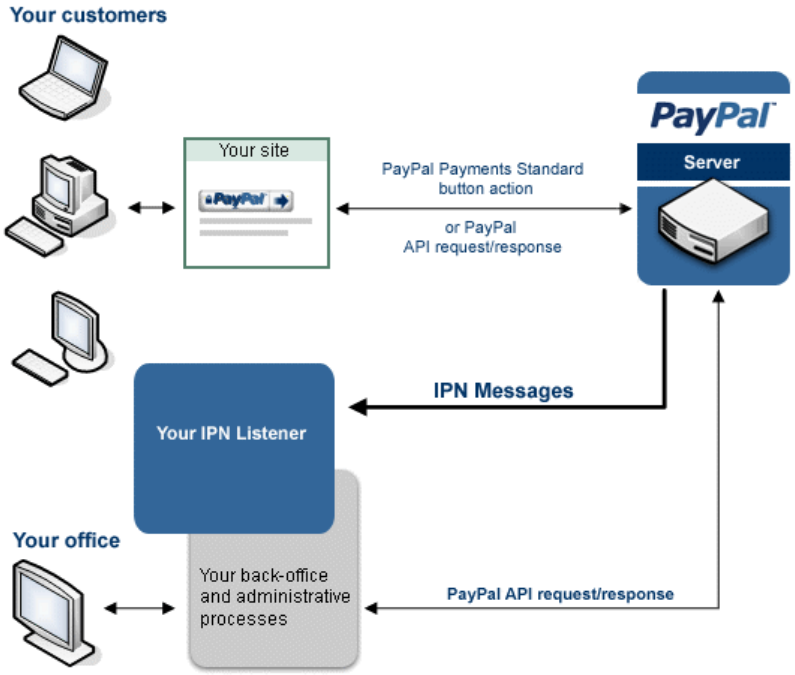


Imagen de la documentación de PayPal:
https://developer.paypal.com/webapps/developer/docs/classic/ipn/integration-guide/IPNIntro/

Protocolo de autenticación de mensajes IPN

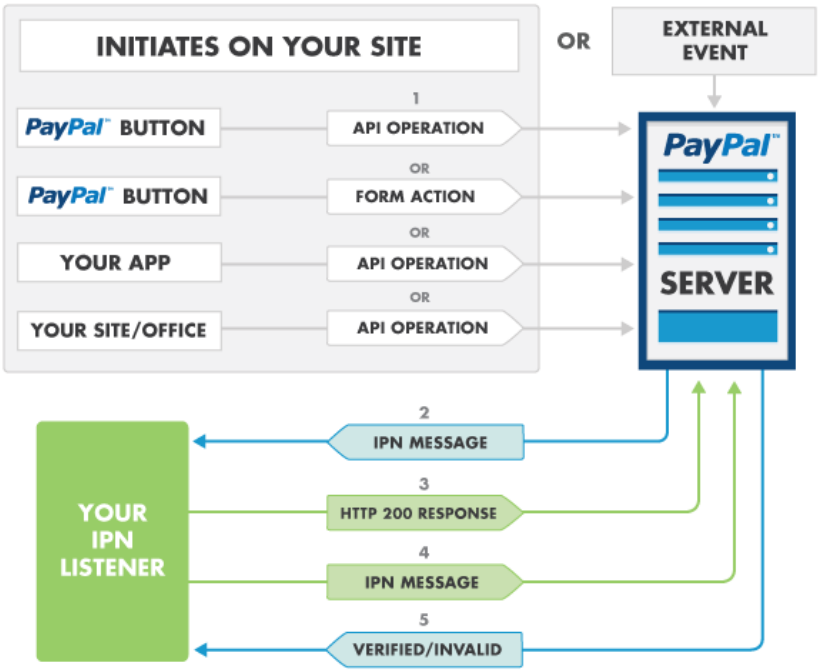


Imagen de la documentación de PayPal:
https://developer.paypal.com/webapps/developer/docs/classic/ipn/integration-guide/IPNIntro/

Funcionamiento

- En la página *ver_carrito.php* se pone un botón <Pagar con PayPal>
- La página siguiente para el cliente será *compra_finalizada.php*

Funcionamiento

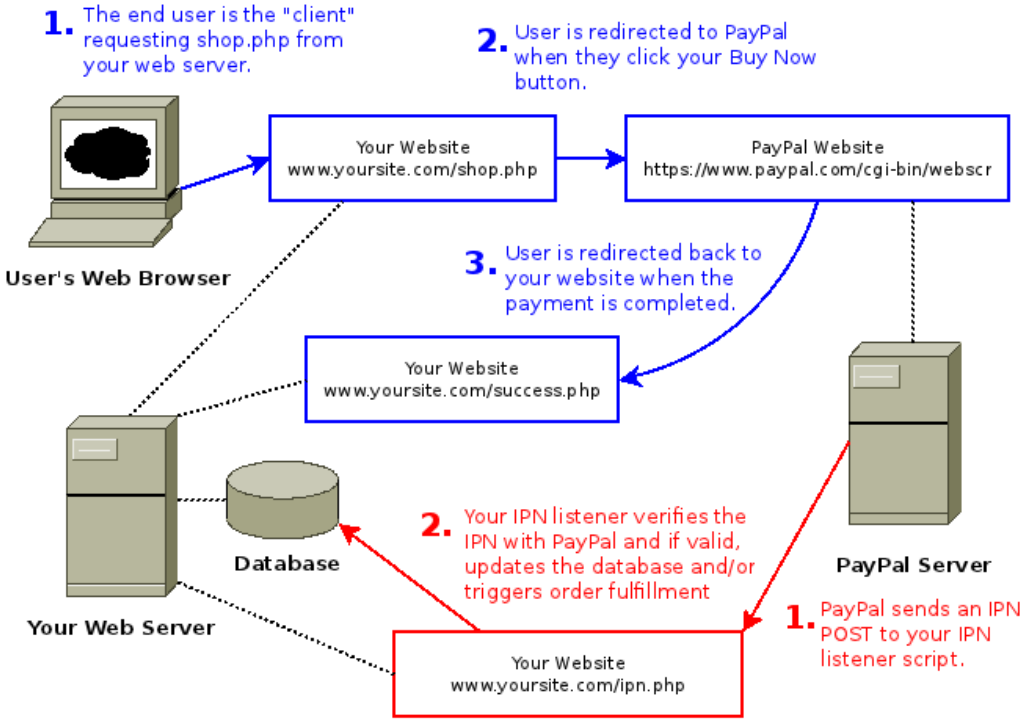


Diagrama de: http://www.micahcarrick.com/paypal-ipn-with-php.html

Desarrollo de script para usar PayPal

- Crear una cuenta sandbox para hacer pruebas
  - Con un servidor y cliente (particular y/o empresa) de pruebas

https://developer.paypal.com/webapps/developer/docs/classic/lifecycle/ug_sandbox

- Crear un listener
  - Fichero ipn.php
    - El URL de este fichero será el que se indique a PayPal para que envíe los POST con los mensajes IPN
    - Como plantilla se puede usar el software de https://github.com/Quixotix/PHP-PayPal-IPN

Anexo: Depurar con XDebug

- Instrucciones de instalación y uso en:
  - http://wiki.eclipse.org/Debugging_using_XDebug
  - http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=depurarEnPHPEclipsePDTXdebug
- Instalación
  - Descargar la versión apropiada de http://xdebug.org/docs/install
    - Ponerla en el directorio ext de php
    - Configurar php.ini para usar Xdebug
      - Especialmente **xdebug.remote_enable** = 1
  - Configurar eclipse
    - Window->Preferences->PHP->Debug
      - Indicar que se va a usar Xdebug
    - Comprobar la URL en Debug Configurations
- Poner breakpoints en el código
  - Por ejemplo en session_start()
  - Depuerar paso a paso