Índice: Tema 4.2

Índice: Tema 4.2

Interacción con el usuario

- ✓ Los formularios HTML constituyen el primer mecanismo de interacción entre las páginas web y el usuario. El usuario rellena el formulario escribiendo datos o seleccionándolos de entre varios posibles.
- ✓ La información suministrada en cada campo se almacena en una variable con el mismo nombre del atributo **`name`** del elemento correspondiente, y puede ser:
  - Enviada a una dirección de correo electrónico.
  - Procesada directamente por la página (por ejemplo, con JavaScript).
  - Enviada al servidor para su procesamiento (por ejemplo, una página en PHP).
- ✓ Para indicar el destino al que serán enviados los datos obtenidos se utiliza el atributo **`action`** del elemento **`<form>`**:

```
<form action="http://localhost/pagina.php">
```

Métodos de envío (1)

- Además, el atributo **`method`** especifica la forma en la que se deben transmitir los datos del formulario. Hay dos valores posibles para dicho atributo:

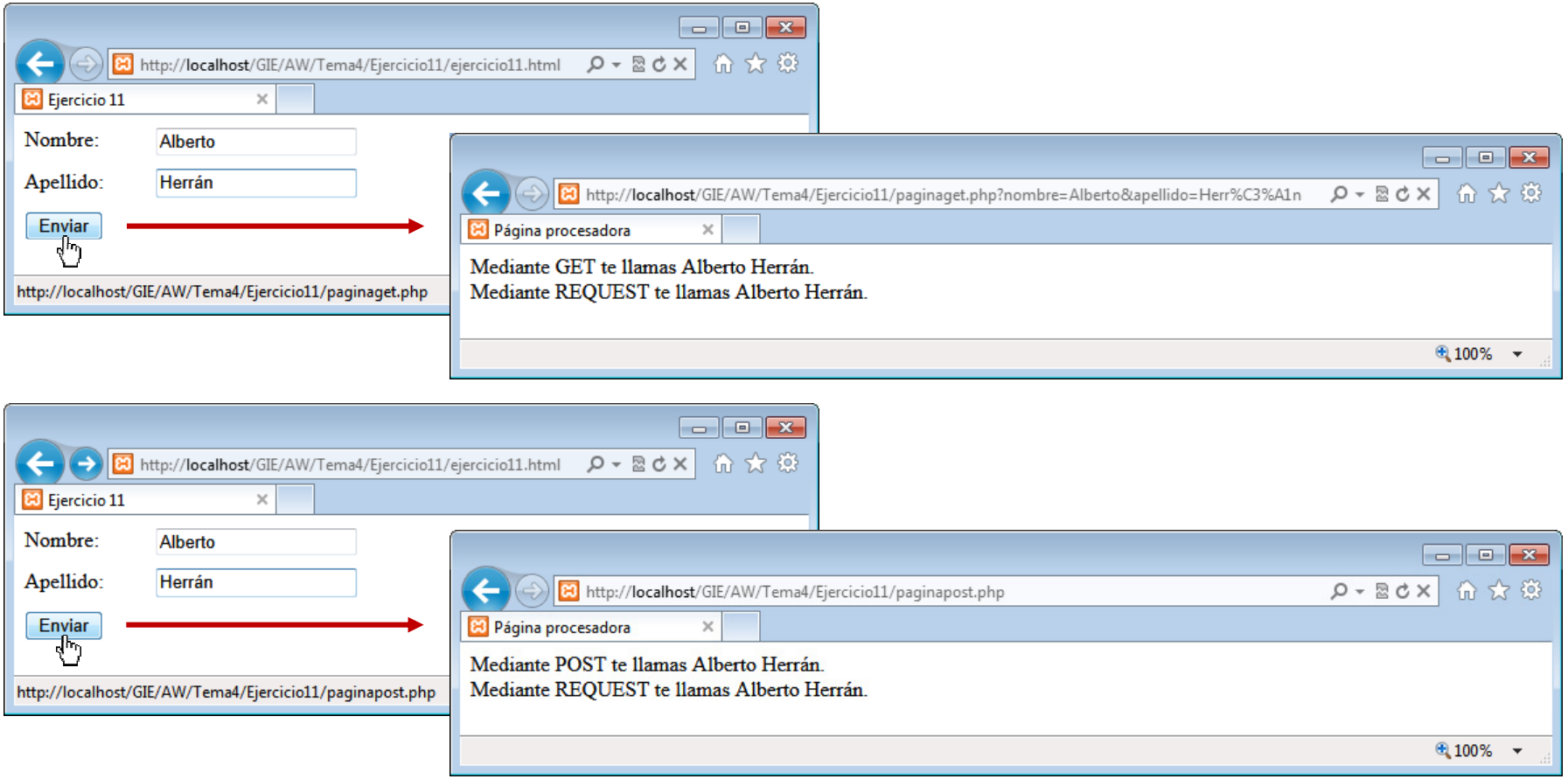
- **`get`** Es el método predeterminado.
  - Los datos se añaden al final del URL precedidos por un signo de interrogación (?).
  - Pares **clave=valor** separados por &.
  - http://localhost/pagina.php?**nombre=Alberto&apellido=Herran**
  - En la página procesadora se accede a los datos mediante el array **`$_GET`**.

- **`post`** Es el método habitual en envíos con formularios.
  - Los datos se incrustan en el encabezamiento de la solicitud HTTP, no quedando a la vista.
  - En la página procesadora se accede a los datos mediante el array **`$_POST`**.

- Además, en ambos casos se dispone del array **`$_REQUEST`**.

➢ Métodos de envío (2)

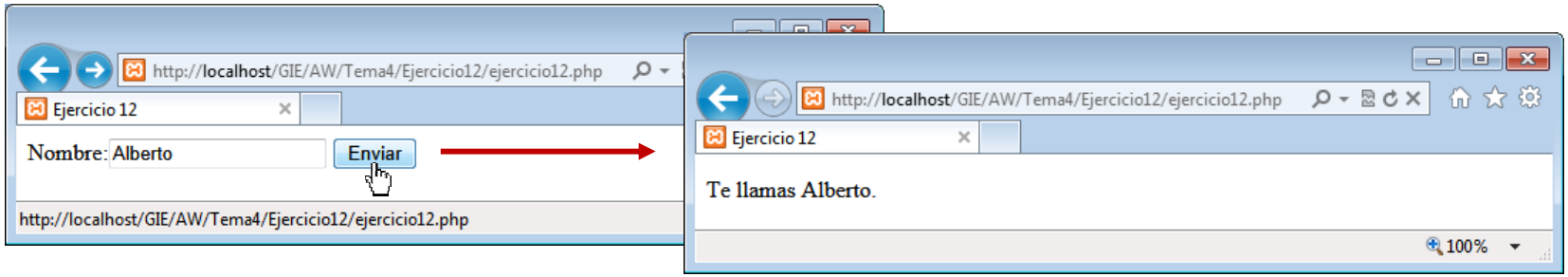
✓ Prueba los métodos anteriores para el envío de los datos introducidos en un par de campos de texto:



Métodos de envío (3)

- La propia página puede ser la encargada de procesar los datos:

```
<body>
    <?php if(isset($_POST["nombre"])) { ?>
        <p>Te llamas <?php echo $_POST["nombre"]; ?>.</p>
    <?php } else { ?>
        <form action="<?php echo($_SERVER["PHP_SELF"]); ?>" method="post">
            Nombre: <input type="text" name="nombre" />
            <input type="submit" value="Enviar" />
        </form>
    <?php } ?>
</body>
```

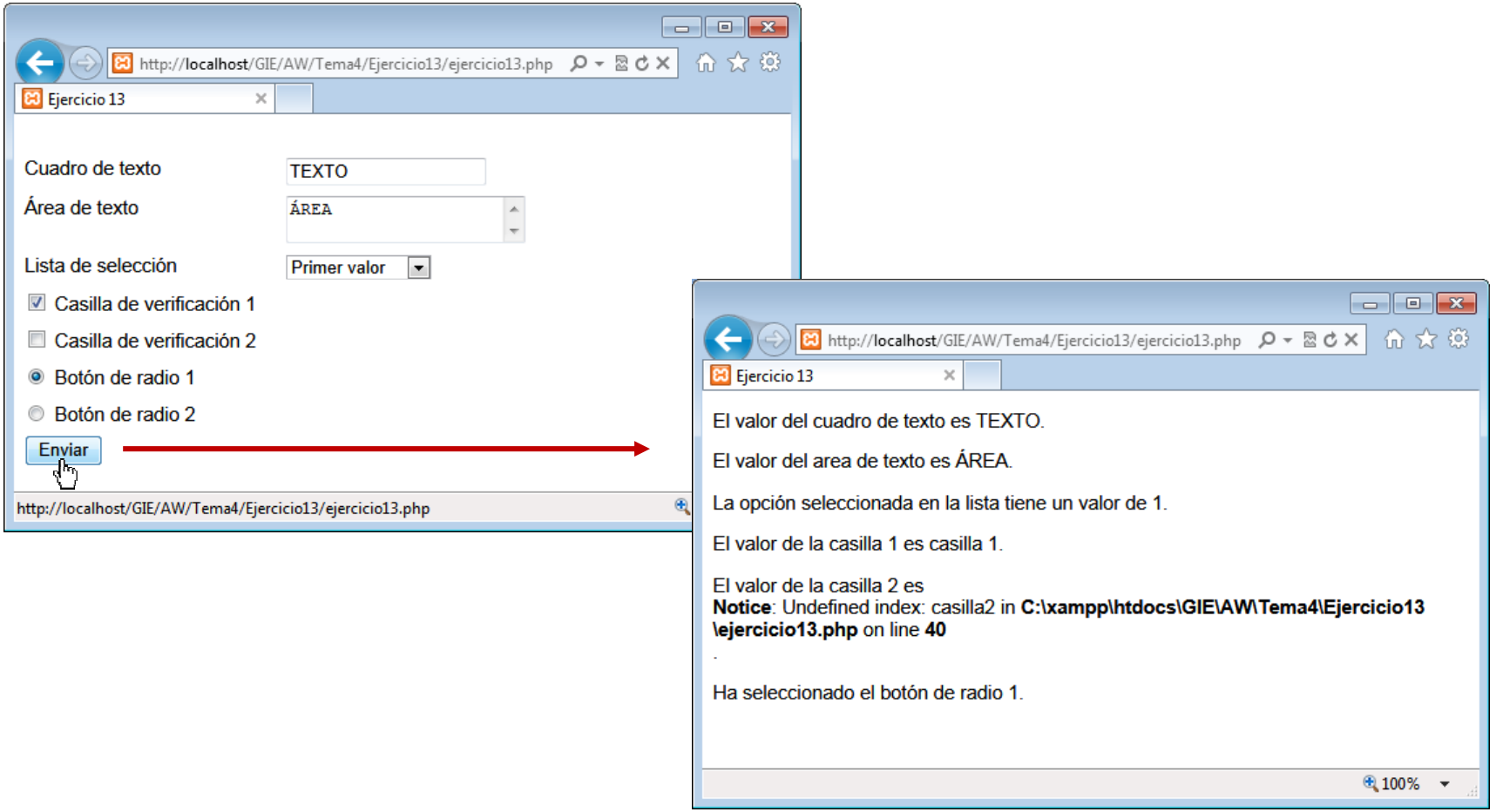


Otros campos (1)

- ✓ Como se acaba de ver, la información suministrada en un cuadro de texto (`text`) se almacena en una variable con el nombre especificado en su atributo `name`, lo mismo ocurre con un área de texto (`textarea`).
- ✓ Para las casillas de verificación (`checkbox`) se envía una variable con el nombre de su atributo `name` sólo si la casilla está activada. El valor de la variable es `on` a no ser que se haya especificado otro mediante su atributo `value`.
- ✓ En el caso de un grupo de botones de radio se asigna a la variable especificada en su atributo `name` el valor especificado en el atributo `value` del botón activado.
- ✓ Finalmente, en las listas de selección (`select`) el valor especificado en el atributo `value` de la opción seleccionada se asigna a la variable especificada en el atributo `name` de la lista. Si la lista es de selección múltiple los valores asociados a las opciones seleccionadas se almacenan en un array, por lo que el atributo `name` de la lista debe contener el nombre de la variable seguido por `[]`.

➢ Otros campos (2)

✓ Repetir con PHP el ejercicio 37 del Tema 3. Seguir el patrón del Ejercicio 12.



Funciones de utilidad (1)

✓ Validar elementos obligatorios:

- Para tener la certeza de que se ha escrito algo en un elemento obligatorio, podemos usar la función **`strlen()`**.

```
if(strlen($_POST["nombre"])==0) {
    $errores[]="Tiene que introducir su nombre";
}
```

✓ Validar elementos numéricos:

- Para asegurarnos que un valor enviado es un entero o un número en coma flotante, podemos usar las funciones **`intval()`** y **`floatval()`** que convierten de cadena a número. Una vez que hemos convertido a número, realizamos la conversión inversa con **`strval()`** y comparamos.

```
if($_POST["precio"]!=strval(floatval($_POST["precio"])) {
    $errores[]="Introduzca un precio válido";
}
```

Funciones de utilidad (2)

- ✓ Validar cadenas:
  - Para comprobar que se ha introducido texto que no sean únicamente caracteres en blanco, podemos usar **`trim()`** y **`strlen()`**.

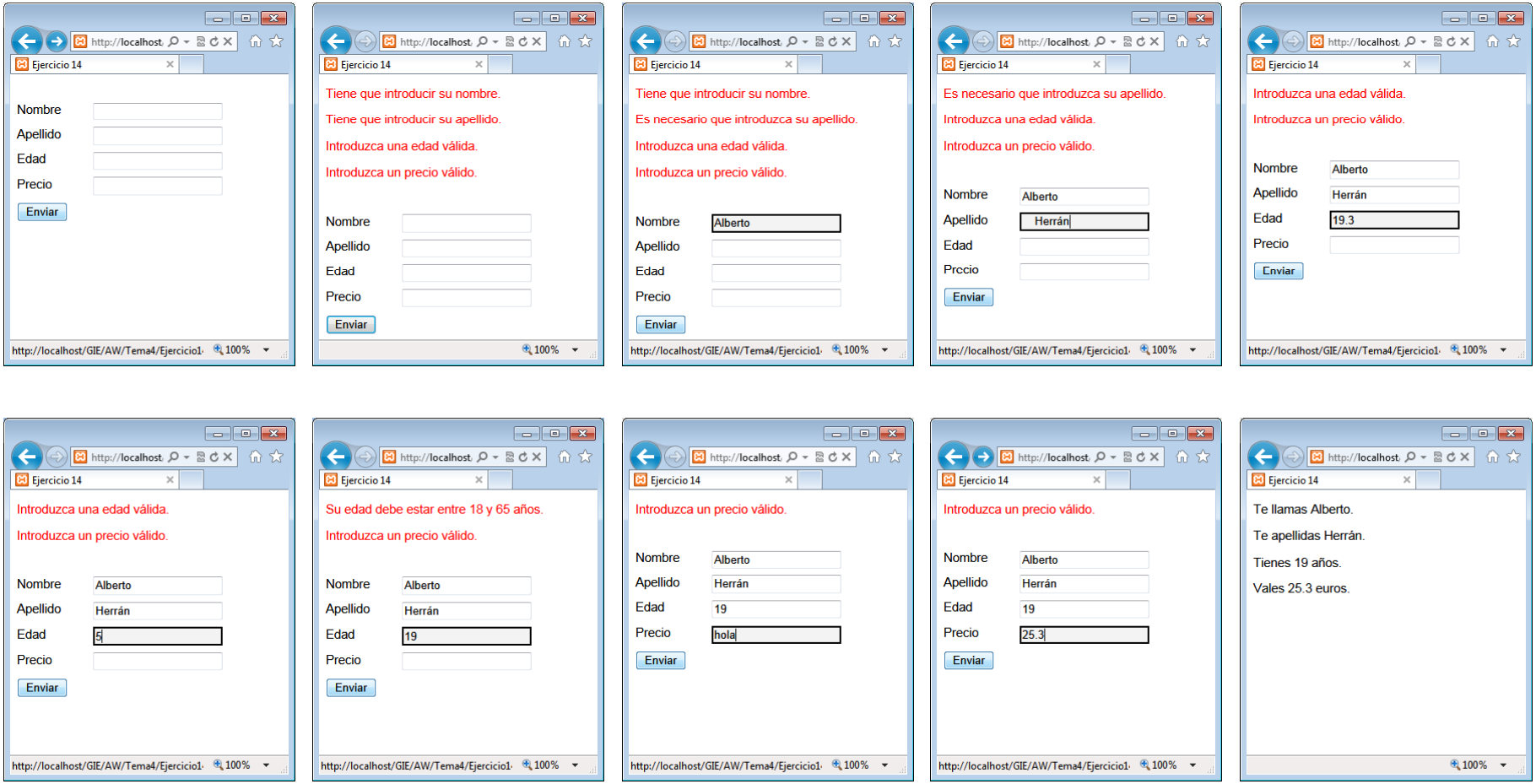
```
if(strlen(trim($_POST["apellido"]))==0) {
    $errores[]="Tiene que introducir su apellido";
}
```

- ✓ Rangos numéricos:
  - Comprobamos primero que se trata de un número y después si está entre el rango.

```
if($_POST["edad"]!=strval(intval($_POST["edad"])) {
    $errores[]="Introduzca una edad válida";
}
elseif(($_POST["edad"]<18)||($_POST["edad"]>65)) {
    $errores[]="Su edad debe estar entre 18 y 65 años";
}
```

➢ Funciones de utilidad (3)

✓ Utilice el código anterior para programar el siguiente formulario con validación:



Índice: Tema 4.2

MySQL

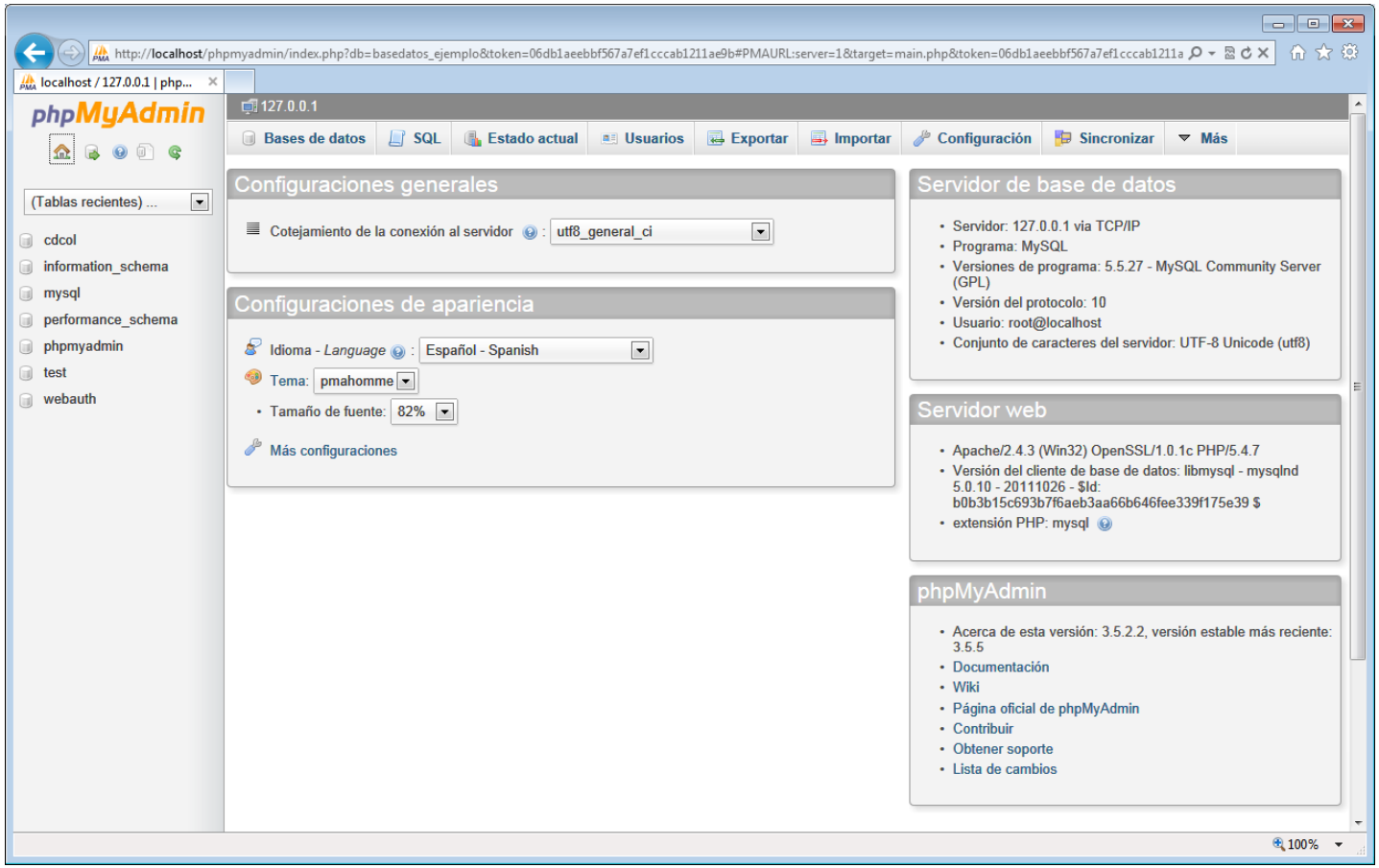
- ✓ MySQL es un sistema de gestión de bases de datos relacionales rápido, sólido y flexible.
- ✓ Es ideal para crear bases de datos con acceso desde páginas web dinámicas, o para cualquier otra solución profesional que implique almacenar datos, teniendo la posibilidad de realizar múltiples y rápidas consultas.
- ✓ Características:
  - Sistema cliente/servidor.
  - Permite trabajar como servidor multiusuario y de procesamiento múltiple.
  - Para cada conexión con el servidor se crea un subproceso para manejar la solicitud del cliente, controlando el acceso simultaneo de varios clientes a los datos.
  - Se puede controlar el acceso a sólo usuarios autorizados.
  - Utiliza el lenguaje SQL (lenguaje de consulta más utilizado para acceder a bases de datos).

phpMyAdmin

- ✓ phpMyAdmin es una alternativa sencilla e intuitiva para administrar las bases de datos de MySQL a través de un servidor web.
- ✓ La aplicación phpMyAdmin no es más que un conjunto de páginas escritas en PHP y alojadas en directorio web del servidor.
- ✓ Mediante las diferentes páginas de la aplicación se puede:
  - Consultar las bases de datos disponibles.
  - Crear nuevas bases de datos así como tablas.
  - Realizar consultas, insertar y borrar registros.
  - Administrar usuarios y sus privilegios.
  - Hacer copias de seguridad de las bases de datos.
  - Etc…

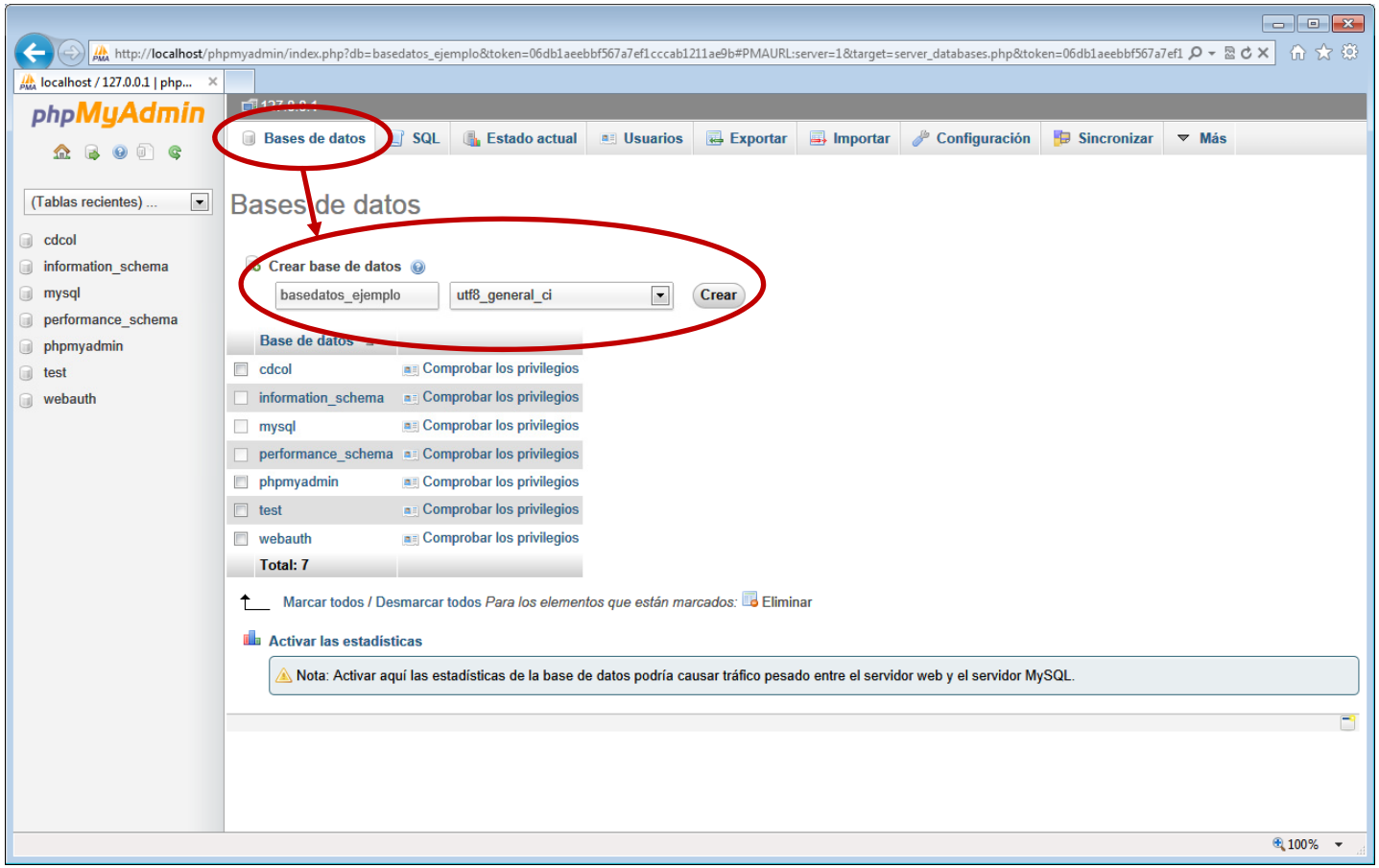
➢ phpMyAdmin (2)

✓ El acceso a la página principal de phpMyAdmin se hace desde el panel de control:



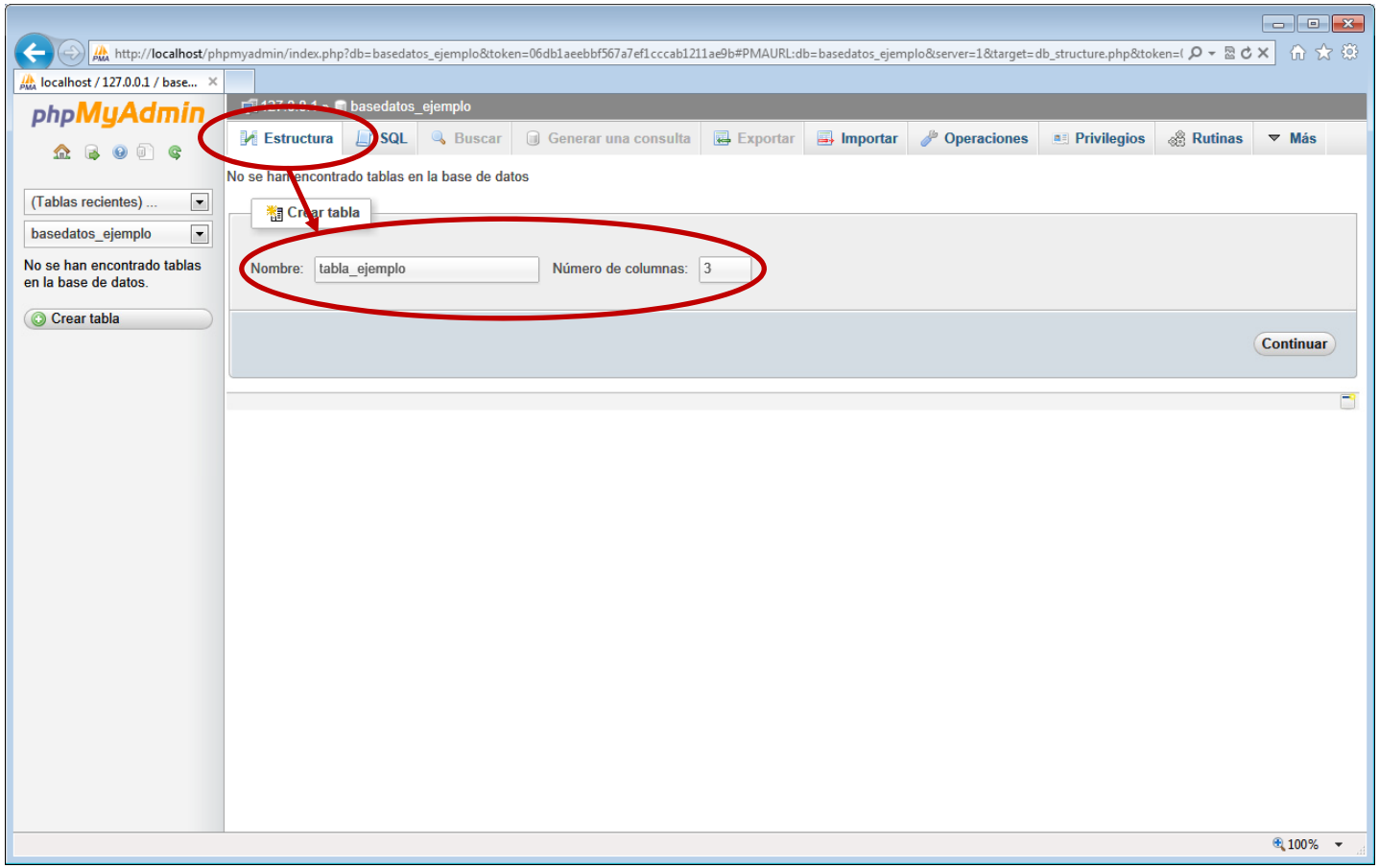
➢ Creación de una base de datos

✓ Crear una base de datos denominada "**basedatos_ejemplo**" utf8:



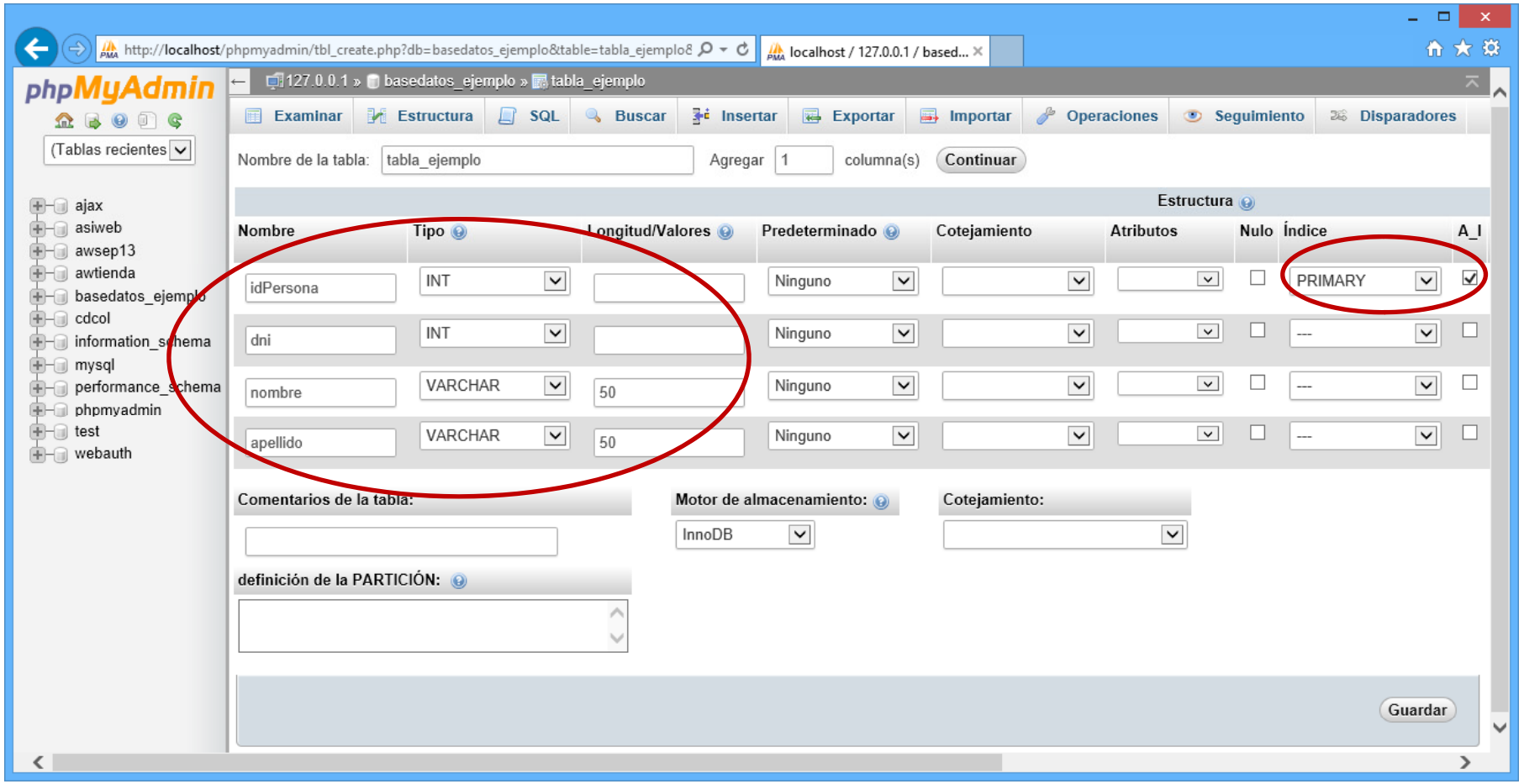
➢ Creación de una tabla (1)

✓ Pulsar sobre la BDB creada y crear una tabla "**tabla_ejemplo**" con 4 columnas:



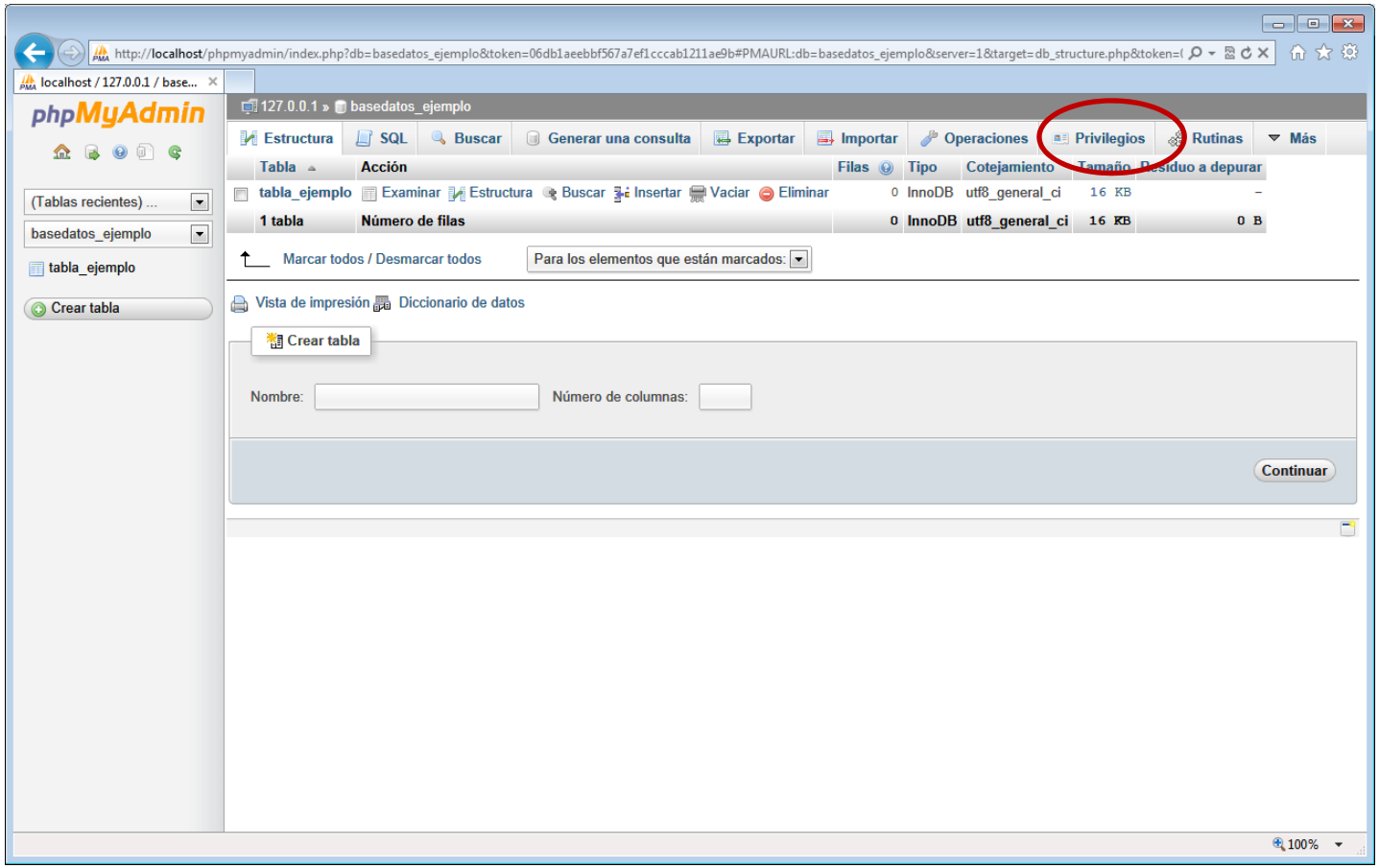
➢ Creación de una tabla (2)

- ✓ Añadir las siguientes columnas para cada registro de la tabla:



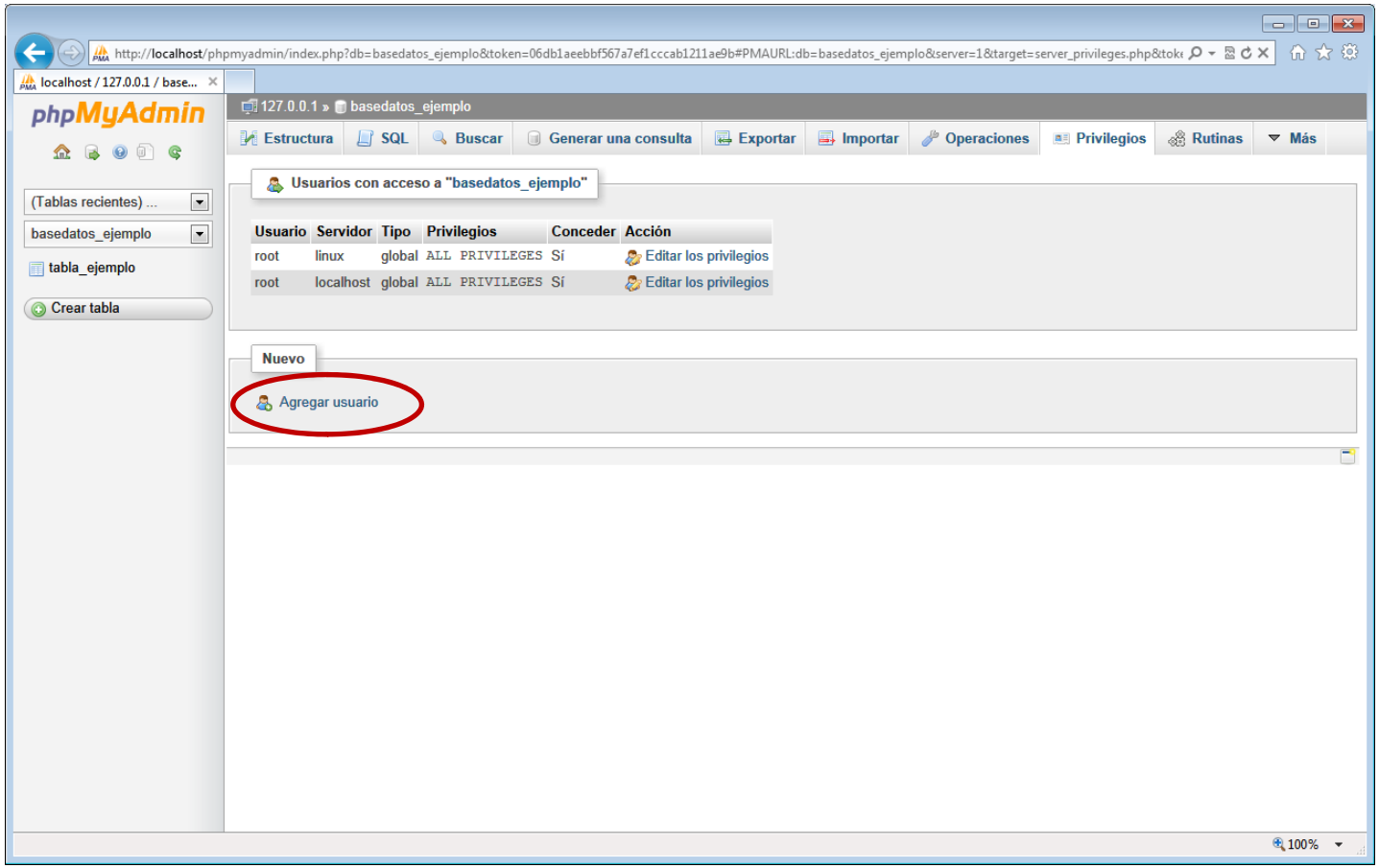
➢ Añadir usuario y contraseña (1)

✓ Tras crear la tabla aparece la siguiente pantalla donde pulsamos en “Privilegios”:



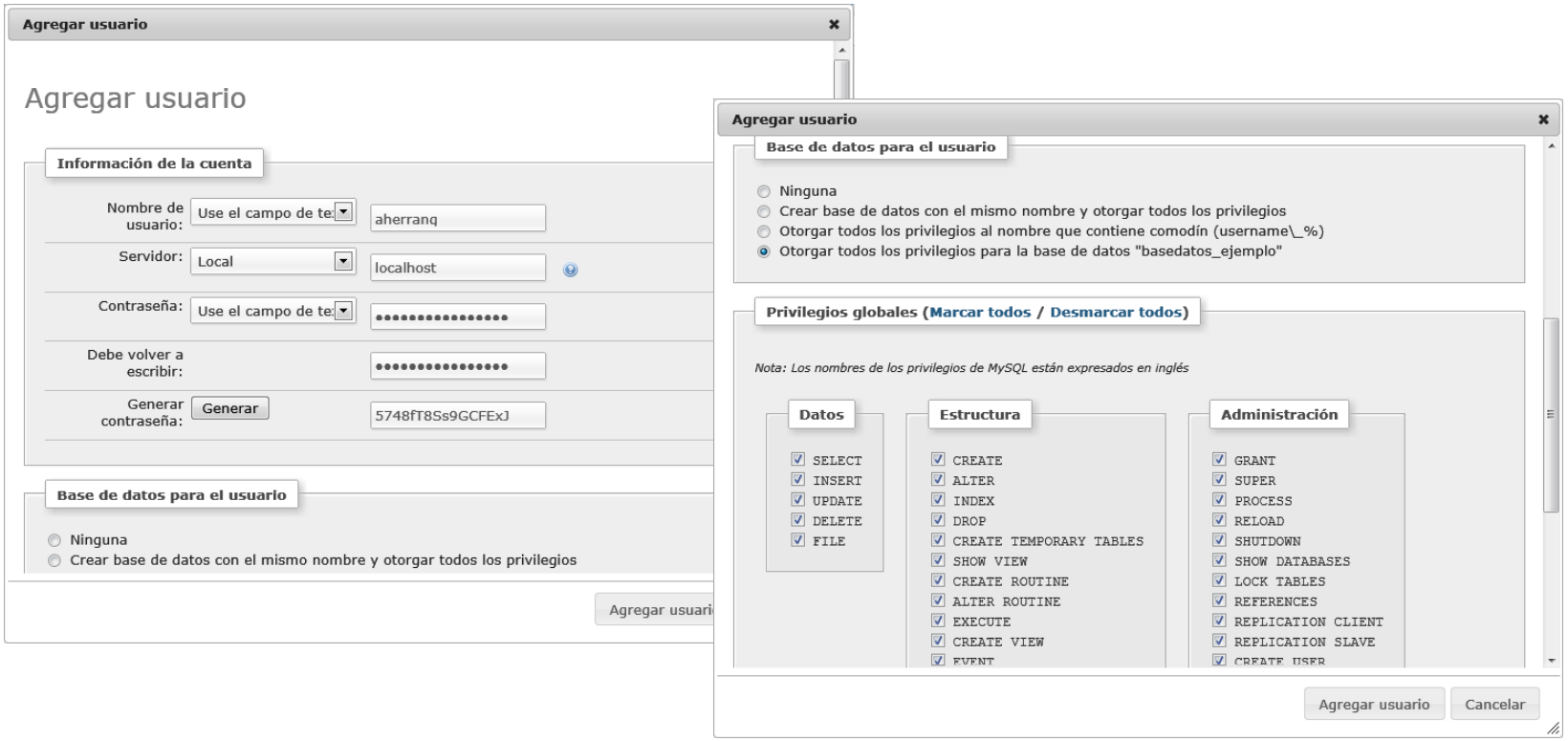
➢ Añadir usuario y contraseña (2)

✓ Dentro de la pestaña privilegios le damos a “Agregar usuario”:



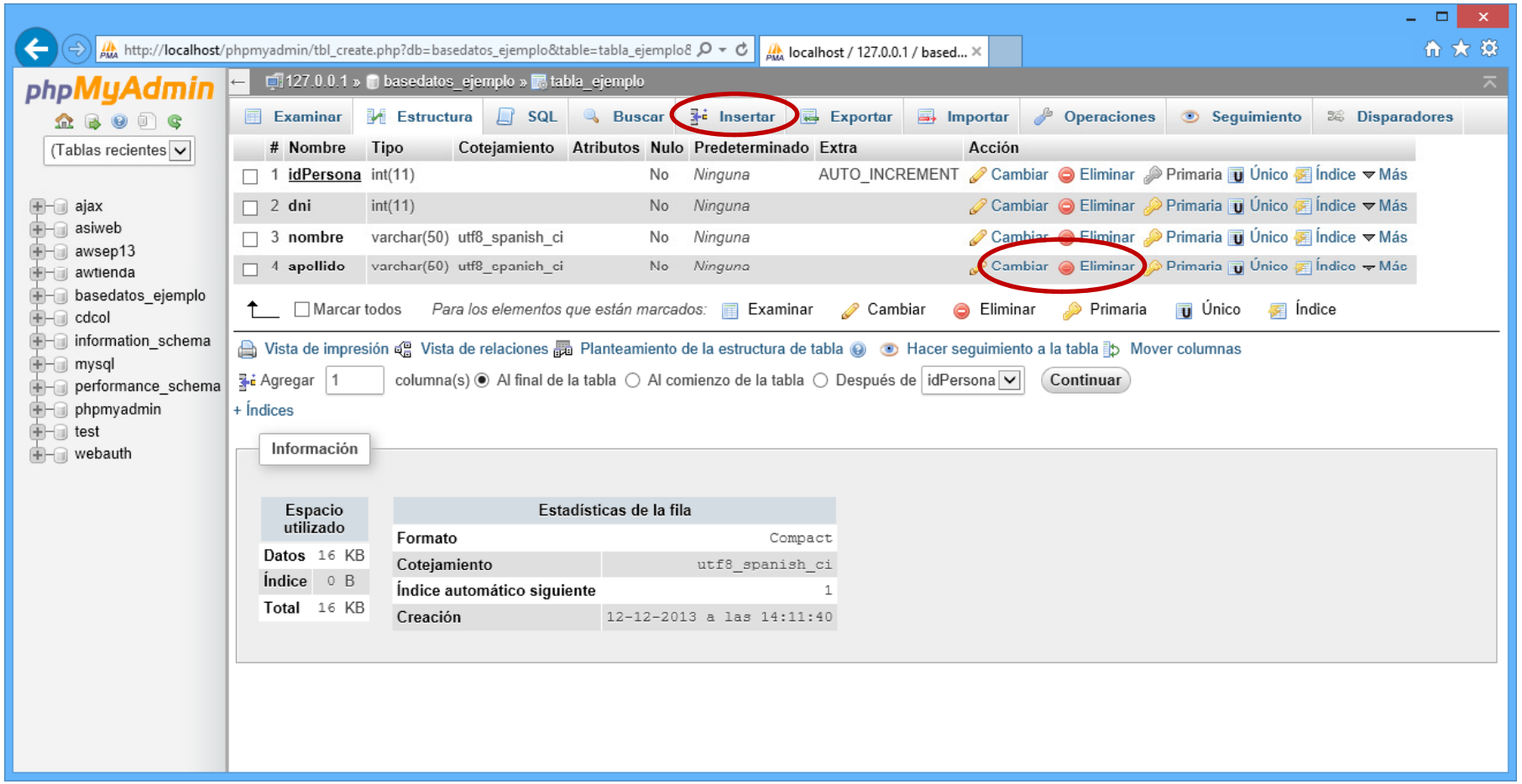
➢ Añadir usuario y contraseña (3)

✓ Escogemos un nombre de usuario, el servidor local (localhost) y una contraseña (en este caso se ha utilizado el botón “Generar”). Más abajo se otorgan todos los privilegios para la base de datos seleccionada y todos los privilegios globales:



➢ Gestión de la tabla desde phpMyAdmin

✓ Dentro de la tabla se pueden insertar, modificar y eliminar registros:



Índice: Tema 4.2

PHP y MySQL (1)

- PHP dispone de numerosas funciones para interaccionar con las bases de datos de MySQL. Algunas de ellas son las siguientes:

  - **`identificador = mysql_connect(servidor, usuario, contraseña)`**

    Establece la conexión con el servidor de bases de datos indicado utilizando un usuario y contraseña registrado en MySQL.

    Cuando el servidor MySQL y el servidor Web se encuentran en el mismo equipo el servidor es **`localhost`**. Este es el valor por defecto si no se indica nada.

    La función devuelve un valor entero que se utiliza como el identificador de la conexión en posteriores operaciones. Si se produce un error devuelve 0.

  - **`mysql_close(identificador)`**

    La conexión se cierra automáticamente al terminar la ejecución del script en el que se ha establecido, o bien explícitamente con esta función.

PHP y MySQL (2)

- `identificador = mysql_pconnect(servidor, usuario, contraseña)`

  Establece una conexión persistente, es decir, que no será cerrada al terminar la ejecución del script en el que se ha establecido.

  De esta forma la conexión permanece abierta y puede ser utilizada en los scripts de otras páginas que se ejecuten posteriormente.

- `mysql_pclose(identificador)`

  Se utiliza para cerrar de forma explícita una conexión persistente.

- `mysql_error()`

  Devuelve un texto con el mensaje de error de la última operación MySQL.

- `die(mensaje)`

  Muestra el mensaje indicado y aborta la ejecución del programa.

PHP y MySQL (3)

- **`mysql_selectdb(nombre_base_datos, identificador)`**

  Se utiliza para seleccionar la base de datos sobre la que llevar a cabo consultas.

  Además se pasa un segundo parámetro para identificar la conexión al servidor de bases de datos en el que se encuentra la base de datos a la que se desea conectar. Dicho parámetro es opcional y, si no se especifica, su valor es el de la última conexión abierta.

- **`recurso = mysql_query(consulta, identificador)`**

  Ejecuta la consulta especificada como una sentencia SQL sobre la base de datos activa en la conexión al servidor de bases de datos especificado con el identificador (opcional).

  Ejemplos de posibles consultas:

```
SELECT * FROM tabla
SELECT columna FROM tabla WHERE condicion
INSERT INTO tabla (columna1, columna2) VALUES (valor1, valor2)
UPDATE tabla SET columna2=valor3 WHERE columna1=valor1
DELETE FROM tabla WHERE columna1=valor1
```

PHP y MySQL (4)

En el caso de consultas tipo SELECT, la función mysql_query() devuelve un conjunto de registros en caso de éxito y FALSE en caso contrario.

Este conjunto de registros se almacena en una estructura de datos conocida como recurso, a partir de la cual se pueden extraer los registros correspondientes mediante las funciones siguientes:

- `registro = mysql_fetch_row(recurso)`

  Devuelve un array numérico que corresponde al registro recuperado y mueve el puntero de datos interno hacia delante. Así, la primera columna del registro se encuentra en la posición 0 de dicho array, la segunda en la 1, etc…

- `registro = mysql_fetch_assoc(recurso)`

  Devuelve un array asociativo que corresponde al registro recuperado y mueve el puntero de datos interno hacia adelante. En este caso el nombre de las columnas se puede utilizar como clave de dicho array para acceder a su valor.

➢ Ejemplo de utilización

✓ Se va a diseñar una aplicación web desde la que poder insertar, consultar, modificar y eliminar registros en la tabla "**tabla_ejemplo**" de la base de datos "**basedatos_ejemplo**" creada en el apartado anterior con phpMyAdmin:

Conexión a la base de datos

- ✓ Es una práctica común crear un fichero “configuración.php” que contenga los datos de configuración como el servidor, usuario y contraseña.
- ✓ Además, se pueden colocar aquí también el nombre de la base de datos, de la tabla o tablas a utilizar, y el identificador de la conexión (abierta de forma persistente) para ser utilizado en el resto de las páginas de la aplicación:

```
<?php
 // Datos de configuración
    $servidorBD = "localhost";
    $usuarioBD  = "aherrang";              // Si no se ha creado: "root"
    $passwordBD = "5748fT8Ss9GCFExJ";      // Si no se ha creado: ""
 // Datos para ser usados en el resto de las páginas
    $identificadorBD = mysql_pconnect($servidorBD,$usuarioBD,$passwordBD)
                       or die("Error: " . mysql_error());
    $nombreBD = "basedatos_ejemplo";
    $tablaBD  = "tabla_ejemplo";
?>
```

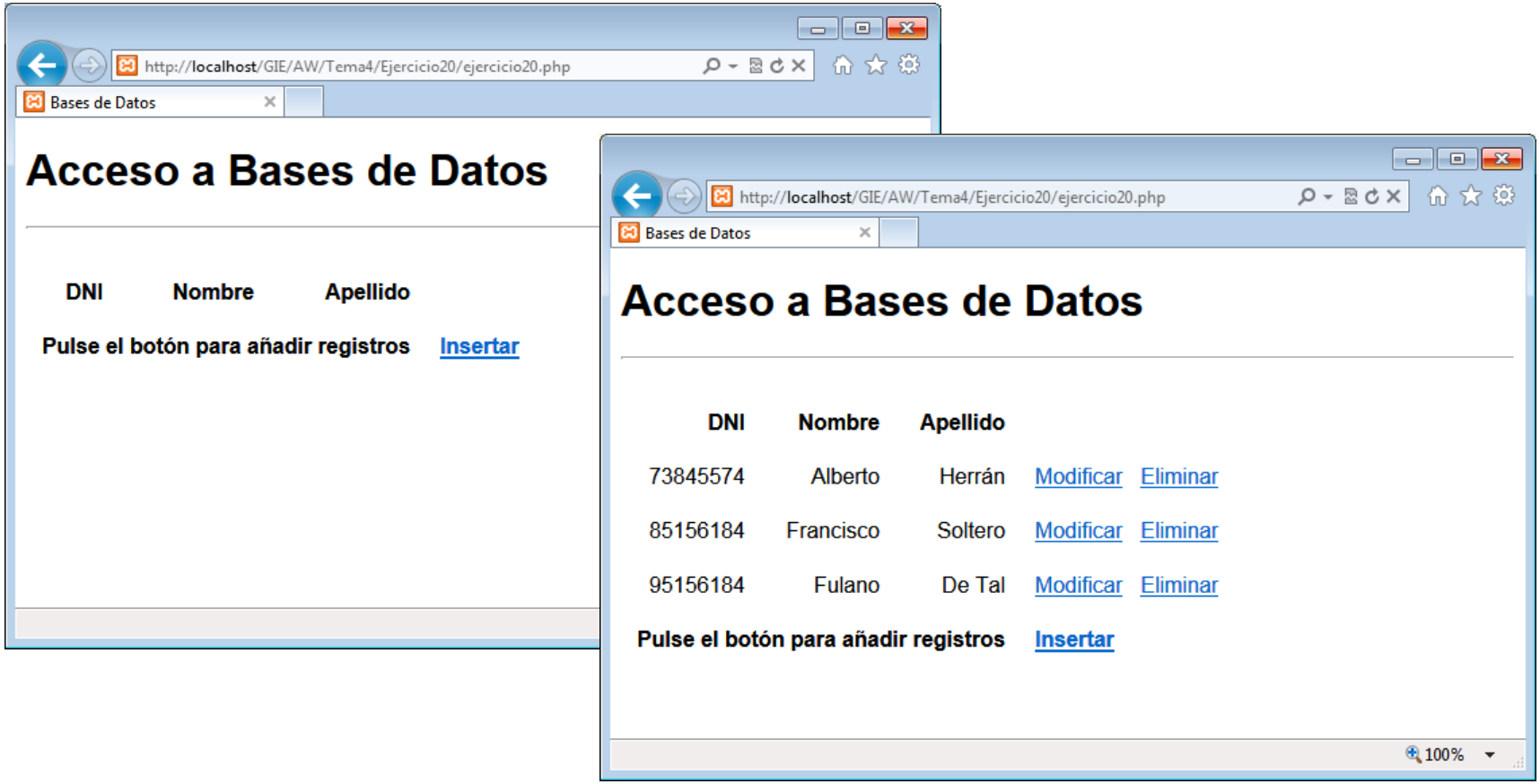
Consulta de tablas (1)

- La página "**ejercicio15.php**" muestra una tabla con los registros de la tabla y un enlace para insertar nuevos registros. Además, al lado de cada registro se han añadido enlaces para modificar y eliminar el registro correspondiente:

```
<?php
    require_once("configuracion.php");
    mysql_select_db($nombreBD,$identificadorBD);
    $consultaBD  = "SELECT * FROM $tablaBD";
    $resultadoBD = mysql_query($consultaBD,$identificadorBD) or die("Error: ".mysql_error());
?>
// Ahora, dentro de la tabla que ubiquemos en el cuerpo de la página...
<?php while($registroBD = mysql_fetch_assoc($resultadoBD)) { ?>
    <tr align="right">
     <td> <?php echo $registroBD['dni']; ?>      </td>
     <td> <?php echo $registroBD['nombre']; ?>   </td>
     <td> <?php echo $registroBD['apellido']; ?> </td>
     <td> <a href="modificar.php?idPersona=<?php echo $registroBD['idPersona']; ?>">Modificar</a>
          <a href="eliminar.php?idPersona=<?php echo $registroBD['idPersona']; ?>">Eliminar</a>
     </td>
    </tr>
<?php }; ?>
<p>Pulse el botón para añadir registros <a href="insertar.php">Insertar</a></p>
```

Consulta de tablas (2)

- Originalmente la tabla se encuentra vacía mostrándose únicamente las cabeceras de las columnas y el botón que nos lleva a la página “insertar.php”:



Inserción de registros (1)

- La página "**insertar.php**" inserta nuevos registros en la tabla. Para ello, ofrece un formulario desde el que introducir los datos y conecta con la base de datos para añadir un nuevo registro con los mismos:

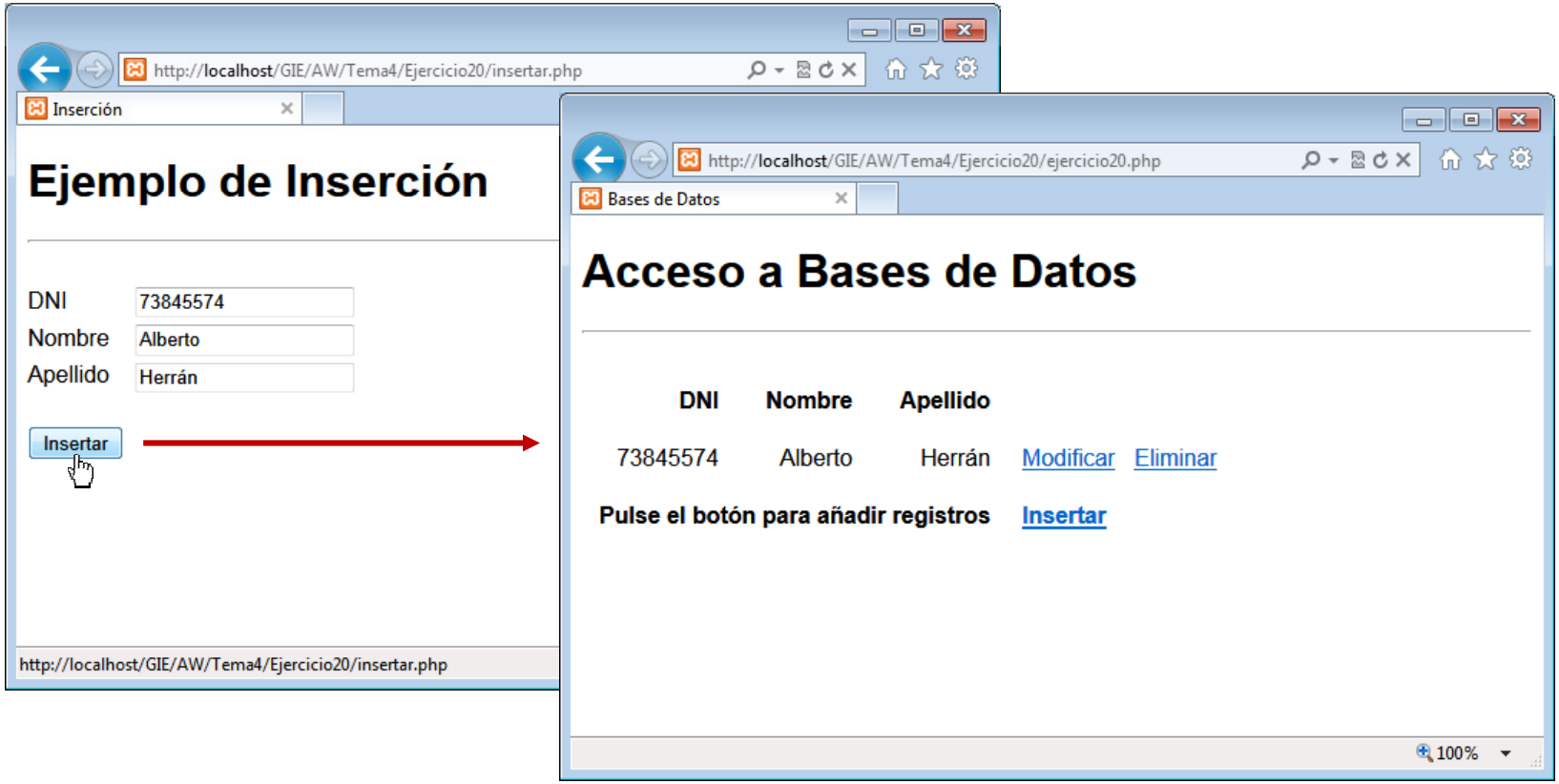
```
<?php
   if(isset($_POST['insertar'])) {
      require_once("configuracion.php");
      mysql_select_db($nombreBD,$identificadorBD);
      $consultaBD = "INSERT INTO $tablaBD (dni,nombre,apellido) "
                  . "VALUES ('{$_POST['dni']}','{$_POST['nombre']}','{$_POST['apellido']}')";
      mysql_query($consultaBD,$identificadorBD) or die("Error:" .mysql_error());
      header("Location: ejercicio15.php");
   }
?>

// Ahora, dentro del cuerpo de la página...

<form action="<?php echo($_SERVER["PHP_SELF"])?>" method="post">
   DNI:      <input type="text"   name="dni" />
   Nombre:   <input type="text"   name="nombre"   maxlength="50" />
   Apellido: <input type="text"   name="apellido" maxlength="50" />
   <input type="submit" name="insertar" value="Insertar">
</form>
```

Inserción de registros (2)

- Al pulsar insertar, los datos son enviados mediante POST a la propia página para insertarlos en la base de datos. Tras insertar los datos se vuelve a la página "ejercicio15.php" que lista la tabla con el nuevo registro:



Modificación de registros (1)

- La página "**modificar.php**" carga los datos del registro a modificar en un formulario para poder modificarlos y reenviarselos a si misma. Una vez recibidos se utilizan para actualizar el registro correspondiente en la base de datos:

```
// Desde el enlace Modificar de la página ejercicio15.php
<a href="modificar.php?idPersona=<?php echo $registroBD['idPersona']; ?>">Modificar</a>

// Ahora, en la página modificar.php
<?php
    require_once("configuracion.php");
    mysql_select_db($nombreBD,$identificadorBD);
    if(!isset($_POST["modificar"])) { // Consulta para cargar los datos en un formulario
      $consultaBD  = "SELECT * FROM $tablaBD WHERE idPersona='{$_GET['idPersona']}'";
      $resultadoBD = mysql_query($consultaBD,$identificadorBD) or die("Error: ".mysql_error());
      $registroBD  = mysql_fetch_assoc($resultadoBD);
    } else { // Tras pulsar modificar se actualizan los datos con los valores del formulario
      $consultaBD = "UPDATE $tablaBD SET dni='{$_POST['dni']}', nombre='{$_POST['nombre']}', "
                  . "apellido='{$_POST['apellido']}' WHERE idPersona='{$_GET['idPersona']}'";
      mysql_query($consultaBD,$identificadorBD) or die("Error:" .mysql_error());
      header("Location: ejercicio15.php");
    }
?>
```

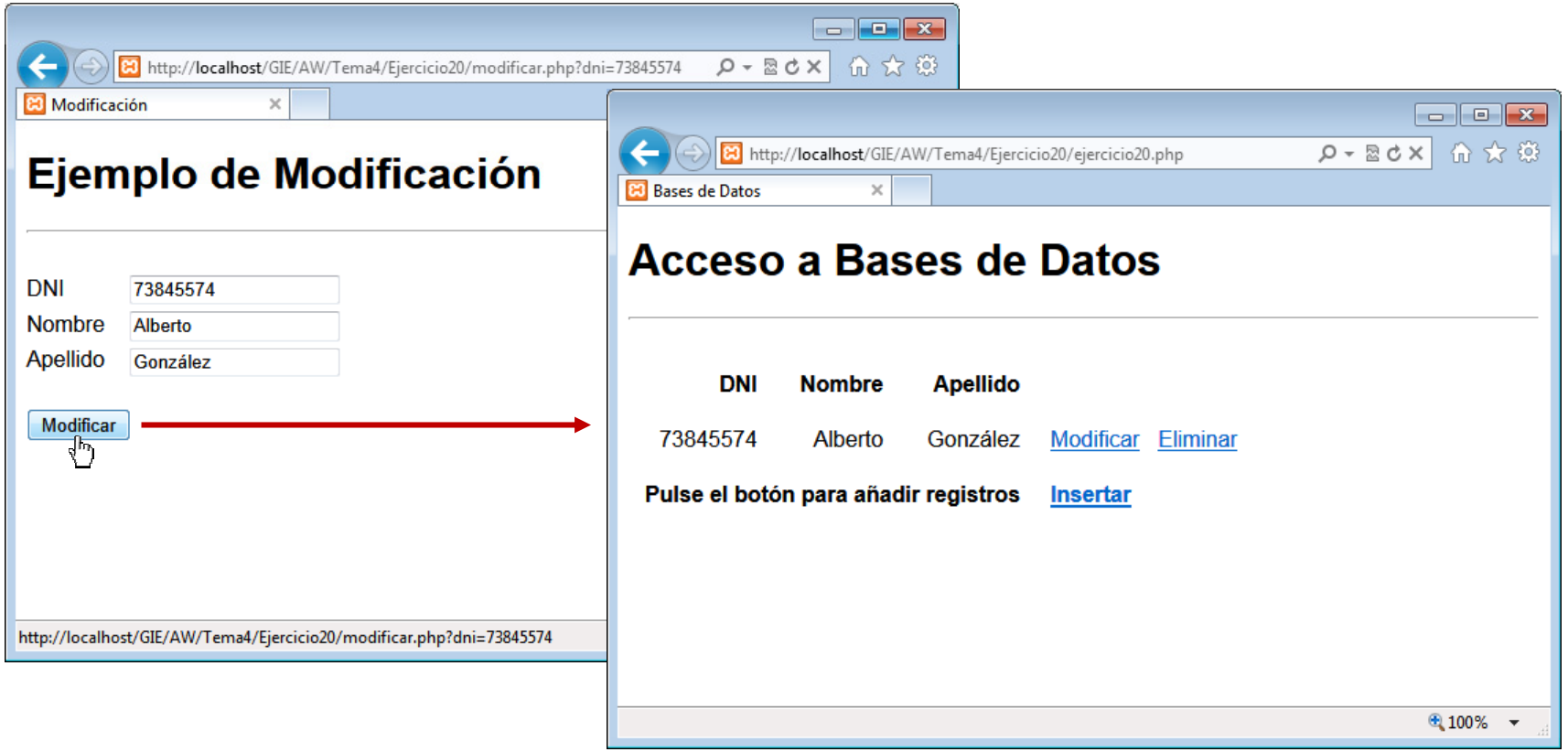
Modificación de registros (2)

- A continuación se muestra el formulario desde el que actualizar los datos en la página "**modificar.php**":

```
<form action="modificar.php?idPersona=<?php echo $_GET['idPersona']; ?>" method="post">
  DNI:
  <input type="text" name="dni"      value="<?php echo $registroBD['dni']; ?>"/>
  Nombre:
  <input type="text" name="nombre"   value="<?php echo $registroBD['nombre']; ?>"/>
  Apellido:
  <input type="text" name="apellido" value="<?php echo $registroBD['apellido']; ?>"/>
  <input type="submit" name="modificar" value="Modificar">
</form>
```

Modificación de registros (3)

- Al pulsar modificar, los datos son enviados mediante POST a la propia página para actualizarlos en la base de datos. Tras insertar los datos se vuelve a la página "ejercicio15.php" que lista la tabla con el nuevo registro modificado:



Borrado de registros (1)

- La página "**eliminar.php**" elimina el registro cuyo dni es el recibido mediante GET de la base de datos:

```
// Desde el enlace Eliminar de la página ejercicio15.php
<a href="eliminar.php?idPersona=<?php echo $registroBD['idersona']; ?>">Eliminar</a>

// Ahora, en la página eliminar.php
<?php
  require_once("configuracion.php");
  mysql_select_db($nombreBD,$identificadorBD);
  $consultaBD = "DELETE FROM $tablaBD WHERE idPersona='{$_GET['idPersona']}'";
  mysql_query($consultaBD,$identificadorBD) or die("Error: ".mysql_error());
  header("Location: ejercicio15.php");
?>
```

➢ Borrado de registros (2)

- ✓ Al pulsar eliminar, el dni del registro a eliminar es enviado mediante GET a la página “eliminar.php” que lo elimina de la base de datos y devuelve el control a la página “ejercicio15.php” que lista la tabla sin el nuevo registro: