C6 – MANTENIMENT DE PORTALS D'INFORMACIÓ

Unitat didàctica 4: PHP. Llenguatges script **de servidor**

Índex

1. Introducció....3
   1.1 Petita història de PHP....3
   1.2 Què és PHP?....3
   1.3 Què es necessita per a què funcioni PHP?....3
2. Programació en PHP....4
   2.1 Introducció....4
      2.1.1 Incloure PHP a les pàgines HTML....4
      2.1.2 Separació de les instruccions....4
      2.1.3 Comentaris a PHP....4
   2.2 Variables....4
      2.2.1 Introducció....4
      2.2.2 Tipus de dades....5
      2.2.3 Interpretació de les cometes....5
      2.2.4 Operador de concatenació de cadenes....6
      2.2.5 Longitud d'una cadena....6
      2.2.6 Caràcters protegits....6
   2.3 Operadors....7
      2.3.1 Operadors i les seves categories....7
      2.3.2 Exemples d'operadors....8
   2.4 Constants....8
      2.4.1 Definició i ús de Constants....8
      2.4.2 Constants predefinides....9
   2.5 Estructures (sentències) de control....9
      2.5.1 Estructures de decisió....9
      2.5.2 Estructures iteratives....11
   2.6 Arrays (taules)....13
      2.6.1 Arrays clàssics o convencionals....13
      2.6.1 Arrays associatius....15
      2.6.2 Arrays multidimensionals....16
   2.7 Funcions d'usuari....16
   2.8 Funcions de cadena....17
3. Tractament de formularis HTML amb PHP....18
   3.1 Introducció....18
   3.2 Comportament de PHP en rebre un formulari....19
      3.2.1 Caixes de text....20
      3.2.2 Àrees de text....20
      3.2.3 Caselles de verificació o checkboxes....21
      3.2.4 Botons de ràdio o radiobuttons....22
      3.2.5 Menús desplegables o comboboxes....22
      3.2.6 Camps ocults....23
      3.2.7 Exemple d'accés als elements d'un formulari amb PHP....24
4. Sessions web....26
   4.1 Introducció....26
   4.1 Configuració de php.ini per a que funcionin les sessions....26
   4.3 Inicialització d'una sessió....27
   4.4 Variables de sessió....27
   4.5 Finalitzar una sessió....28
   4.6 Error comú amb sessions....28
   4.7 Exemple de sessions....28

4.8 Carro de la compra....29
array_search:....29
array_splice:....29
exit:....30
header:....30
foreach:....30
5. PHP amb bases de dades MySQL....32
5.1 Creació de bases de dades y taules....32
5.2 Funcions de connexió i desconnexió a la base de dades....33
5.3 Selecció d'una base de dades....33
5.4 Funció genèrica de connexió i selecció de base de dades....34
5.5 Manipulació de les dades de la base de dades....34
5.6 Inserir registres a la base de dades....34
5.7 Seleccionar registres de la base de dades....36
5.8 Modificar registres de la base de dades....38
5.9 Esborrar registres de la base de dades....40
5.10 Paginació de resultats....41
5.11 Pujar fitxers al servidor....43
5.11.1 Configuració HTML....43
5.11.2 Configuració PHP – Servidor....44
5.11.3 Gestió completa de fitxers....47
5.12 Identificació d'usuaris....50

1. Introducció

1.1 Petita història de PHP.

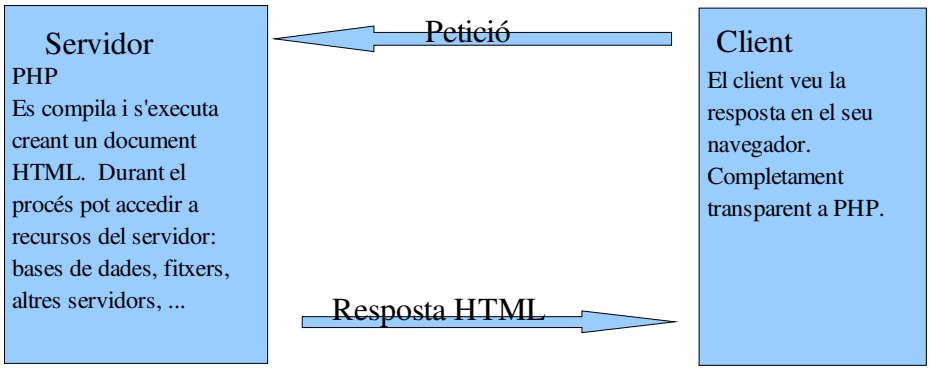
PHP és un llenguatge creat per una comunitat de programadors. Es va desenvolupar originariament l'any 1994 per Rasmus Lerdof com un CGI (Common Gateway Interface) escrit en C que permetia la interpretació d'un nombre limitat de paràmetres. El sistema es va anomenar Personal Home Page Tools (PHP Tools) i va tenir cert èxit gràcies a que d'altres persones van demanar Lerdof que els deixés fer servir el seu programa per a les seves pàgines.

A mitjans de 1997 es va tornar a programar l'analitzador sintàctic, es van incloure noves funcionalitats i es van asentar les bases per a la creació del PHP3. Actualment la versió de PHP en ús és la 5.2.6 les característiques de la qual són:

- Velocitat: gràcies a que primer es compila i després s'executa.
- Independència el servidor: gràcies a la creació de versions natives per a més servidors.
- API (Application Programing Interface) més elaborada i amb més funcions.

1.2 Què és PHP?

El llenguatge PHP és un llenguatge de programació d'estil clàssic. PHP s'executa al servidor, per això ens permet accedir als recursos que tingui al servidor com per exemple una base de dades. El resultat de l'execució de PHP al servidor s'envia al navegador de l'usuari. És imprescindible que el servidor suporti PHP.



1.3 Què es necessita per a què funcioni PHP?

Per a fer servir PHP necessitarem:

- Versió compilada de PHP (http://www.php.net)
- Un servidor web (Apache, IIS, ...)

Si es vol fer servir PHP amb algun tipus d'accés a base de dades es recomana MySQL Server ja que PHP ofereix suport natiu per a aquest motor de base de dades.

Per últim, l'entorn de feina més adequat per a la utilització de PHP serà el IDE de desenvolupament ECLIPSE amb el plugin phpEclipse.

2. Programació en PHP

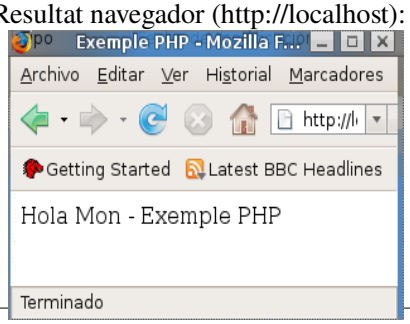
2.1 Introducció

2.1.1 Incloure PHP a les pàgines HTML

Una vegada instal·lat el servidor i el mòdul PHP podem començar a fer programes en aquest llenguatge. El primer que hem de saber és com incloure codi PHP a dins d'una pàgina HTML. Veiem un exemple que ens ho ensenyi.

```
<html>
<head>
<title>Exemple PHP</title>
</head>
<body>
<?php
echo “Hola Mon – Exemple PHP”;
?>
</body>
</html>
```

Resultat navegador (http://localhost):



Com es pot veure el que fem és escriure codi HTML amb cert **codi PHP embegut** en el mateix, que produirà certa sortida (en el nostre exemple produir text). El codi PHP s'inclou entre **etiquetes especials** de començament i acabament que són `<?php` i `?>`.

Els fitxers que contindran codi PHP hauran de tenir l'extensió .php per tal que el servidor els reconeixi com a tals i els processi adequadament. Evidentment haurem d'**accedir a aquests fitxers a través del navegador NO directament obrint-los**. Penseu que si obrim aquests fitxers directament i no ho fem a través del servidor web el codi PHP no s'interpreta.

Podem obrir i tancar les etiquetes anteriors tantes vegades com vulguem durant el desenvolupament de l'HTML.

2.1.2 Separació de les instruccions

Les instruccions es separen igual que a C o Pascal, **finalitzant cada sentència amb el caràcter punt i coma obligatòriament**. Si no poseu el caràcter ; el codi PHP no funciona.

2.1.3 Comentaris a PHP

Hi han dues formes d'introduir comentaris al codi PHP. Aquestes dues formes són:

- Comentaris d'una única línia: Es fan amb els caràcters `//`.
- Comentaris de més d'una línia: Es fan amb els caràcters d'obertura `/*` i de tancament `*/`.

2.2 Variables

2.2.1 Introducció

Una variable és un element al qual li donem un nom i li atribuïm un determinat tipus d'informació. Són la base de la programació en qualsevol llenguatge de programació. En PHP tenen les següents característiques:

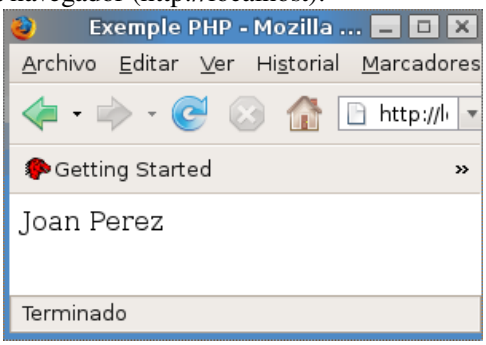
- No existeix una paraula reservada amb la missió de permetre la declaració de variables, per tant, quan volem fer servir una variable la inicialitzem i queda, implícitament, declarada.

- Les variables es representan amb un caràcter `$` seguit del nom de la variable.
- El nom de les variables és sensible a majúscules i minúscules (per tant la variable `$hola` és diferent de la variable `$Hola`)

Exemple:

```
<html>
<head>
<title>Exemple PHP</title>
</head>
<body>
<?php
$nom = “Joan Perez”;
echo $nom;
?>
</body>
</html>
```

Resultat navegador (http://localhost):



2.2.2 Tipus de dades

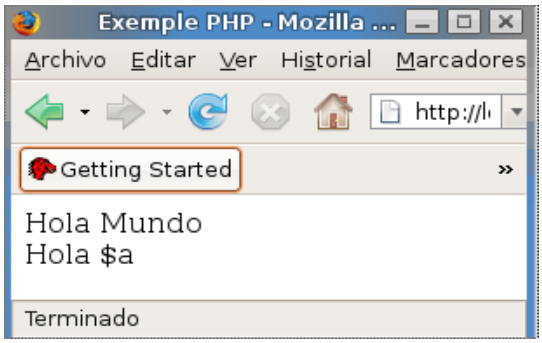
Com que a PHP les variables no es declaren no existeixen tipus de dades explícites. Encara que això sigui així, internament, PHP reserva la memòria necessària segons el tipus de dada implícit amb el que inicialitzem la variable. Tenim els següent tipus de dades:

1. **Enters:** Els nombres enters es poden especificar fent servir la següent sintaxis:
   - `$a = 1234;` // Representa un nombre decimal
   - `$a = -1234;` // Representa un nombre decimal negatiu
   - `$a = 0123;` // Representa un nombre octal (equival al 83 decimal)
   - `$a = 0x123;` // Representa un nombre hexadecimal (equival al 291 decimal)
2. **Nombres en punt flotant:** Els nombres en punt flotant es poden especificar fent servir alguna de les següent sintaxis:
   - `$a = 1.234;` // Representa un nombre en punt flotant.
   - `$a = 1.2e3;` // Representa un nombre en punt flotant en format exponencial.
3. **Cadenes de text:** Les cadenes de text es poden definir fent servir cometes dobles o simples. Per exemple:
   - `$a = “Hola”;` // Representa una cadena de text (definida amb cometes dobles).
   - `$a = 'Mundo';` // Representa una cadena de text (definida amb cometes simples).
4. **Dada booleana:** Són aquelles variables que emmagatzemen només dos valors, vertader (true) o fals (false). Per a crear-les fem servir la següent sintaxis:
   - `$a = true;` // Representa una variable booleana de valor vertader.
   - `$a = false;` // Representa una variable booleana de valor false.

2.2.3 Interpretació de les cometes

Encara que una cadena de text es pugui definir amb cometes dobles o simples, en el moment de mostrar una informació sobre la pàgina HTML, PHP tracta de manera diferent les cometes dobles i simples. Veiem-ho a partir d'un exemple:

```
<html>
<head>
<title>Exemple PHP</title>
</head>
<body>
<?php
$a = “Mundo”;
echo “Hola $a<br>”;
echo 'Hola $a';
?>
</body>
</html>
```



Resultat navegador (http://localhost):

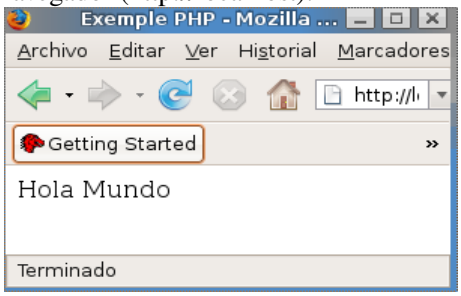
Com es pot veure, la comanda `echo “Hola $a”` mostra `Hola Mundo` mentre que la comando `echo 'Hola $a'` mostra `Hola $a`. Això es deu a que, quan mostrem una cadena, PHP interpreta el contingut si la cadena es troba entre cometes dobles i no ho fa si es troba entre cometes simples.

2.2.4 Operador de concatenació de cadenes

L'únic operador de cadenes que existeix és el de concatenació, el **punt**. La millor forma d'entendre el funcionament d'aquest operador és a través d'un exemple.

```
<html>
<head>
<title>Exemple PHP</title>
</head>
<body>
<?php
$a = “Hola”;
$b = “Mundo”;
echo $a . “ “ . $b;
?>
</body>
</html>
```

Resultat navegador (http://localhost):



2.2.5 Longitud d'una cadena

Per a saber la longitud d'una cadena hem de fer servir la funció `strlen()`. Ho farem de la següent manera:

```
<?php
$a = “Hola Mundo”;
echo “Longitud=“ . strlen($a);
?>
```

Resultat navegador (http://localhost):

2.2.6 Caràcters protegits

Igual que en llenguatges de programació com C o Perl, el caràcter de barra invertida (\) es fa servir per a especificar caràcters especials. Un llistat de caràcters especials són:

`\n`	Nova línia.
`\r`	Retorn de carro.
`\t`	Tabulació horitzontal.
`\\`	Barra invertida.
`\$`	Símbol del dólar.

\"	Cometes dobles.
\'	Cometes simples.

2.3 Operadors

2.3.1 Operadors i les seves categories

Com tots els llenguatges de programació a PHP hi ha diferents tipus d'operadors que ens permeten formar expressions. Els operadors es poden classificar en els següents conjunts:

- Operadors d'assignació:

`$a = $b`	Assigna a `$a` el contingut de `$b`
`$a += $b` `$a = $a + $b`	Li suma `$b` a `$a`.
`$a -= $b` `$a = $a - $b`	Li resta `$b` a `$a`.
`$a *= $b` `$a = $a * $b`	Multiplica `$a` per `$b` i ho assigna a `$a`
`$a /= $b` `$a = $a / $b`	Divideix `$a` per `$b` i ho assigna a `$a`
`$a %= $b` `$a = $a % $b`	Assigna a `$a` el mòdul de la divisió entera de `$a` per `$b`
`$a .= $b` `$a = $a . $b`	L'operador . és l'operador de concatenació de cadenes. Així dons aquesta expressió afegeix la cadena `$b` a la cadena `$a`

- Operadors lògics:

`$a && $b`	Veritable si tots dos són veritables.
`$a || $b`	Veritable si algun dels dos és veritable.
`!$a`	Veritable si `$a` és fals.

- Operadors de comparació:

`$a < $b`	`$a` és més petit que `$b`
`$a > $b`	`$a` és més gran que `$b`
`$a <= $b`	`$a` és més petit o igual que `$b`
`$a >= $b`	`$a` és més gran o igual que `$b`
`$a == $b`	`$a` és igual a `$b`
`$a != $b`	`$a` és diferent a `$b`

2.3.2 Exemples d'operadors

Veiem alguns exemples d'ús dels operadors:

- Exemple 1: Càlcul de l'àrea d'un rectangle.

```
<?php
$base = 15;
$altura = 12;
$area = ($base * $altura) / 2;
echo “Area = $area”;
?>
```

- Exemple 2: Donats dos números calcularem la suma, la resta, la divisió, la multiplicació i el mòdul de la divisió entera.

```
<?php
$num1 = 8;
$num2 = 5;
$suma=$num1+$num2;
$resta=$num1-$num2;
$mult=$num1*$num2;
$div=$num1/$num2;
$mod=$num1%$num2;
echo “Suma = $suma”;
echo “Resta = $resta”;
echo “Multiplicació = $mult”;
echo “Divisió = $div”;
echo “Mòdul = $mod”;
?>
```

- Exemple 3: Càlcul del salari d'un treballador amb un impost.

```
<?php
$salari = 3500;
$impost = 20; // En percentatge
$salariReal = $salari - (($salari/100)*$impost);
echo “Sou brut = $salari“;
echo “Sou net = $salariReal“;
?>
```

2.4 Constants

2.4.1 Definició i ús de Constants

PHP proporciona un mecanisme per a definir constants en temps d'execució. Les constants emmagatzemen valors com les variables però no poden ser modificades més tard amb un altre valor o eliminades. Només es poden definir com a constants valors escalars, no arrays, com els booleans, enters, punts flotants o cadenes de text. Les constants es defineixen amb la funció `define()` amb la següent sintaxi:

```
define(“NOM_CONSTANT”,VALOR_CONSTANT);
```

Per exemple:

- Definició de la constant PI: `define(“PI”,3,141516);`

```
<?php
define(“PI”,3.141516);
echo PI;
?>
```

- Definició d'una cadena de text: `define("NOM","Joan");`

```
<?php
define("NOM","Joan");
echo NOM;
?>
```

Les diferències entre constants i variables són les següents:

- Les constants no van precedides pel caràcter $.
- Les constants només es poden definir fent servir la funció `define()`.
- Les constants no poden ser redefinides o eliminades.
- Les constants només poden contenir valors escalars, no arrays.

Per altre banda, les constants, igual que les variables, són sensibles a majúscules i minúscules.

2.4.2 Constants predefinides

PHP defineix un conjunt de constants per defecte que es troben sembre a disposició de l'usuari. Les més comunes són:

- `TRUE` – Valor veritable. Si fem echo TRUE; ens mostra 1.
- `FALSE` – Valor fals. Si fem echo FALSE; no mostra res.
- `M_PI` – És la constant 3.142592653589835.
- `E_WARNING` – Denota una condició a on PHP reconeix que hi ha alguna cosa errònia, pero continuarà de totes formes.
- `E_PARSE` – L'intèrpret va trobar sintaxis invàlida a l'arxiu de comandes. La recuperció no és possible.
- `PHP_VERSION` – Conté la versió de PHP que estem fent servir.

2.5 Estructures (sentències) de control

Les sentències de control permeten executar un bloc de codis depenent de unes condicions. A PHP, la definició de veritable i fals és igual que al llenguatge C: el 0 equival a fals i qualsevol altre número equival a veritable.

2.5.1 Estructures de decisió

- `if ... else`: La sentencia `if ... else` permet executar un bloc d'instruccions si la condició és veritable i un altre bloc si la condició és falsa. La condició ha d'estar, obligatòriament, tancada entre parèntesis. La sintaxi d'aquesta estructura és:

```
if (condicio) {
   // Bloc que s'executa si la condició és certa
} else {
   // Bloc que s'executa si la condició és falsa
}
```

Exemple 1:

```
if ($a > 4) {
   echo "La variable $a és més gran que 4";
} else {
   echo "La variable $a és més petita o igual que 4";
}
```

Exemple 2:

```
if ($nombre == “”) {
   echo “No tens nom”;
} else {
   echo “El teu nom és $nombre”;
}
```

- `if ... elseif ... else`: La sentencia `if ... elseif ... else` permet executar vàries condicions en cascada en lloc d'una de sola. Les condicions han d'estar, obligatòriament, tancades entre parèntesis. La sintaxi d'aquesta estructura és:

```
if (condicio1) {
   // Bloc que s'executa si la condicio1 és certa
} elseif (condicio2) {
// Bloc que s'executa si la condicio2 és certa
}...
...
} elseif (condicioN) {
// Bloc que s'executa si la condicioN és certa
} else {
   // Bloc que s'executa si cap de les condicions
   // anteriors és certa
}
```

Exemple:

```
if ($nombre == “”) {
   echo “No tens nom”;
} elseif (($nombre == “eva”) || ($nombre == “Eva”)) {
   echo “El teu nom és EVA”;
} else {
   echo “El teu nom és ” . $nombre;
}
```

- `switch ... case ... default`: Aquesta instrucció és una alternativa al `if ... elseif ... else` en el cas que les condicions siguin d'igualtat sobre només una variable. La seva sintaxi és:

```
switch($VARIABLE) {
   case VALOR1:
      // Instruccions si VARIABLE = VALOR1
      [break;]
   case VALOR2:
      // Instruccions si VARIABLE = VALOR2
      [break;]
   ...
   ...
   case VALORn:
      // Instruccions si VARIABLE = VALORn
      [break;]
   default:
      // Instruccions si VARIABLE és diferent
      // de tots els valors anteriors
}
```

El que fem es avaluar la `$VARIABLE` a cada `case`. Si la variable coincideix amb el valor del case llavors s'executa el conjunt d'instruccions associades a aquest. La sentència `break` permet evitar que el `switch` no segueixi buscant a la llista.

Exemple:

```
switch ($dia) {
   case “Dilluns”:
      echo “És dilluns”;
      break;
   case “Dimarts”:
      echo “És dimarts”;
      break;
   case “Dimecres”:
      echo “És dimecres”;
      break;
   case “Dijous”:
      echo “És dijous”;
      break;
   case “Divendres”:
      echo “És divendres”;
      break;
   default:
      echo “És cap de setmana”;
}
```

2.5.2 Estructures iteratives

- `while`: La sentència `while` executa un bloc de codi mentre es compleixi una determinada condició. La seva sintaxi és:

```
while(condicio) {
   // Sentències que s'executen mentre la
   // condició sigui certa
}
```

Exemple:

```
$num=1;
while ($num < 10) {
   echo $num . “<br>”;
   $num++;
}
```

Podem trencar el bucle `while` fent servir `break`.

Exemple:

```
$num=1;
while ($num < 10) {
   echo $num . “<br>”;
   if ($num == 6) {
      echo “Finalitzem”;
      break;
   }
   $num++;
}
```

Podem continuar el bucle `while` sense fer l'acció fent servir `continue`.

Exemple:

```
$num=1;
while ($num < 10) {
   if ($num == 6) {
```

```
      $num++;
      continue;
   } else {
      echo $num . “<br>”;
      $num++;
   }
}
```

- `do ... while`: Aquesta sentència és similar a `while`, excepte que amb aquesta sentència primer executem el bloc de codi i després avaluem la condició, per la qual cosa el bloc de codi s'executa sempre, com a mínim, una vegada. La seva sintaxi és:

```
do {
   // Sentències que s'executen mentre la
   // condició sigui certa
} while(condicio)
```

Exemple:

```
$num=1;
do {
   echo $num . “<br>”;
   if ($num == 7) {
      echo “Sortim”;
      break;
   }
   $num++;
} while ($num < 10)
```

Com es pot veure, al igual que el `while`, es poden fer servir les sentències `break` i `continue`.

- `for`: Aquesta sentència es pot substituir perfectament per un `while` però resulta útil quan hem d'executar un bloc de codi a condició que una variable es trobi entre un valor mínim i un valor màxim. El cicle també es pot trencar amb la sentència `break` i `continue` i la seva sintaxi és:

```
for ($variable=valor_inicial; condicio; increment) {
   // Sentències que s'executen mentre la
   // condició sigui certa
   // La variable va canviant de valor
}
```

Exemple 1:

```
for ($num=1; $num <= 5; $num++) {
   if ($num == 3) {
      continue;
   }
   echo $num;
}
```

Exemple 2:

```
for ($num=1; $num <= 100; $num=$num+2) {
   echo $num;
}
```

Exemple 3:

```
for ($num=100; $num >= 0; $num=$num-3) {
```

```
   echo $num;
}
```

2.6 Arrays (taules)

Les taules o arrays són molt importants a PHP ja que, generalment, les funcions que retornen diversos valors, com les funcions lligades a les bases de dades, ho fan en forma d'array. Un array o taula no és res més que una variable que emmagatzema molts valors amb un únic nom. PHP, a diferència de la majoria de llenguatges de programació, disposa de dos tipus d'arrays:

2.6.1 Arrays clàssics o convencionals

Al igual que les variables, els arrays a PHP no es declaren i per tant els hem d'anar creant sobre la marxa. Veiem això a través de dos exemples:

Exemple 1:

```
$ciutat[0] = “París”;
$ciutat[1] = “Mèxic”;
$ciutat[2] = “Roma”;
$ciutat[3] = “Sevilla”;
$ciutat[4] = “Londres”;
echo “Jo visc a “ . $ciutat[3];
```

Exemple 2:

```
$ciutat[] = “París”; // Comença a numerar per 0
$ciutat[] = “Mèxic”;
$ciutat[] = “Roma”;
$ciutat[] = “Sevilla”;
$ciutat[] = “Londres”;
echo “Jo visc a “ . $ciutat[3];
```

Exemple 3:

```
$ciutat = array(“París”,“Mèxic”,”Roma”,“Sevilla”, “Londres”);
echo “Jo visc a “ . $ciutat[3];  // Jo visc a Sevilla
$poble = array(1=>“Vilanova”,“Calafell”,”Cunit”,“Vendrell”);
echo “Jo visc a “ . $poble[3];  // Jo visc a Cunit
```

Les característiques d'aquests arrays són les següents:

- Per a accedir a un element de l'array s'ha de fer servir l'índex (posició de l'element) i els corxets.
- A PHP els arrays són dinàmics i per tant, per a crear un array, podem fer servir la forma de l'exemple 2 a on no especifiquem l'índex.
- L'altra forma de crear un array és com es fa a l'exemple 3. Fent servir la funció `array()` i l'operador `=>` que ens permet indicar l'índex inicial si volem que sigui diferent de 0.
- Per a contar el nombre total d'elements que té un array farem servir la funció `count()`. Veiem-ho a través d'un exemple:

Exemple 4:

```
$ciutat = array(“París”,“Mèxic”,”Roma”,“Sevilla”, “Londres”);
echo “El nombre total d'elements de l'array és: “ . count($ciutat);
   // El nombre total d'elements de l'array és: 5
for ($i=0; $i<count($ciutat); $i++) {
   echo $ciutat[$i] . “<br>”;
}
```

- PHP admet l'assignació d'arrays, així com la comparació de variables array. Vegem-ho a través d'un exemple:

Exemple 5:

```
$matriu1 = array(1,2,3,4,5);
$matriu2 = array(1,2,3,4,6);
$matriu3 = $matriu1; // Assignació d'un array

for ($i=0; $i<count($matriu3); $i++) {
   echo $matriu3[$i] . “ ”;
} // Ens dóna: 1 2 3 4 5

if ($matriu1 == $matriu2) {  // Comparació d'un array
   echo “Iguals”;
} else {
   echo “Diferents”;
} // Dóna: Diferents

if ($matriu2 == $matriu3) {  // Comparació d'un array
   echo “Iguals”;
} else {
   echo “Diferents”;
} // Dóna: Diferents

if ($matriu1 == $matriu3) {  // Comparació d'un array
   echo “Iguals”;
} else {
   echo “Diferents”;
} // Dóna: Iguals
```

- Els arrays a PHP poden ser multitipus, això vol dir que poden contenir dades de diferents tipus. Per exemple:

Exemple 6:

```
$matriu1 = array(1,”Hola”,3.1416,TRUE);
echo $matriu1[0]; // Dóna: 1
echo $matriu1[1]; // Dóna: Hola
echo $matriu1[2]; // Dóna: 3.1416
echo $matriu1[3]; // Dóna: 1 (Valor per defecte de TRUE)
```

- Hi han dues funcions que ens permeten ordenar, tant alfabèticament com numèricament, els arrays. Aquests funcions són:
  - `sort()`: Aquesta funció ordenar un array. Si aquest està format per cadenes ho ordena alfabèticament i si està format per números ho ordena numèricament. Si l'array està format per cadenes de caràctes i números, primer apareixen les cadenes ordenades alfabèticament i després els números ordenats numèricament. Vegem-ho a través del següent exemple:

Exemple 7:

```
$lletres = array(“d”,”c”,”b”,”a”);
sort($lletres);
for ($i=0; $i<count($lletres); $i++) {
   echo $lletres[$i] . “ ”;
} // Dóna: a b c d

$nums = array(101,20,10,30);
sort($nums);
for ($i=0; $i<count($nums); $i++) {
   echo $lletres[$i] . " ";
} // Dóna: 10 20 30 101

$lletres = array("d","a",101,20);
sort($lletres);
```

```
for ($i=0; $i<count($lletres); $i++) {
   echo $lletres[$i] . " ";
} // Dóna: a d 20 101
```

- `rsort()`: Fa el mateix que l'anterior però en ordre invers. Exemples:

Exemple 8:

```
$lletres = array(“d”,”c”,”b”,”a”);
sort($lletres);
for ($i=0; $i<count($lletres); $i++) {
   echo $lletres[$i] . “ ”;
} // Dóna: d c b a

$nums = array(101,20,10,30);
sort($nums);
for ($i=0; $i<count($nums); $i++) {
   echo $lletres[$i] . " ";
} // Dóna: 101 30 20 10

$lletres = array("d","a",101,20);
sort($lletres);
for ($i=0; $i<count($lletres); $i++) {
   echo $lletres[$i] . " ";
} // Dóna: 101 20 d a
```

2.6.1 Arrays associatius

Aquest és el segon tipus d'arrays que hi ha a PHP. Aquests arrays es caracteritzen perquè els índexs no són numèrics sino de tipus cadena. Per tant, a cada element de l'array li assignem un valor (*kery*) per a accedir a ell.

Suposem que tenim una taula en que cada element emmagatzema el nombre de visites a la nostra web per cada dia de la setmana. Fent servir taules associatives podem fer el següent:

```
$visites[“dilluns”] = 346;
$visites[“dimarts”] = 340;
$visites[“dimecres”] = 246;
$visites[“dijous”] = 110;
$visites[“divendres”] = 198;
$visites[“dissabte”] = 412;
$visites[“diumenge”] = 590;
```

O també:

```
$visites = array(“dilluns” => 346, “dimarts” => 340, “dimecres” => 246, ...);
```

ATENCIÓ: Els arrays associatius no són interpretats dins de les dobles cometes, per tant per a poder-los imprimir a la pàgina web amb cadenes de text haurem de fer servir la concatenació. Vegem-ho a través d'un exemple:

```
$clients[“nom”] = “Joan”;
echo “El nom és $clients['nom']”; // Això genera un ERROR
echo “El nom és “ . $clients[“nom”]; // El nom és Joan
```

Els arrays associatius prendran tot el significat quan estudiem com obtenir dades a partir d'una base de dades.

2.6.2 Arrays multidimensionals

Els arrays multidimensionals no són res mes que arrays en els quals cada element és una altre taula. Al igual que els arrays normals hi han de tipus convencional o associatiu. Vegem-ho a través de dos exemples:

- Arrays multidimensionals convencionals.

```
$cal[] = array (1,"gener",31);
$cal[] = array (1,"febrer",28);
$cal[] = array (1,"març",31);
$cal[] = array (1,"abril",30);
echo $cal[2][1]; // març
echo $cal[0][2]; // 31
echo $cal[1][1]; // febrer
echo $cal[3][0]; // 1
```

- Arrays multidimensionals associatius:

```
$cal['mes1'] = array (1,"gener",31);
$cal['mes2'] = array (1,"febrer",28);
$cal['mes3'] = array (1,"març",31);
$cal['mes4'] = array (1,"abril",30);
echo $cal['mes3'][1]; // març
echo $cal['mes1'][2]; // 31
echo $cal['mes2'][1]; // febrer
echo $cal['mes4'][0]; // 1
```

2.7 Funcions d'usuari

A PHP per a definir funcions es fa servir la paraula reservada `function`. Les funcions poden estar situades a qualsevol part del codi PHP però, en general, es col·loquen al principi del *script*. En la invocació d'una funció és imprescindible l'ús de parèntesis, encara que la funció no rebi cap paràmetre. La sintaxi general d'una funció PHP és la següent:

```
function nom_funcio([argument1, argument2, ...])
{
   // Codi de la funció
   [return valor;]
}
```

L'objectiu de les funcions és agrupar fragments de codi que es repeteixen freqüentment en el desenvolupament d'un programa. Veiem alguns exemples de l'ús de funcions:

Exemple 1:

```
function mostrar_parametre($text)
{
   echo "El text enviat es: " . $text;
}

mostrar_parametre("Mostra això per pantalla");
```

Exemple 2:

```
function concatenar($text1, $text2)
{
   $concatenacio = $text1 . $text2;
   return $concatenacio;
}

echo "El resultat és: " . concatenar("Hola, ", "que tal?");
```

Exemple 3:

```
function area_triangle($base, $alcada)
{
   $area = ($base * $alcada)/2;
   return $concatenacio;
}

echo “L'area del triangle és “ . area_triangle(25,12);
```

Exemple 4:

```
function interes($valor, $percent)
{
   $resultat = ($valor * $percent) / 100;
   return $resultat;
}

echo “El 4.8% de 2864 és: “ . interes(2864,4.8);
```

2.8 Funcions de cadena

PHP té una gran facilitat per tractar les cadenes de text. Això és degut a que, una de les característiques a les que ha de donar resposta, és l'accés a base de dades i la manipulació d'aquestes dades. Per a tractar les cadenes PHP té una gran quantitat de funcions. Veurem ara les més importants:

- `array explode(string separador, string cadena [,int límit])`: Retorna una matriu de cadenes, cadascuna de les quals és una subcadena de `cadena` formada mitjançant la seva divisió en les fronteres marcades per la cadena `separador`. Si s'especifica el `límit` la matriu retornada tindrà, com a màxim, un nombre d'elements igual a `límit`. Exemple:

```
$data = “23/12/2008”;
$fragments = explode(“/”,$data);
echo $fragments[0]; // 23
echo $fragments[1]; // 12
echo $fragments[2]; // 2008
```

- `string implode(string separador, array peces)`: Retorna una cadena que conté una representació de tots els elements de la matriu `peces` en el mateix ordre, però amb el `separador` entre cadascun d'ells. Exemple:

```
$fragments[0] = 23;
$fragments[1] = 12;
$fragments[2] = 2008;
echo implode(“-”,$fragments); // 23-12-2008
```

- `string ltrim(string cadena)`: Retorna la cadena `cadena` però sense espais en blanc al principi de la cadena.
- `string rtrim(string cadena)`: Retorna la cadena `cadena` però sense espais en blanc al final de la cadena.
- `string trim(string cadena)`: Retorna la cadena `cadena` però sense espais en blanc ni al principi ni al final de la cadena. Exemples:

```
$salutacio = “   Hola   ”;
echo “¡” . ltrim($salutacio) . ”!”; // ¡Hola   !
echo “¡” . rtrim($salutacio) . ”!”; // ¡   Hola!
echo “¡” . trim($salutacio) . ”!”; // ¡Hola!
```

- `string ucfirst(string cadena)`: Retorna la cadena `cadena` però amb el primer caràcter en majúscules si és alfabètic.
- `string ucwords(string cadena)`: Retorna la cadena `cadena` però amb la primera lletra de cada paraula en majúscules, sempre que sigui alfabètic.
- `string strtolower(string cadena)`: Retorna la cadena `cadena` però amb totes les lletres en minúscules.
- `string strtoupper(string cadena)`: Retorna la cadena `cadena` però amb totes les lletres en majúscules. Exemples:

```
$salutacio = “hola, com et trobes?”;
echo ucfirst($salutacio); // Hola, com et trobes?
echo ucwords($salutacio); // Hola, Com Et Trobes?
$gransalut = strtoupper($salutacio);
echo $gransalut; // HOLA, COM ET TROBES?
echo strtolower($gransalut); // hola, com et trobes?
```

- `string substr(string cadena, int comencament [, int llarg])`: Retorna la porció de la cadena `cadena` especificada pels paràmetres `comencament` i `llarg`. Si no especifiquem `llarg` llavors retorna la cadena des de `comencament` fins al final. Numera la cadena començant per 0. Exemple:

```
$salutacio = “hola, com et trobes?”;
echo substr($salutacio, 6); // com et trobes?
echo substr($salutacio, 6, 5); // com e
echo substr($salutacio, 14, 3); // rob
```

- `int strcmp(string cad1, string cad2)`: Retorna <0 si `cad1` és més petit que `cad2`. Retorna >0 si `cad1` és mes gran que `cad2` i 0 si són iguals. Aquesta comparació és sensible a majúscules i minúscules.
- `int strcasecmp(string cad1, string cad2)`: Retorna <0 si `cad1` és més petit que `cad2`. Retorna >0 si `cad1` és mes gran que `cad2` i 0 si són iguals. Aquesta comparació no és sensible a majúscules i minúscules. Exemples:

```
$sa1 = “hola”;
$sa2 = “adeu”;
$sa3 = “Hola”;
echo strcmp($sa1, $sa2); // Retorna: >0
echo strcmp($sa2, $sa1); // Retorna: <0
echo strcmp($sa1, $sa3); // Retorna: >0
echo strcasecmp($sa1, $sa3); // Retorna: 0
```

- `string nl2br(string cadena)`: Retorna la cadena `cadena` amb `<BR>`'s en lloc del caràcter de nova línia. Exemples:

```
$salut = “hola!!\nCom anem?”;
echo nl2br($salut);
// Retorna
// hola!!
// Com anem?
```

3. Tractament de formularis HTML amb PHP

3.1 Introducció

Els formularis no formen part de PHP sino el llenguatge estàndard d'internet HTML. PHP ofereix una gran facilitat en el tractament de formularis i aquests són la base de la inserció de dades o les cerques a les bases de dades.

Veiem en primer lloc com podem, fent servir PHP, tractar un formulari. El primer que hem de fer és indicar al formulari quin document PHP l'ha de tractar, per a fer-ho hem de posar el nom del document PHP al `ACTION` del formulari de la següent forma: `<FORM NAME="fromphp" ACTION="php_que_el_tracta.php">`.

Quan vam estudiar els formularis vam veure també que hi ha havia dues formes d'enviar-los a través de la web. Aquestes dues formes eren:

- GET: Amb aquesta forma l'adreça web es completava amb la informació del formulari.
- POST: D'aquesta forma les dades del formulari s'empaquetaven i s'enviaven sense que quedés reflectit a l'adreça web.

En qualsevol cas, el més recomanable és enviar el formulari via POST per a evitar que a l'adreça web es visualitzi la informació introduïda. Així doncs el formulari ha de contenir els següents paràmetres:

```
<FORM NAME="fromphp" ACTION="php_que_el_tracta.php" METHOD="post">
```

Enviar un formulari vol dir fer un `submit` d'aquest. Per això és important que a tots els formularis hi ha hagi un botó com el següent `<INPUT TYPE="submit" VALUE="Enviar">`.

Així dons tenim la següent estructura:

```
<FORM NAME="fromphp" ACTION="php_que_el_tracta.php" METHOD="post">
   <!-- ALTRES ELEMENTS DEL FORMULARI -->
   <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

El que es produirà en el moment en que nosaltres clickem sobre el botó *Enviar* és que totes les dades del formulari s'enviaran, fent servir el mètode POST, a l'arxiu PHP que tractarà el formulari, que en el nostre cas és `php_que_el_tracta.php`. Veiem, doncs, ara com, dins de l'arxiu PHP, podem tractar els diferents elements d'un formulari.

3.2 Comportament de PHP en rebre un formulari

Per a saber tractar les dades enviades per un formulari amb PHP, primer hem d'entendre què és el que fa PHP quan rep un formulari. En rebre un formulari PHP fa el següent:

- Si el formulari ha estat enviat mitjançant el mètode POST, PHP crea un array associatiu anomenat `$_POST` el elements del qual són els valors dels diferents objectes del formulari enviat. Per accedir al valor d'un objecte en concret haurem de fer `$_POST['nom_del_name_del_objecte']`. ATENCIÓ: Accedim al *value* de l'objecte, per tant és imprescindible que tots els objectes del formulari tinguin el seu *value* definit.
- Si el formulari ha estat enviat mitjançant el mètode GET, PHP crea un array associatiu anomenat `$_GET` el elements del qual són els valors dels diferents objectes del formulari enviat. Per accedir al valor d'un objecte en concret haurem de fer `$_GET['nom_del_name_del_objecte']`.

Veiem un exemple:

```
<FORM NAME="fromphp" ACTION="php_que_el_tracta.php" METHOD="post">
   <INPUT TYPE="text" NAME="dada">
   <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

```
php_que_el_tracta.php
<?php
echo $_POST['dada'];
?>
```

En aquest exemple el formulari envia les dades que conté (de fet només una caixa de text) a l'arxiu `php_que_el_tracta.php`. Aquest arxiu obté el valor introduït a la caixa de text mitjançant l'array associatiu `$_POST['dada']`.

- Un formulari només pot enviar les dades fent servir un mètode. Però si completem nosaltres l'adreça web de destí podem fer servir els dos mètodes. Aquesta tècnica és molt útil a vegades quan volem enviar un formulari indicant alguna opció adicional. Vegem-ho a través d'un exemple:

```
<FORM NAME="fromphp" ACTION="tractador.php?nom=pep" METHOD="post">
   <INPUT TYPE="text" NAME="dada">
   <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

```
tractador.php
<?php
echo "Dada enviada via POST: " . $_POST['dada'] . "<br>";
echo "Dada enviada via GET: " . $_GET['nom'] . "<br>";
?>
```

Com es pot observar, per a completar una adreça web hem de fer servir el símbol ? i després especifiquem el nom de la variable i el seu valor. Si volem incloure més variables i valors hem de fer servir el símbol &. L'esquema general per a completar adreces web amb variables i valors és el següent:

```
tractador.php?var1=valor1&var2=valor2&var3=valor3& ... &varN=valorN
```

Veiem ara la forma de tractar els diferents elements d'un formulari fent servir les tècniques que hem vist.

3.2.1 Caixes de text

La millor forma de veure com accedir a una caixa de text d'un formulari dins de PHP és a través d'un exemple:

```
<FORM NAME="fromphp" ACTION="tractador.php" METHOD="post">
   <INPUT TYPE="text" NAME="dada">
   <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

```
tractador.php
<?php
echo "Dada enviada via POST: " . $_POST['dada'] . "<br>";
?>
```

Com es pot veure per accedir al valor de la caixa de text hem de fer servir l'array associatiu `$_POST` que com a paràmetre ha de rebre el nom de la caixa.

3.2.2 Àrees de text

Vegem-ho a través d'un exemple:

```
<FORM NAME=”fromphp” ACTION=”tractador.php” METHOD=”post”>
   <TEXTAREA NAME=”descripcio” ROWS=”5” COLS=”20”>Això és una prova</TEXTAREA>
   <INPUT TYPE=”submit” VALUE=”Enviar”>
</FORM>
```

```
tractador.php
<?php
echo “Dada enviada via POST: ” . nl2br($_POST['descripcio']) . “<br>”;
?>
```

Com es pot veure es fa de la mateixa manera que les caixes de text, la única diferència resideix en l'ús de la funció `nl2br()`. L'ús d'aquesta funció és imprescindible perquè les àrees de text emmagatzemen els salts de línia en format `\n` i no `<BR>`, per tant si no incloem aquest funció l'àrea de text es mostra sense salts de línies.

3.2.3 Caselles de verificació o checkboxes

Per a controlar aquest element d'un formulari haurem d'explicar una nova funció anomenada `isset()`. La sintaxi d'aquesta funció és `boolean isset(variable)` i rep com a paràmetre una variable i ens diu si aquesta variable està definida (retorna `true`) o no (retorna `false`). Encara que no ho sembli aquesta funció és molt útil a PHP a on no s'han de declarar les variables per a fer-les servir.

Vegem ara com aplicar-ho als *checkboxes* a través d'un exemple:

```
<FORM NAME=”fromphp” ACTION=”tractador.php” METHOD=”post”>
   Qu&egrave; fas en aixecar-te?<BR>
   <INPUT TYPE=”checkbox” NAME=”sel1” VALUE=”Rentar-me la cara”>
      Rentar-me la cara<BR>
   <INPUT TYPE=”checkbox” NAME=”sel2” VALUE=”Rentar-me les dents”>
      Rentar-me les dents<BR>
   <INPUT TYPE=”checkbox” NAME=”sel3” VALUE=”Pentinar-me”>
      Pentinar-me<BR>
   <INPUT TYPE=”submit” VALUE=”Enviar”>
</FORM>
```

```
tractador.php
<?php
if (isset($_POST['sel1'])) {
   echo $_POST['sel1'] . “<br>”;
}
if (isset($_POST['sel2'])) {
   echo $_POST['sel2'] . “<br>”;
}
if (isset($_POST['sel3'])) {
   echo $_POST['sel3'] . “<br>”;
}
?>
```

Com es pot veure per a cada *checkbox* enviat és imprescindible comprovar si l'hem rebut o no perquè si el *checkbox* no està marcat el formulari HTML no l'envia al PHP.

3.2.4 Botons de ràdio o radiobuttons

Vegem com accedir als *radiobuttons* a través d'un exemple:

```
<FORM NAME=”fromphp” ACTION=”tractador.php” METHOD=”post”>
   Quant &eacute;s 2+2?<BR>
   <INPUT TYPE=”radio” NAME=”rad” VALUE=”44”>44<BR>
   <INPUT TYPE=”radio” NAME=”rad” VALUE=”22” CHECKED>22<BR>
   <INPUT TYPE=”radio” NAME=”rad” VALUE=”2”>2<BR>
   <INPUT TYPE=”submit” VALUE=”Enviar”>
</FORM>
```

```
tractador.php
<?php
if (isset($_POST['rad')) echo “Has triat la opcio “ . $_POST['rad'];
?>
```

Aquí hem inclòs també la funció `isset()`, però si obliguem a que un botó de ràdio estigui obligatòriament marcat amb el `CHECKED` llavors no caldria perquè s'enviaria amb tota seguretat al script PHP.

3.2.5 Menús desplegables o comboboxes

En el cas els menús desplegables diferenciarem entre dos possibilitats: menús desplegables de selecció simple i menús desplegables de selecció múltiple.

1. Menús desplegables de selecció simple. Vegem el següent exemple per a veure com tractar aquests controls:

```
<FORM NAME=”fromphp” ACTION=”tractador.php” METHOD=”post”>
   Quin ordinador?<BR>
   <SELECT NAME=”ord”>
      <OPTION VALUE=”pen”>Pentium</OPTION>
      <OPTION VALUE=”cel”>Celeron</OPTION>
      <OPTION VALUE=”amd”>AMD</OPTION>
      <OPTION VALUE=”mac”>Mac</OPTION>
   </SELECT>
   <INPUT TYPE=”submit” VALUE=”Enviar”>
</FORM>
```

```
tractador.php
<?php
echo “Has triat l'ordinador ” . $_POST['ord'];
?>
```

Com es pot veure, pels menús desplegables de selecció simple, la situació és igual de simple que per a les

caixes de text. Hem de tenir en compte que la informació que envia a PHP el formulari és la continguda al VALUE del control.

2. Menús desplegables de selecció múltiple. Aquests menús són una mica més complexos. Com que es pot triar més d'una opció i el SELECT només té un nom, els valors seleccionats s'enviaran a PHP en format d'array. Ho entendrem millor a través d'un exemple:

```
<FORM NAME="fromphp" ACTION="tractador.php" METHOD="post">
   Dispositius?<BR>
   <SELECT NAME="disp[]" MULTIPLE>
      <OPTION VALUE="kit">Kit multim&egrave;dia</OPTION>
      <OPTION VALUE="gra">Grabadora de DVD's</OPTION>
      <OPTION VALUE="cam">Web Cam</OPTION>
      <OPTION VALUE="mic">Micr&ograve;fon</OPTION>
   </SELECT>
   <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

```
tractador.php
<?php
$num_elements_triats = count($_POST['disp']);
echo "Has triat $num_elements_triats elements<BR>";
for ($i=0; $i<$num_elements_triats; $i++) {
   echo $_POST['disp'][$i] . "<BR>";
}
?>
```

Com es pot veure, el nom que haurem de posar al SELECT haurà de contenir [] per a indicar que, potser, s'enviaran més d'una dada amb aquest nom i s'haurà de fer en format d'array. Després, a PHP, tractarem $_POST['nom_select'] com si fos un array.

3.2.6 Camps ocults

Moltes vegades pot ser útil que el formulari contingui més informació de la que l'usuari pugui veure i que s'enviï juntament amb ell. Aquesta informació es pot incloure en el VALUE d'un camp ocult com es pot veure en el següent exemple:

```
<FORM NAME="fromphp" ACTION="tractador.php" METHOD="post">
   <INPUT TYPE="hidden" NAME="dada" VALUE="Dada oculta">
   <INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

```
tractador.php
<?php
echo "Dada OCULTA enviada via POST: " . $_POST['dada'] . "<br>";
?>
```

3.2.7 Exemple d'accés als elements d'un formulari amb PHP

Veurem un exemple en que es demanarà a l'usuari:

- Nom – Caixa de text.
- Cognoms – Caixa de text.
- Edat – Caixa de text.
- Domicili – Àrea de text.
- Rang de sou que es vol cobrar – Menú desplegable.
- Consideració de la seva feina – Botons de ràdio.

I després es mostraran totes aquestes dades formatades.

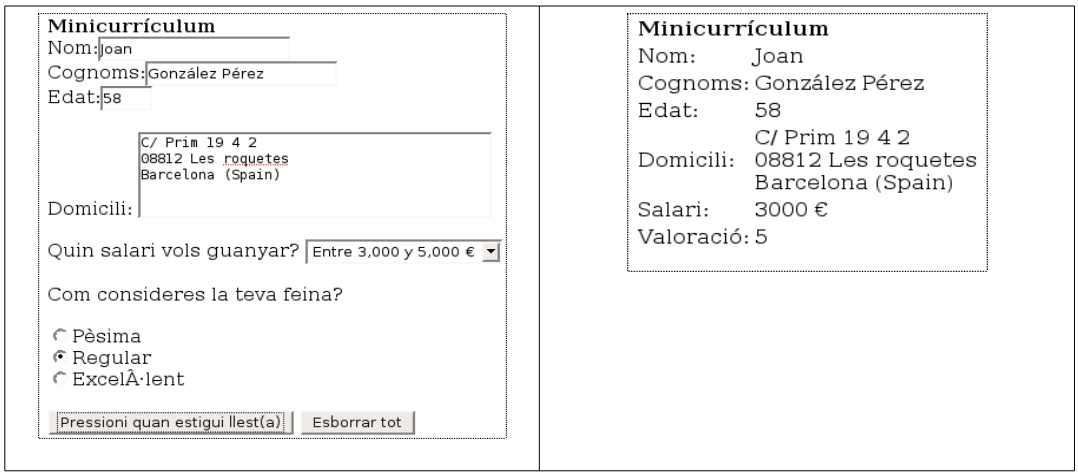
El codi HTML de la pàgina del formulari serà el següent (`index.html`):

```
<HTML>
<HEAD><TITLE>Curr&iacute;culum</TITLE></HEAD>
<BODY>
<B>Minicurr&iacute;culum</B>
<FORM NAME=”formdades” METHOD="post" ACTION="minicu.php">
   Nom:<INPUT NAME="nom" TYPE="text"><BR>
   Cognoms:<INPUT NAME="cognoms" TYPE="text"><BR>
   Edat:<INPUT NAME="edat" TYPE="Text" SIZE="3">
  <BR><BR>
   Domicili:
   <TEXTAREA NAME="domicili" ROWS="4" COLS="40"></TEXTAREA>
  <BR><BR>
  Quin salari vols guanyar?
   <SELECT NAME="salari">
      <OPTION VALUE=”0”> Menos de 1000 &euro;</OPTION>
      <OPTION VALUE=”1000”>Entre 1,000 y 3,000 &euro;</OPTION>
      <OPTION VALUE=”3000”>Entre 3,000 y 5,000 &euro;</OPTION>
      <OPTION VALUE=”5000”>mas de 5,000 &euro;</OPTION>
   </SELECT>
  <BR><BR>
   Com consideres la teva feina?
  <BR>
   <BR><INPUT NAME="feina" TYPE="Radio" VALUE="0">P&egrave;sima
   <BR><INPUT NAME="feina" TYPE="Radio" VALUE="5">Regular
   <BR><INPUT NAME="feina" TYPE="Radio" VALUE="10">Excel·lent
  <BR><BR>
   <INPUT TYPE="SUBMIT" VALUE="Pressioni quan estigui llest(a)">
   <INPUT TYPE="RESET" VALUE="Esborrar tot">
</FORM>
</BODY>
</HTML>
```

I la pàgina `minicu.php` serà:

```
<HTML>
<HEAD><TITLE>Curr&iacute;culum</TITLE></HEAD>
<BODY>
<TABLE>
   <TR><TD COLSPAN=”2”><B>Minicurr&iacute;culum</B></TD></TR>
   <TR><TD>Nom:</TD><TD><?php echo $_POST['nom']; ?></TD></TR>
   <TR><TD>Cognoms:</TD><TD><?php echo $_POST['cognoms']; ?></TD></TR>
   <TR><TD>Edat:</TD><TD><?php echo $_POST['edat']; ?></TD></TR>
   <TR><TD>Domicili:</TD><TD><?php echo nl2br($_POST['domicili']); ?></TD></TR>
   <TR><TD>Salari:</TD><TD><?php echo $_POST['salari']; ?> &euro;</TD></TR>
   <TR><TD>Valoraci&oacute;:</TD><TD><?php echo $_POST['feina']; ?></TD></TR>
</TABLE>
</BODY>
</HTML>
```

I gràficament:



Una altre forma de fer l'exercici anterior consistiria en fer que només hi hagués una pàgina en lloc de dues. Per a poder gestionar això hauríem de fer servir la funció `isset()`. Veiem que codi que ens permet fer l'exemple anterior però només en una pàgina.

La pàgina `minicu.php` serà:

```
<HTML>
<HEAD><TITLE>Curr&iacute;culum</TITLE></HEAD>
<BODY>
<?php
if (!isset($_POST['nom'])) {
?>
<B>Minicurr&iacute;culum</B>
<FORM NAME=”formdades” METHOD="post" ACTION="minicu.php">
   Nom:<INPUT NAME="nom" TYPE="text"><BR>
   Cognoms:<INPUT NAME="cognoms" TYPE="text"><BR>
   Edat:<INPUT NAME="edat" TYPE="Text" SIZE="3">
  <BR><BR>
   Domicili:
   <TEXTAREA NAME="domicili" ROWS="4" COLS="40"></TEXTAREA>
  <BR><BR>
  Quin salari vols guanyar?
   <SELECT NAME="salari">
      <OPTION VALUE=”0”> Menos de 1000 &euro;</OPTION>
      <OPTION VALUE=”1000”>Entre 1,000 y 3,000 &euro;</OPTION>
      <OPTION VALUE=”3000”>Entre 3,000 y 5,000 &euro;</OPTION>
      <OPTION VALUE=”5000”>mas de 5,000 &euro;</OPTION>
   </SELECT>
  <BR><BR>
   Com consideres la teva feina?
  <BR>
   <BR><INPUT NAME="feina" TYPE="Radio" VALUE="0">P&egrave;sima
   <BR><INPUT NAME="feina" TYPE="Radio" VALUE="5">Regular
   <BR><INPUT NAME="feina" TYPE="Radio" VALUE="10">Excel·lent
  <BR><BR>
   <INPUT TYPE="SUBMIT" VALUE="Pressioni quan estigui llest(a)">
   <INPUT TYPE="RESET" VALUE="Esborrar tot">
</FORM>

<?php
} else {
?>

<TABLE>
   <TR><TD COLSPAN=”2”><B>Minicurr&iacute;culum</B></TD></TR>
   <TR><TD>Nom:</TD><TD><?php echo $_POST['nom']; ?></TD></TR>
   <TR><TD>Cognoms:</TD><TD><?php echo $_POST['cognoms']; ?></TD></TR>
   <TR><TD>Edat:</TD><TD><?php echo $_POST['edat']; ?></TD></TR>
   <TR><TD>Domicili:</TD><TD><?php echo nl2br($_POST['domicili']); ?></TD></TR>
   <TR><TD>Salari:</TD><TD><?php echo $_POST['salari']; ?> &euro;</TD></TR>
   <TR><TD>Valoraci&oacute;:</TD><TD><?php echo $_POST['feina']; ?></TD></TR>
</TABLE>

<?php
```

```
}
?>

</BODY>
</HTML>
```

Com es pot comprovar el que fem és:

- Creem l'estructura HTML normal, però mostrarem un contingut o un altre depenent d'una condició.
- Si és la primera vegada que entrem a la pàgina (és a dir la variable `$_POST['nom']` no està definida) mostrarem el formulari d'enviament de dades.
- Si la variable `$_POST['nom']` sí que està definida és perquè s'ha enviat el formulari i per tant mostrarem el resultat formatat en una taula.

4. Sessions web

4.1 Introducció

A vegades convé emmagatzemar informació sobre un determinat usuari, com poden ser:

- Preferències en l'ús de la pàgina.
- Informació que l'usuari ha triat (carro de la compra).
- Identificació de l'usuari (login i contrasenya).

Aquesta informació és exclusiva de cada sessió que l'usuari inicialitza amb la nostra plana web (cada vegada s'ha d'identificar o ha de triar els productes que vol comprar), i per tant aquesta informació s'ha d'emmagatzemar en algun lloc durant la vida de la sessió de l'usuari. La forma d'emmagatzemar aquesta informació dóna lloc a dues formes de controlar les sessions:

- Cookies: La informació s'emmagatzema a l'ordinador de l'usuari.
- Sessions web: La informació s'emmagatzema al servidor.

Com que les cookies poden donar problemes (l'usuari les pot tenir desabilitades) i no poden emmagatzemar gran quantitat de dades nosaltres ens centrarem en les sessions web.

Bàsicament una sessió no és res més que la seqüència de pàgines que un usuari visita en un lloc web; des de que entra al nostre *site* fins que surten. PHP el que fa es crear un **identificador únic de sessió (SID)** al servidor per a cada usuari i s'assigna a cada una de les pàgines que sol·licita un mateix usuari.

Per defecte, l'únic que emmagatzema la sessió al navegador és una cookie que conté l'id de sessió de l'usuari. Una de les avantatges de les sessions és que continuen funcionant encara que les cookies estiguin desabilitades. Quan PHP detecta que les cookies no estan actives afegeix, automàticament, l'id de la sessió, com una cadena de consulta, a tots els enllaços d'una pàgina. Perquè això funcioni s'han de complir dos requisits:

1. Les pàgines han de tenir extensió *.php*.
2. És necessari que *session.use_trans_sid* estigui activat a l'arxiu php.ini.

4.1 Configuració de php.ini *per a que funcionin les sessions*

L'arxiu de configuració del comportament de PHP és el *php.ini* que a Linux es troba a la carpeta `/etc/php5/cli` i a Windows a la carpeta `c:\directori_instal·lacio\php\`. Obrim el *php.ini* i anem a la secció de les sessions. Allà trobarem, entre d'altres, tres valors que hem de verificar per tal que les sessions funcionin correctament. Aquests tres valors són:

- `session.save_path`: Lloc a on s'emmagatzemaran les sessions.
- `session.use_cookies`: Si està a 1 indica a PHP que estableixi el SID mitjançant cookies.
- `session.use_trans_sid`: Si està a 1 indica a PHP que estableixi el SID com una cadena de consulta a

la URL. Per defecte està a 0, per tant si el client no té activades les cookies, no podrem fer servir les sessions.

Ha de quedar clar que si `session.use_cookies=0` i `session.use_trans_sid=0` vol dir que les sessions estan desabilitades.

4.3 Inicialització d'una sessió

Per a inicialitzar una sessió farem servir la funció `session_start()`. Aquesta funció s'encarrega de:

- Crear una nova sessió si és la primera vegada que l'executem.
- Reestablir la sessió existent si ja la hem inicialitzat.

Aquesta funció té una particularitat que fa que el seu ús estigui molt localitzat. El que fa la funció `session_start()` és completar el *header* de la pàgina HTML. Això vol dir que si executem aquesta funció després d'haver enviat algun fragment HTML al client la funció no podrà realitzar la seva tasca. En altres paraules, ÉS ABSOLUTAMENT NECESSARI QUE LA FUNCIÓ `session_start()` APAREGUI ABANS D'ENVIAR QUALSEVOL COSA AL CLIENT. Per evitar qualsevol error en aquest sentit el que es fa és posar la funció al principi de la pàgina que conté el codi PHP.

4.4 Variables de sessió

En inicialitzar una sessió es crea un array associatiu anomena `$_SESSION[]`. Aquest array es crea buit i som nosaltres els que l'anirem omplint amb la informació que creiem convenient. Com que és un array associatiu l'omplirem tal com es veu en els següents exemples:

```
$_SESSION['nom'] = “Joan”;

$_SESSION['anys'] = 18;

$_SESSION['num_articles'] = 25;

$_SESSION['preu'] = 123.25;

$_SESSION['mes'][0] = “Gener”;
$_SESSION['mes'][1] = “Febrer”;
...
$_SESSION['mes'][11] = “Desembre”;

$_SESSION['mes'][] = “Gener”;
$_SESSION['mes'][] = “Febrer”;
...
$_SESSION['mes'][] = “Desembre”;
```

Com es pot veure l'array associatiu `$_SESSION[]` ens permet crear variables associades només a un usuari i un navegador que són emmagatzemades al servidor, no a l'ordinador de l'usuari.

A vegades ens interessarà esborrar una variable de sessió, per això farem servir la funció unset(). Aquesta funció ens permet eliminar una variable totalment. Veiem un exemple:

```
// Volem esborrar la variable $_SESSION['num_articles'] perquè ja no
// la necessitem a la nostra sessió

unset($_SESSION['num_articles']);

// Amb això la nostra variable de sessió desapareix completament.
```

D'altres vegades ens convindria esborrar totes les variables de sessió de cop. Això es pot fer redefininit l'array de `$_SESSION[]`. Vegem-ho a través d'un exemple.

```
// Volem esborrar totes les variables que conté l'array $_SESSION[]

$_SESSION = array();

// Amb això totes les variables de sessió desapareixen completament.
```

4.5 Finalitzar una sessió

El primer que crida l'atenció del programador que comença a PHP és la necessitat d'iniciar la sessió per a poder finalitzar-la. Per a finalitzar una sessió farem servir la funció `session_destroy()`, que destrueix el lligam entre el client i el servidor. La manera correcta de fer-ho és la següent:

```
// Iniciem la sessió
session_start();

// Esborrem totes les variables de sessió
$_SESSION = array();

// Destruïm la sessió
session_destroy();

// Amb això hem eliminat per complet la sessió que tenim amb el servidor
// Ara podríem iniciar una de nova
```

4.6 Error comú amb sessions

Com hem dit abans la sessió s'ha d'inicialitzar abans d'enviar qualsevol dada al client. Un dels errors més comuns quan es fan servir sessions es deixar línies en blanc abans de la inicialització de PHP o enviar alguna sortida al client. Si cometem algun dels errors anteriors obtindrem un missatge d'error similar al següent

Warning: session_start() [function.session-start]: Cannot send session cache limiter - headers already sent (output started at /var/www/C20072008/hola.php:2) in **/var/www/C20072008/hola.php** on line 3

4.7 Exemple de sessions

Veiem un exemple a on hi ha 3 pàgines:

- `index.php`: Aquesta pàgina ens permetrà fer un *login* per a identificar-nos. Quan ho fem ens mostrarà informació personal i ens permetrà accedir a la pàgina `mesdades.php`.
- `mesdades.php`: Ens mostrarà més informació personal si ens hem identificat i la nostra sessió té les variables de sessió adequades.
- `desconnectar.php`: Aquesta pàgina finalitzarà la sessió.

`index.php`:

```
<?php
// Iniciem la sessió
session_start();
// Comprovem si l'usuari ens ha enviat les dades o no
if (isset($_POST['login'])) {
   if (($_POST['login'] == “admin”) && ($_POST['pass'] == “passadmin”)) {
      // L'usuari s'ha identificat correctament
      $_SESSION['identificat'] == true;
      $_SESSION['login'] == $_POST['login'];
   }
}
// Ara mostrarem el formulari HTML si l'usuari no s'ha identificat
if (!isset($_SESSION['identificat'])) {
   ?>
   <form name=”form1” action=”index.php” method=”post”>
   Login: <input type=”text” name=”login”><br>
   Password: <input type=”password” name=”pass”><br>
   <input type=”submit” value=”Enviar”>
   </form>
   <?php
} else { // Aquí mostrarem la informació personal perquè l'usuari s'ha
identificat
   ?>
```

```
      Benvingut <?php echo $_SESSION['login'] ?>.<br>
      Si vols sortir clicka <a
href=”desconnectar.php”>aqu&iacute;</a>.<br>
      El teu tel&egrave;fon &eacute;s: 977845215.
      <a href=”mesdades.php”>Aqu&iacute;</a> tens m&eacute;s
informaci&oacute; sobre tu.
   <?php
}
?>
```

mesdades.php:

```
<?php
// Iniciem la sessió
session_start();
// No mostrarem més informació si l'usuari no s'ha identificat abans
if (!isset($_SESSION['identificat'])) {
   ?>
   No t'has identificat.  Fes-ho <a href=”index.php”>aqu&iacute;</a>
   <?php
} else { // Aquí mostrarem la informació personal perquè l'usuari s'ha identificat
   ?>
      Benvingut <?php echo $_SESSION['login'] ?>.<br>
      Si vols sortir clicka <a href=”desconnectar.php”>aqu&iacute;</a>.<br>
      El teu email &eacute;s: am@loquesea.com
   <?php
}
?>
```

desconnectar.php:

```
<?php
// Iniciem la sessió
session_start();
// Esborrem totes les variables de sessió
$_SESSION = array();
// Destruïm la sessió
session_destroy();
?>
Has finalitzat amb la sessió.  <a href=”index.php”>Torna</a> a la p&agrave;gina principal.
```

Comproveu que no podreu veure la informació a `mesdades.php` si abans no us heu identificat a la pàgina `index.php` i que us haureu de tornar a identificar després de visitar la pàgina `desconnectar.php`.

4.8 Carro de la compra

Una de les aplicacions que més es fan servir amb les sessions són aquelles que permeten a l'usuari tractar la web com un carro de la compra a on se li ofereixen diferents productes que pot anar triant. Per a poder fer un carro de la compra serà necessari explicar algunes funcions i estructures que ens seran molt útils:

array_search:

Aquesta funció té la següent sintaxi: `mix array_search( “el_que_vull cercar”,array);` El que fa aquesta funció és cercar `“el_que_vull cercar”` entre els valors de l'array `array`. Ens retornarà `false` si no trova cap coincidència o l'índex de l'array a on troba la coincidència. Veiem un exemple:

```
<?php
$fruites = array(0=>”poma”, “taronja”, ”maduixa”, “plàtan”);
$resultat = array_search(“maduixa”,$fruites);
echo $resultat // El resultat serà 2
$resultat = array_search(“pera”,$fruites);
echo $resultat // El resultat serà FALSE
?>
```

array_splice:

Aquesta funció ens permet suprimir una part d'una array. La sintaxi d'aquesta funció és la següent:

`array_splice(array, lloc, desplaçament);` El que fa aquesta funció és eliminar el fragment d'array des del `lloc` el nombre determinat pel paràmetre `desplaçament`. Veiem un exemple per a entendre el seu funcionament:

```
<?php
$fruites = array(0=>”poma”, “taronja”, ”maduixa”, “plàtan”, ”pera”, “castanya”);
array_splice($fruites, 2, 2);
// Si fem ...
for ($i=0; i<count($fruites); $i++) {
   echo $fruites[$i] . “ ”;
}
// El resultat és poma taronja pera castanya
// Com es pot veure hem eliminat des de la posició d'índex 2 dos ítems de l'array.
?>
```

exit:

Ens permet sortir finalitzar l'execució dels *scripts* d'uns pàgina PHP. La seva sintaxi és `exit();`

header:

El header ens permetrà realitzar redireccions a d'altres pàgines. La sintaxi d'aquesta funció és: `header(“Location: www.lloc.com”);` S'ha de fer servir combinada amb la funció `exit()` per a evitar possibles errors. Veiem un exemple:

```
<?php
session_start();
if (!isset($_SESSION['identificat'])) {
   header(“Location: login.php”);
   exit();
}
// Aquest funció ens envia a la pàgina login.php
// si no estem identificats.
?>
```

foreach:

Aquesta estructura ens permet recorrer arrays convencionals i associatius. La seva sintaxi és la següent: `foreach ($array as $clau => $valor)`. Això ens permet recorrer l'array `$array` d'un en un assignant a la variable `$clau` l'índex i a la variable `$valor` el valor contingut. Veiem un exemple:

```
<?php
$mesos['gener'] = 31;
$mesos['febrer'] = 28;
$mesos['marc'] = 31;
foreach ($mesos as $clau => $valor) {
   echo “El mes $clau t&acute; $valor dies”;
}
// Resultat
// gener té 31 dies
// febrer té 28 dies
// marc té 31 dies
?>
```

Per a crear un carro de la compra haurem de fer servir les funcions anterior, sobretot les dues primeres. Veiem com podem fer un carro de la compra. Crearem tres planes web:

1. `carro.php`: Aquesta plana ens permetrà afegir o treure productes del nostre carro de la compra i sortir de la sessió (*logout*). Tindrem una llista de diferents productes que estaran triats o no i que podrem afegir o treure del carro.
2. `desconnectar.php`: Aquesta plana ens eimina la sessió i ens redirecciona a la pàgina anterior.

desconnectar.php

```php
<?php
session_start();
$_SESSION = ARRAY();
session_destroy();
header("Location: carro.php");  // Això ens redirecciona a carro.php
exit();
?>
```

carro.php

```php
<?php
session_start();
if (isset($_GET['c'])) {
   if (!isset($_SESSION['cistella'])) {
      // Encara no hem triat cap producte aquest és el primer
      $_SESSION['cistella'][$_GET['c']] = 1;
   } else {
      // Ja hem triat algun producte.
      // Hem de veure si està o no a la cistella
      // per a afegir-lo o treure'l
      if (!isset($_SESSION['cistella'][$_GET['c']])) {
         // No hi és a la cistella
        $_SESSION['cistella'][$_GET['c']] = 1;
      } else {
         // És a la cistella  L'he d'esborrar
         $i=0;
         foreach ($_SESSION['cistella'] as $clau => $valor) {
             if ($clau == $_GET['c']) {
                    array_splice($_SESSION['cistella'],$i,1);
             }
             $i++;
         }
      }
   }
}
?>
<html>
<head><title>Carro</title></head>
<body>
<table>
<?php
// Array que conté els productes que podem comprar
$productes = array(0=>"Memòria","Disc dur","Processador",
                   "Placa base","DVD","Caixa");
$productesRef = array(0=>"mem","hd","cpu","pb","dvd","cx");
for ($i=0;$i<count($productes); $i++) {
?>
<tr><td>Producte: <?php echo $productes[$i]; ?></td>
<?php
if (!isset($_SESSION['cistella'])) {
   $text = "Afegir al cistell";
} else {
   if (!isset($_SESSION['cistella'][$productesRef[$i]])) {
      // No hi és a la cistella
      $text = "Afegir al cistell";
   } else {
      // És a la cistella
      $text = "Treure del cistell";
   }
}
?>
<td>
<A href="carro.php?c=<?php echo $productesRef[$i]; ?>"><?php echo $text; ?></A>
</td>
</tr>
```

```
<?php
}
?>
</table>
<A href="desconnectar.php">Abandona la sessi&oacute;</A>
<br>
</body>
</html>
```

Com es pot veure amb això podem controlar un carro de la compra amb un nombre determinat de productes. El que fa és el següent:

- Si enviem un producte (variable `$_GET['c']`) el que farà serà incorporar-lo a l'array de sessió `$_SESSION['cistella']` si no estava triat o eliminar-lo de l'array si ja ho estava.
- Per a esborrar-lo farem servir l'estructura `foreach` i `array_splice`.

  També es pot servir, amb modificacions, per a controlar un conjunt de productes i les unitats de cadascun.

5. PHP amb bases de dades MySQL

PHP pot accedir a, pràcticament, qualsevol base de dades: Oracle, Microsoft Access, MySQL, ... Però per al motor MySQL té suport natiu, fet que indica que el seu rendiment serà superior amb aquesta base de dades. Això, combinat amb el fet que MySQL sigui un servidor de base de dades de lliure distribució, fa que la combinació PHP-MySQL sigui molt estesa. De fet, una gran solució per al desenvolupament d'aplicacions sense cost de llicència és la combinació que s'ha anomenat LAMP (Linux – Apache – MySQL – PHP).

Ens centrarem ara en l'ús de PHP amb motors de bases de dades MySQL.

5.1 Creació de bases de dades y taules

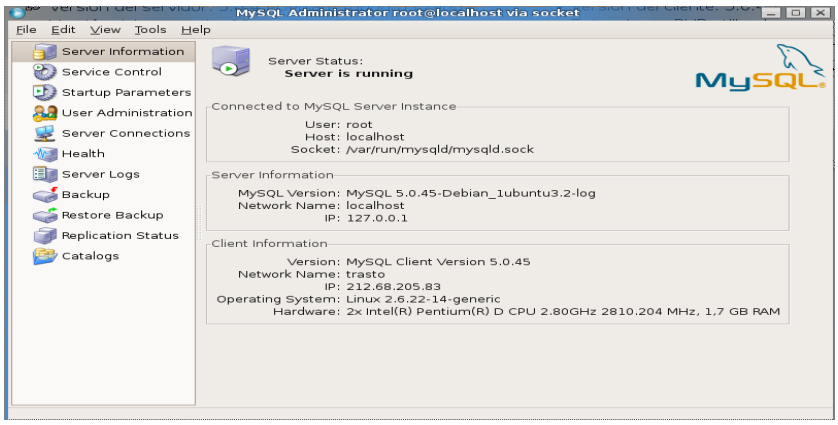
Hi ha diferents eines i formes de crear, definir i modificar les nostres bases de dades. Les dues eines que més es fan servir són:

1. MySQL Administrator. Eina gràfica que ens permet connectar-nos al motor MySQL i crear bases de dades, taules, afegir registres, ...
2. PHPMyAdmin. Eina web que ens permet gestionar les bases de dades del nostre servidor MySQL. A diferència de l'anterior no ens permet connectar-nos a qualsevol servidor MySQL. Normalment s'accedeix mitjançant l'adreça web: http://servidorMySQL/phpmyadmin.

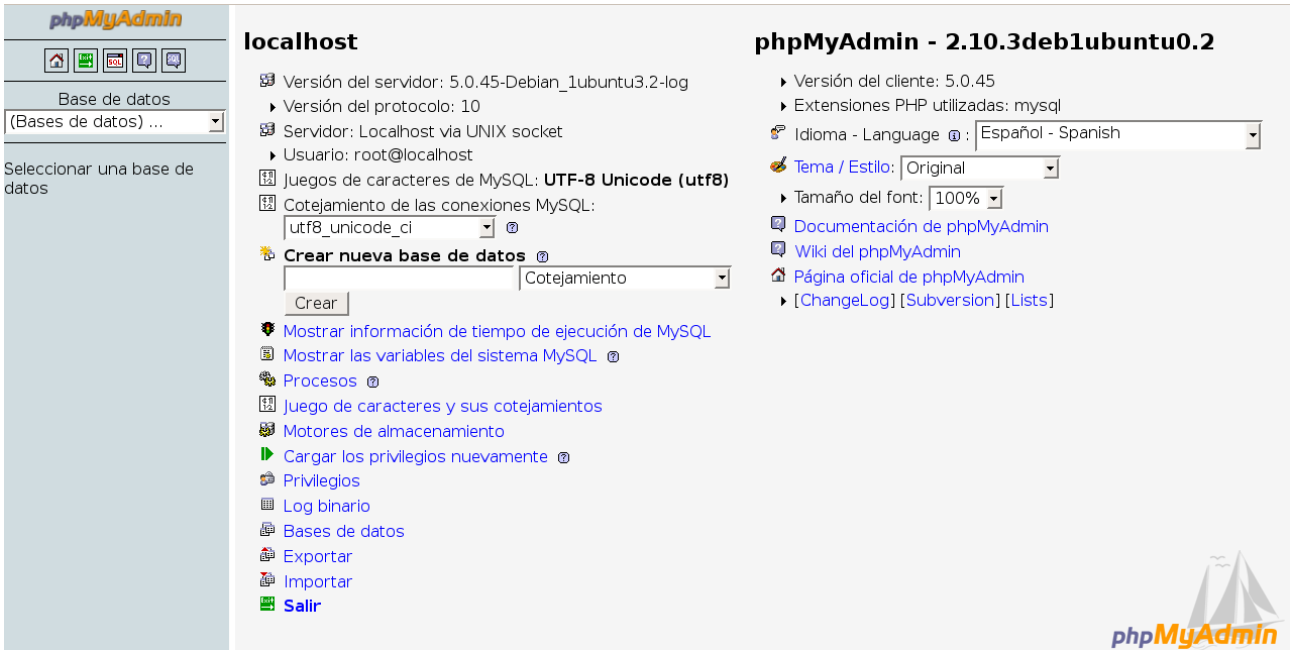
   Una altre forma és:

3. SQL. Fent servir el llenguatge SQL per crear les nostres bases de dades i taules.

1. MysSQL Administrator:



2. PHPMyAdmin:



Nosaltres ens centrarem en la connexió, inserció i modificació de dades a la base de dades, i no tant en el disseny d'aquesta.

5.2 Funcions de connexió i desconnexió a la base de dades

La funció que farem servir per a establir la connexió amb la base de dades és mysql_connect. La sintaxi d'aquesta funció és la següent:

```
int mysql_connect(string NomDelHost, string Usuari, string Password);
```

Aquesta funció rep tres cadenes de text:

1. `NomDelHost`: És el l'adreça web a la que volem connectar-nos. Pot ser `localhost` o qualsevol altre adreça web.
2. `Usuari`: És el nom de l'usuari que fa la connexió. Dins de MySQL es defineixen diferents perfils d'usuari amb permisos diferents en funció del seu perfil.
3. `Password`: És la contrasenya de l'usuari que fa la connexió.

I retorna `FALSE` si no s'ha pogut establir la connexió i l'identificador d'un enllaç amb la base de dades si la connexió ha estat possible i amb èxit.

L'enllaç que s'ha establert amb la base de dades continuarà actiu fins que finalitzi el programa PHP o fins que cridem a la funció mysql_close. Aquesta funció té la següent sintaxi:

```
bool mysql_close(int Enllaç)
```

Aquesta funció rep com paràmetre l'identificador de l'enllaç amb la base de dades i retorna `TRUE` si s'aconsegueix tancar l'enllaç amb èxit i `FALSE` en cas contrari.

5.3 Selecció d'una base de dades

A un motor de base de dades MySQL hi poden haver moltes bases de dades diferents. Quan creem l'enllaç amb la base de dades haurem de seleccionar amb quina d'aquestes bases de dades volem treballar. Per a fer-ho farem servir la funció `mysql_select_db` que té la següent sintaxi:

```
bool mysql_select_db(string NomBaseDades, int Enllaç)
```

Aquesta funció rep una cadena a on s'especifica el nom de la base de dades que volem fer servir i l'enllaç amb le qual ens hem connectat al motor MySQL. Retornarà `TRUE` si la selecció de la base de dades ha estat correcte i `FALSE` en cas contrari.

5.4 Funció genèrica de connexió i selecció de base de dades

Veurem ara com crear una funció genèrica que ens permetrà establir l'enllaç amb el motors de base de dades MySQL i seleccionar la base de dades que volem fer servir.

```php
<?php
function connexio() {
	// VARIABLES DE DEFINICIÓ
	$HOST = "HostAlQueEmConnecto";
	$USER = "ElMeuUsuari";
	$PASSWORD = "LaMevaPassword";
	$DATABASE = "NomDeLaBaseDeDades";

	// CONNEXIÓ AL MOTOR MYSQL
	if (!($link=mysql_connect($HOST,$USER,$PASSWORD))) {
		echo "Error de conexi&oacute; al motor MySQL";
		exit();
	}
	// SELECCIÓ DE LA BASE DE DADES
	if (!mysql_select_db($DATABASE,$link)) {
		echo "Error de selecci&oacute; de la base de dades";
		exit();
	}
	return $link;
}
?>
```

Aquesta funció ens permetrà crear un enllaç al motor de base de dades i seleccionar la base de dades amb la que volem treballar.

5.5 Manipulació de les dades de la base de dades

Per a manipular les dades de la base de dades hem de fer servir codi SQL. Per a enviar aquest codi SQL a la base de dades des de PHP farem servir la funció `mysql_query` que té la següent sintaxi:

```
recurs mysql_query (string Consulta, int Enllaç);
```

Aquesta funció rep els següents paràmetres:

1. `Consulta`: És una cadena que conté una acció sobre la base de dades en codi SQL.
2. `Enllaç`: És l'identificador de l'enllaç a la base de dades.

I retorna el següent:

1. Per a les accions SQL de tipus `INSERT`, `UPDATE` o `DELETE` retorna `TRUE` si l'acció s'ha realitzat amb èxit i `FALSE` en cas contrari.
2. Per a les accions SQL de tipus SELECT retorna `FALSE` si no s'ha realitzat amb èxit la selecció de dades o totes les dades resultants de la selecció.

5.6 Inserir registres a la base de dades

Suposem que tenim una base de dades anomenada `prueba` que conté una taula anomenada `usuaris` amb la següent estructura:

```
mysql> show tables;
+------------------+
| Tables_in_prueba |
+------------------+
| usuaris          |
+------------------+
1 row in set (0.00 sec)

mysql> describe usuaris;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| idUsuari     | int(11)      | NO   | PRI | NULL    | auto_increment |
| nomUsuari    | varchar(100) | NO   |     |         |                |
| cognomUsuari | varchar(250) | NO   |     |         |                |
+--------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

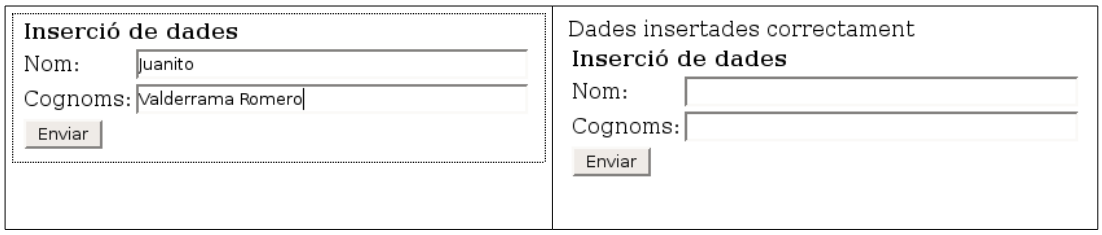
Construirem ara una pàgina que ens permetrà afegir dades a aquesta taula de la base de dades.

`insertar.php`:

```php
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() {
  // DEFINEIXO LES VARIABLES DE CONNEXIÓ
  $HOST = "localhost";
  $USER = "root";
  $PASSWORD = "";
  $DATABASE = "prueba";
  if (!($link=mysql_connect($HOST,$USER,$PASSWORD))) {
    echo "Error de conexi&oacute; a la base de dades";
    exit();
  }
  if (!mysql_select_db($DATABASE,$link)) {
    echo "Error de selecci&oacute;n de la base de dades";
    exit();
  }
  return $link;
}

// GESTIONO L'ENVIAMENT DE PARÀMETRES
if (isset($_POST['nom'])) {
  // L'USUARI M'HA ENVIAT LES DADES
  // HE D'INSERTAR AQUESTES DADES A LA BASE DE DADES

  // Construïm l'enllaç a la base de dades
  $enllac = connexio();

  // Construïm la cadena amb el codi SQL
  $insert = "INSERT INTO usuaris SET nomUsuari='" . $_POST['nom'] . "',
cognomUsuari='" . $_POST['cognoms'] . "'";
  // Observeu que ara la variables $insert té la següent cadena:
    // INSERT INTO usuaris SET nomusuari='ElNomIntroduit',
          cognomUsuari='ElCognomIntroduit'
  // Que com podeu observar és una sentència d'inserció SQL

  // Faig la inserció
  $esCorrecte = mysql_query($insert, $enllac);

  if ($esCorrecte) echo "Dades insertades correctament";
  if (!$esCorrecte) echo "Dades insertades err&ograve;niament";
}
?>
<html>
<head>
  <title>Inserci&oacute; de dades</title>
</head>
```

```
<body>
<form name="Dades" method="post" action="insertar.php">
<table>
  <tr>
    <td colspan="2"><b>Inserci&oacute; de dades</b></td>
  </tr>
  <tr>
    <td>Nom:</td>
    <td><input type="text" name="nom" size="40" maxlength="40"/></td>
  </tr>
  <tr>
  <td>Cognoms:</td>
    <td><input type="text" name="cognoms" size="40" maxlength="40"/></td>
  </tr>
  <tr>
    <td colspan="2"><input type="submit" value="Enviar"/></td>
  </tr>
</table>
</form>
</body>
</html>
```

El funcionament és el següent:



I observeu que hem introduït un nou usuari a la base de dades:

```
mysql> select * from usuaris;
+----------+-----------+--------------------+
| idUsuari | nomUsuari | cognomUsuari       |
+----------+-----------+--------------------+
|        2 | Juanito   | Valderrama Romero |
+----------+-----------+--------------------+
1 row in set (0.00 sec)
```

5.7 Seleccionar registres de la base de dades

Per a seleccionar registres de la base de dades farem servir, igual que a l'exemple d'inserció, la funció `mysql_query()`. La diferència resideix en que ara aquesta funció ens retornarà, si tot funciona de manera correcta, una taula amb el resultat de la consulta. La taula retornada no es podrà tractar de manera inmediata sinó que l'haurem de recorrer fila a fila i per això necessitarem la funció `mysql_fetch_array` que té la següent sintaxis:

```
array mysql_fetch_array(int ResultatConsulta);
```

Aquesta funció rep el resultat de la consulta (és a dir la taula de resultats) i ens retorna `FALSE` si ja no hi han més files i un array associatiu amb la fila corresponent.

La primera vegada que es crida a la funció `mysql_fetch_array()` ens retorna la primera fila de resultats, la segona vegada ens retorna la segona fila, i així successivament fins que completem tota la taula de resultats.

Una altre funció que ens pot resultar útils es la funció `mysql_num_rows` que ens dóna el nombre total de files que té la taula resultat de la selecció. La sintaxi d'aquesta funció és la següent:

```
int mysql_num_rows(int ResultatConsulta);
```

Aquesta funció rep com a paràmetre el resultat de la consulta (és a dir la taula de resultats) i ens retorna el nombre de files (registres) que té la consulta.

Veiem un exemple. Imaginem que a la taula anterior fem `SELECT * FROM usuaris ORDRE BY idUsuari` i obtenim el següent resultat:

```
mysql> SELECT * FROM usuaris ORDER BY idUsuari;
+----------+-----------+-------------------+
| idUsuari | nomUsuari | cognomUsuari      |
+----------+-----------+-------------------+
|        1 | Juan      | Gomez Perez       |
|        2 | Juanito   | Valderrama Romero |
|        3 | Fernando  | De la Rosa Morada |
+----------+-----------+-------------------+
3 rows in set (0.00 sec)
```

Ara el que volem és construir un PHP que accedeixi a la base de dades, realitzi la mateixa consulta que hem fet amb MySQL i mostri els resultats als navegador. Farem el següent:

seleccionar.php

```
<html>
<head>
<title>Selecci&oacute;</title>
</head>
<body>
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() {
  // DEFINEIXO LES VARIABLES DE CONNEXIÓ
  $HOST = "localhost";
  $USER = "root";
  $PASSWORD = "";
  $DATABASE = "prueba";
  if (!($link=mysql_connect($HOST,$USER,$PASSWORD))) {
    echo "Error de conexi&oacute; a la base de dades";
    exit();
  }
  if (!mysql_select_db($DATABASE,$link)) {
    echo "Error de selecci&oacute;n de la base de dades";
    exit();
  }
  return $link;
}
// Construim l'enllaç a la base de dades
$enllac = connexio();
// Construim la cadena amb el codi SQL
$seleccio = "SELECT * FROM usuaris ORDER BY idUsuari";
// Observeu que ara la variables $seleccio té la següent cadena:
      // SELECT * FROM usuaris ORDER BY idUsuari
// Que com podeu observar és una sentència de selecció SQL

// Faig la selecció
$ResultatConsulta = mysql_query($seleccio, $enllac);

// Si  $ResultatConsulta es FALSE hi ha hagut un error
if (!$ResultatConsulta) {
  echo "Error en la consulta a la base de dades";
  exit();
}

// Calculem el nombre de resultats
$totalRegistres = mysql_num_rows($ResultatConsulta);
?>
Hi ha un total de <?php echo $totalRegistres; ?> registres<br>
<?php
```

```php
// Mostrem els resultats
if ($totalRegistres == 0) {
  ?>No hi ha resultats<?php
} else {
  ?>
  <table>
  <tr><td>Nom</td><td>Cognom</td><td>Acci&oacute;</td></tr>
  <?php
  // AQUESTA ÉS LA SENTÈNCIA MÉS IMPORTANT.  COM ES POT VEURE
  // RECORREM EL RESULTAT DE LA CONSULTA FILA A FILA FINS QUE NO
  // HI QUEDIN MÉS I MOSTREM LES FILES FORMATADES EN UNA TAULA
  while ($filaResultat = mysql_fetch_array($ResultatConsulta)) {
    ?>
    <tr>
    <td><?php echo $filaResultat['nomUsuari']; ?></td>
    <td><?php echo $filaResultat['cognomUsuari']; ?></td>
    <td>
    <a href="modificar.php?id=<?php echo $filaResultat['idUsuari']; ?>">
                                                    Modificar</a></td> /
    <a href="esborrar.php?id=<?php echo $filaResultat['idUsuari']; ?>">
                                                    Esborrar</a></td>
    <tr>
    <?php
  }
  ?>
  </table>
  <?php
}
?>
</body>
</html>
```

El resultat d'aquest script PHP és:

Hi ha un total de 3 registres

Nom	Cognom	Acció
Juan	Gomez Perez	Modificar / Esborrar
Juanito	Valderrama Romero	Modificar / Esborrar
Fernando	De la Rosa Morada	Modificar / Esborrar

Hem incorporat dos links:

- Un anomenat `Modificar` que conté l'enllaç a la pàgina `modificar.php` passant el paràmetre `id` amb l'identificador de l'usuari corresponent. Això enllaça amb el següent apartat que és el que tracta de la modificació de dades.
- Un segon anomenat `Esborrar` que conté l'enllaç a la pàgina `esborrar.php` passant el paràmetre `id` amb l'identificador de l'usuari corresponent. Això enllaça amb l'apartat que tracta de l'esborrament de dades.

5.8 Modificar registres de la base de dades

En aquest apartat veurem com carregar les dades de la base de dades en un formulari i després realitzar les modificacions que ens demani l'usuari.

De moment tenim a la base de dades les següents dades:

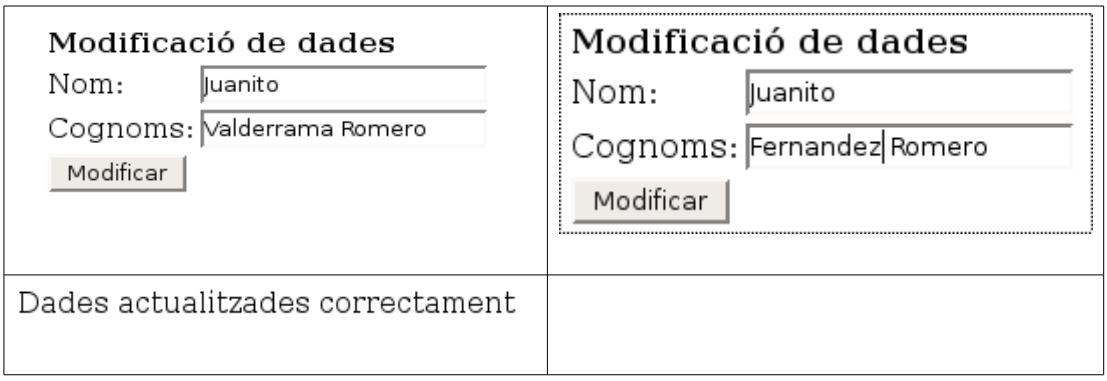
```
mysql> SELECT * FROM usuaris ORDER BY idUsuari;
+----------+-----------+-------------------+
| idUsuari | nomUsuari | cognomUsuari      |
+----------+-----------+-------------------+
|        1 | Juan      | Gomez Perez       |
|        2 | Juanito   | Valderrama Romero |
|        3 | Fernando  | De la Rosa Morada |
+----------+-----------+-------------------+
3 rows in set (0.00 sec)
```

Creem ara la pàgina `modificar.php`:

```
<html>
<head>
<title>Modificaci&oacute;</title>
</head>
<body>
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() {
  // DEFINEIXO LES VARIABLES DE CONNEXIÓ
  $HOST = "localhost";
  $USER = "root";
  $PASSWORD = "";
  $DATABASE = "prueba";
  if (!($link=mysql_connect($HOST,$USER,$PASSWORD))) {
    echo "Error de conexi&oacute; a la base de dades";
    exit();
  }
  if (!mysql_select_db($DATABASE,$link)) {
    echo "Error de selecci&oacute;n de la base de dades";
    exit();
  }
  return $link;
}
// Obtinc l'identificador de l'usuari que m'han enviat via GET
$identificador = $_GET['id'];

// Construim l'enllaç a la base de dades
$enllac = connexio();
// Comprovo si m'han enviat les dades del formulari
if (isset($_POST['nom'])) {
  // He de realitzar la modificació
  // Construeixo l'update
  $actualitzar = "UPDATE usuaris SET nomUsuari='" . $_POST['nom'] . "',";
  $actualitzar  .= "cognomUsuari='" . $_POST['cognoms'] . "' WHERE idUsuari = $identificador";
  // Observeu que ara la variables $actualitzar té la següent cadena:
    // UPDATE usuaris SET nomUsuari='NOM_MODIFICAT',
        cognomUsuari='COGNOM_MODIFICAT' WHERE idUsuari = Num_identificador
  // Que com podeu observar és una sentència de modificació SQL

  // Faig l'actualització
  $esCorrecte = mysql_query($actualitzar, $enllac);

  if ($esCorrecte) echo "Dades actualitzades correctament";
  if (!$esCorrecte) echo "Dades actualitzades err&ograve;niament";
  exit();
}
// No m'han enviat dades del formulari per tant
// He de carregar el formulari amb les dades de la base de dades

// Construim la cadena amb el codi SQL
$seleccio = "SELECT * FROM usuaris WHERE idUsuari = $identificador";
// Observeu que ara la variables $seleccio té la següent cadena:
       // SELECT * FROM usuaris WHERE idUsuari = Num_identificador_usuari
// Que com podeu observar és una sentència de selecció SQL

// Faig la selecció
$ResultatConsulta = mysql_query($seleccio, $enllac);

// Si  $ResultatConsulta es FALSE hi ha hagut un error
if (!$ResultatConsulta) {
  echo "Error en la consulta a la base de dades";
  exit();
}

// Obtinc la fila resultat de la consulta
$filaResultat = mysql_fetch_array($ResultatConsulta)

// Mostro el formulari omplert amb les dades
?>
<form method="post" action="modificar.php?id=<?php echo $identificador ?>">
<table>
  <tr>
```

```
    <td colspan="2"><b>Modificaci&oacute; de dades</b></td>
  </tr>
  <tr>
    <td>Nom:</td>
    <td><input type="text" name="nom" value="<?php echo $filaResultat['nomUsuari']; ?>">
    </td>
  </tr>
  <tr>
    <td>Cognoms:</td>
    <td><input type="text" name="cognoms" value="<?php echo $filaResultat['cognomUsuari']; ?>">
    </td>
  </tr>
  <tr>
    <td colspan="2"><input type="submit" value="Modificar"/></td>
  </tr>
</table>
</form>
<?php
?>
</body>
</html>
```

Si jo actualitzo l'usuari amb `idUsuari=2` enllaçaré amb `modificar.php?id=2`. En primer lloc ens apareixerà el formulari amb les dades actuals de l'usuari. En aquest formulari jo realitzaré les modificacions que cregui convenients com es veu a la imatge.



Una vegada hagi fet les modificacions les faré efectives a la base de dades amb el botó `Modificar`. I la base de dades contindrà el següent:

```
mysql> SELECT * FROM usuaris ORDER BY idUsuari;
+----------+-----------+-------------------+
| idUsuari | nomUsuari | cognomUsuari      |
+----------+-----------+-------------------+
|        1 | Juan      | Gomez Perez       |
|        2 | Juanito   | Fernandez Romero  |
|        3 | Fernando  | De la Rosa Morada |
+----------+-----------+-------------------+
3 rows in set (0.00 sec)
```

5.9 Esborrar registres de la base de dades

En aquest apartat veurem com esborrar registres de la base de dades. De moment tenim a la base de dades el contingut següent:

```
mysql> SELECT * FROM usuaris ORDER BY idUsuari;
+----------+-----------+-------------------+
| idUsuari | nomUsuari | cognomUsuari      |
+----------+-----------+-------------------+
|        1 | Juan      | Gomez Perez       |
|        2 | Juanito   | Fernandez Romero  |
|        3 | Fernando  | De la Rosa Morada |
+----------+-----------+-------------------+
3 rows in set (0.00 sec)
```

Creem ara l'arxiu `esborar.php`:

```
<html>
<head>
<title>Esborrar</title>
</head>
<body>
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() {
  // DEFINEIXO LES VARIABLES DE CONNEXIÓ
  $HOST = "localhost";
  $USER = "root";
  $PASSWORD = "";
  $DATABASE = "prueba";
  if (!($link=mysql_connect($HOST,$USER,$PASSWORD))) {
    echo "Error de conexi&oacute; a la base de dades";
    exit();
  }
  if (!mysql_select_db($DATABASE,$link)) {
    echo "Error de selecci&oacute;n de la base de dades";
    exit();
  }
  return $link;
}
// Obtinc l'identificador de l'usuari que m'han enviat via GET
$identificador = $_GET['id'];

// Construim l'enllaç a la base de dades
$enllac = connexio();

// Construeixo el delete
$esborrar = "DELETE FROM usuaris WHERE idUsuari = $identificador";
// Observeu que ara la variables $esborrar té la següent cadena:
       // DELETE FROM usuaris WHERE idUsuari = Identificador_de_usuari
// Que com podeu observar és una sentència d'esborrament SQL

// Faig l'esborrament
$esCorrecte = mysql_query($esborrar, $enllac);

if ($esCorrecte) echo "Dades esborrades correctament";
if (!$esCorrecte) echo "Dades esborrades err&ograve;niament";
?>
</body>
</html>
```

Si ara vull esborrar l'usuari amb `idUsuari=3` enllaçaré amb `esborrar.php?id=3`. I això farà que a la base de dades quedin els següents registres:

```
mysql> SELECT * FROM usuaris ORDER BY idUsuari;
+----------+-----------+------------------+
| idUsuari | nomUsuari | cognomUsuari     |
+----------+-----------+------------------+
|        1 | Juan      | Gomez Perez      |
|        2 | Juanito   | Fernandez Romero |
+----------+-----------+------------------+
2 rows in set (0.00 sec)
```

5.10 Paginació de resultats

Quan fem una selecció sobre la base de dades podem obtenir una gran quantitat de resultats. Mostrar tots aquests resultats en una mateixa pàgina no seria massa útil, per tant és necessari mostrar aquests resultats ordenats en diferents pàgines.

Per a fer la paginació farem servir la clàusula LIMIT de MySQL. La seva sintaxi és: `SELECT dades`

`FROM taula WHERE condicions LIMIT registreInici, quantitatRegistres`.

La sentència anterior ens retorna `quantitatRegistres` registres des de la posició `registreInici+1` de les dades seleccionades. Per exemple si fem: `SELECT * FROM usuaris ORDER BY cognomUsuari LIMIT 5, 10`, obtindrem 10 registres de de la taula usuaris començant pel registre 6è ordenats pel cognom de l'usuari. Si, contant des del registre 6è no hi ha 10 registres més a la base de dades, la consulta anterior ens retorna tots els que hi hagi.

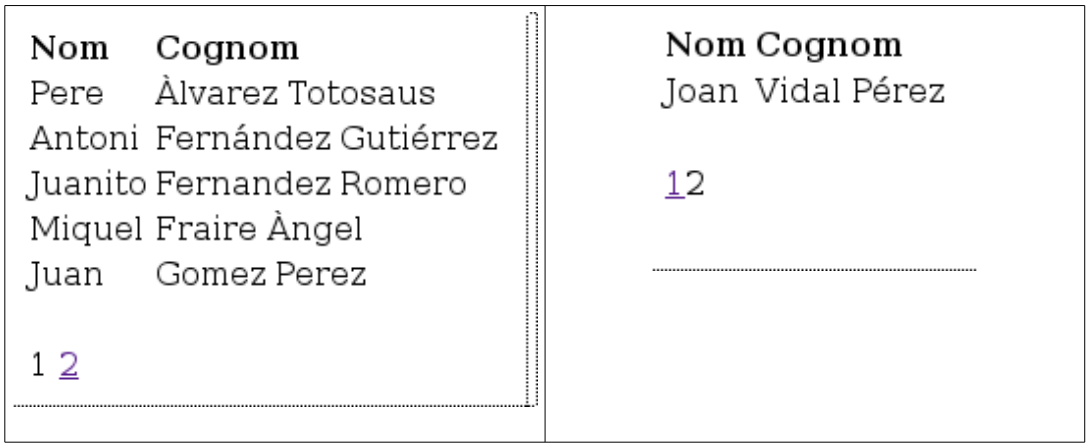
La única funció especial de PHP que necessitarem per a fer la paginació és la funció `ceil()`. Aquesta funció rep un valor numèric i ens retorna el següent valor sencer més gran que el que ha rebut com a paràmetre. És a dir `ceil(4.3)` és igual a `5`, o `ceil(5.8)` és igual a `6`.

Veiem ara com fer la paginació a través d'un exemples que ens permet obtenir els resultats de 5 en 5. La pàgina s'anomena `paginacio.php`:

```php
<html>
<head><title>Paginaci&oacute;</title></head>
<body>
<?php
/*EXEMPLE DE PAGINACIÓ */
function connexio() { ... }

// Enllaço amb la base de dades
$link = connexio();

// Variable que conte els registres per pagina que volem veure
$registresPerPagina = 5;

// Mirem quantes pàgines de 5 registres hi ha
$seleccio = "SELECT count(*) AS totalRegistres FROM usuaris";
$resultat = mysql_query($seleccio,$link);
if (!$resultat) {
        echo "Error de selecci&oacute; de dades";
        exit();
}
$filaResultat = mysql_fetch_array($resultat);
$totalRegistres = $filaResultat['totalRegistres'];
$totalPagines = ceil($totalRegistres / $registresPerPagina);

// Mirem quina és la pàgina que volem carregar
if (!isset($_GET['p'])) { // No m'han passat la pàgina, per tant és la primera
        $paginaActual = 1;
} else {
        $paginaActual = $_GET['p'];
}

// Ara ja podem construir la selecció a la base de dades
$seleccio = "SELECT * FROM usuaris ORDER BY cognomUsuari LIMIT " . ($paginaActual-1)*
                                        $registresPerPagina . "," . $registresPerPagina;
//  Observeu amb atenció la clàusula LIMIT
// Si volem carregar la pàgina 1 tindrem LIMIT 0,5 - Registres 1,2,3,4,5
// Si volem carregar la pàgina 2 tindrem LIMIT 5,5 - Registres 6,7,8,9,10
// Si volem carregar la pàgina 3 tindrem LIMIT 10,5 - Registres 11,12,13,14,15. etc...
$resultat = mysql_query($seleccio,$link);
if (!$resultat) {
        echo "Error en la selecci&oacute; de dades";
        exit();
}
// Mostrem els resultats
?>
<table>
<tr><td><b>Nom</b></td><td><b>Cognom</b></td></tr>
<?php
while ($filaResultat = mysql_fetch_array($resultat)) {
        ?>
        <tr><td><?php echo $filaResultat['nomUsuari']; ?></td><td><?php echo
                                                $filaResultat['cognomUsuari']; ?></td></tr>
        <?php
}
?>
```

```
<tr><td colspan="2"> </td></tr>
<tr>
        <td colspan="2">
<?php

// Ara mostro les diferents pàgines a carregar
for ($i=1; $i<=$totalPagines; $i++) {
        if ($i == $paginaActual) {
                echo "$i ";
        } else {
                ?><a href="paginacio.php?p=<?php echo $i; ?>"><?php echo $i; ?></a><?php
        }
}
?>
        </td>
<tr>
</table>
</body>
</html>
```

Així el resultat és el següent:



5.11 Pujar fitxers al servidor

Veiem ara com podem pujar fitxers al nostre servidor. Aquesta és una de les funcionalitats més importants per tal d'omplir de contingut i millorar l'aparència del nostre lloc web. Veiem què necessitem per tal de pujar un fitxer al nostre lloc web.

5.11.1 Configuració HTML

De la part de l'HTML serà necessari incloure els següents elements:

- Afegir `enctype="multipart/form-data"` a l'etiqueta del formulari. Així doncs l'etiqueta del formulari quedarà similar a la següent:

  ```
  <form name="form1" action="pagina.php" method="post" enctype="multipart/form-data">
  ```

- Afegir un control de selecció de fitxers al formulari. Aquests controls són de la forma:

  ```
  <input type="file" name="nom_control_arxiu" accept="tipus_MIME_d'arxius_acceptats">
  ```

La llista de tipus MIME d'arxius acceptats és la següent:

Tipus MIME	Extensió
Imatges	
image/bmp	.bmp
image/x-windows-bmp	.bm
image/gif	.bmp
image/jpeg	.gif

image/jpeg	.jpe
image/png	.jpg / .png
So	
audio/basic	.au / .snd
audio/x-au	.au
audio/midi	.mid
audio/x-midi	.midi
audio/x-wav	.mid
audio/mod	.midi
audio/x-mod	.wav
audio/mpeg3	.mod
audio/x-mpeg3	.mod
audio/x-mpeg-3	.mp3
audio/x-pn-realaudio	.ra / .ram
Video	
video/avi	.avi
video/x-motion-jpeg	.mjpg
video/quicktime	.mov
video/mpeg	.mpg
application/x-shockwave-flash	.swf

Si, per exemple, voleu incloure totes les diferents imatges podeu especificar `accept="image/*"`. A internet trobareu una llista molt més completa de tipus MIME per aplica-ho als vostres formularis.

5.11.2 Configuració PHP – Servidor

Primer hem de configurar l'arxiu `php.ini` per tal d'admetre pujar fitxers i la mida màxima d'aquests. Per a fer-ho editeu l'arxiu i cerqueu les entrades:

- `file_uploads`: Indica a PHP si es poden, o no, pujar fitxers al servidor. Per tal d'acceptar fitxers haurem de posar el valor On en aquesta variable.
- `upload_max_filesize`: Indica a PHP quina és la mida màxima dels arxius que pot acceptar. Es pot fer servi el sufix M per tal d'indicar el valor en MegaBytes. Per exemple, amb un valor 2M s'acceptaran arxius de com a màxim 2 MegaBytes.
- `upload_tmp_dir`: Indica el directori en el que es copien temporalment els fitxers quan es pujen al servidor.

Ara ja podrem pujar fitxers al nostre servidor. Imaginem que hem creat al nostre formulari la següent entrada: `<input type="file" name="fitxer" accept="image/*">`. Això admet tot tipus de fitxers de tipus imatges. Imaginem ara que l'`action` del formulari ens envia a la pàgina `gestionaFitxer.php`. Dins d'aquesta pàgina es crearà l'array associatiu `$_FILES` que contindrà els diferents arxius que s'hagin carregat. Per accedir al nostre farem:

- `$_FILES['fitxer']['name']`. Per a obtenir el nom del fitxer.
- `$_FILES['fitxer']['type']`. Per a obtenir el tipus MIME del fitxer.
- `$_FILES['fitxer']['tmp_name']`. Per a obtenir la ruta i el nom temporals de l'arxiu pujat.
- `$_FILES['fitxer']['size']`. Per a obtenir la mida de l'arxiu en bytes.

Dins de la pàgina de gestió del fitxer podrem fer servir les següents funcions:

- `bool is_uploaded_file($_FILES['fitxer_pujat']['tmp_name'])`. Ens diu si hem pujat el fitxer (retorna TRUE) o no (retorna FALSE).
- `bool mkdir(String nom_directori_a_crear, int mode)`: Ens permet crear un directori al servidor. Retorna TRUE si l'ha pogut crear correctament i FALSE en cas contrari. El mode són els permisos que volem que tingui el directori. S'han de posar en octal (fet que vol dir que han de començar per 0) i especifica el permís amb 3 números tal i com es fa a Linux. Per exemple, si volem aplicar tots els permisos farem 0777.
- `bool copy(String origen, String destí)`: El que fa aquesta funció és copiar l'arxiu que es troba en la localització especificada per la cadena `origen` a la localització especificada per la cadena `destí`. Retorna TRUE si la còpia es fa de manera correcta i FALSE si la còpia no s'ha realitzat amb correcció.
- `bool move_uploaded_file($_FILES['nom_arxiu']['tmp_name'], String destí)`: El que fa aquesta funció és moure l'arxiu pujat a la carpeta del servidor indicada per la cadena `destí` comprovant que, en efecte, és un arxiu carregat vàlid. Retorna TRUE si l'operació es fa de manera correcta i FALSE si no s'ha realitzat amb correcció.
- `bool unlink(String arxiu)`: Aquesta funció és esborrar l'arxiu que es troba en la localització especificada per la cadena `arxiu`. Retorn TRUE si l'esborra correctament i FALSE si no ho fa.
- `bool rmdir(String directori)`: Aquesta funció és esborrar el directori que es troba en la localització especificada per la cadena `directori`. Retorn TRUE si l'esborra correctament i FALSE si no ho fa.
- `bool is_dir(String directori)`: Aquesta funció indica si la cadena `directori` apunta realment a un directori o no. Retorna TRUE si és certa i FALSE en cas contrari.

Tingueu en compte les següents consideracions generals:

- Només podreu copiar o esborrar fitxers o directoris dins de carpetes del servidor en les que tingueu els permisos adients. Per tant, sempre que vulgueu pujar fitxers al servidor o crear directoris feu una carpeta al servidor a on poder fer-ho i doneu els permisos 777 a aquesta carpeta.
- MySQL admet crear camps que contenen fitxers. Això pot carregar la base de dades i només és útil si vols fer cerques dins dels documents. El més habitual es emmagatzemar el fitxer en una carpeta del servidor i guardar a la base de dades la ruta d'accés al fitxer en el servidor i NO el fitxer. Per exemple si creem una carpeta anomenada `descarregues` i pugem un fitxer anomenat `dades.doc` a aquesta carpeta, la base de dades tindrà un camp que contindrà `descarregues/dades.doc`.
- Una de les maneres de guardar els fitxers es crear una carpeta associada per a cada registre i guardar dins d'aquesta carpeta els fitxers associats a aquest registre. ATENCIÓ: Si guardeu el fitxers amb els noms originals quan l'usuari pugi un fitxer amb el mateix nom que un existent al servidor, aquest es substituirà pel nou.
    - Com que la carpeta ha de pertànyer al registre i l'element que identifica unívocament el registre és el seu `id`, el més habitual és crear una carpeta amb que tingui per nom l'`id` del registre.

Imaginem ara que la nostra base de dades `prueba` conté la taula usuaris amb la següent estructura:

```
mysql> describe usuaris;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| idUsuari     | int(11)      | NO   | PRI | NULL    | auto_increment |
| nomUsuari    | varchar(100) | NO   |     |         |                |
| cognomUsuari | varchar(250) | NO   |     |         |                |
| fitxer       | varchar(150) | NO   |     |         |                |
+--------------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

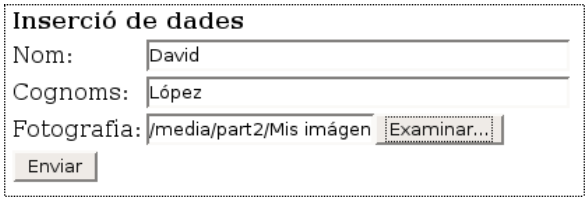
Al camp `fitxer` posarem la ruta en la qual es trobarà el nostre fitxer. I que al servidor hem creat una carpeta amb el nom `fotosUsuaris` amb els permisos 777.

Ara podem crear el següent codi

`gestionaFitxer.php`:

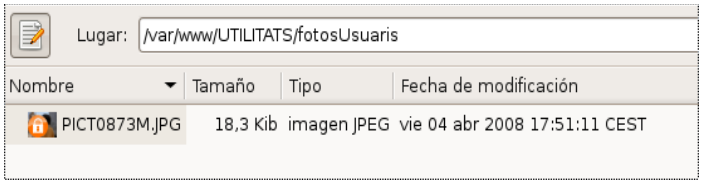
```php
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() { ... }

// GESTIONO L'ENVIAMENT DE PARÀMETRES
if (isset($_POST['nom'])) {
        // L'USUARI M'HA ENVIAT LES DADES
        // HE D'INSERTAR AQUESTES DADES A LA BASE DE DADES

        // Construim l'enllaç a la base de dades
        $enllac = connexio();

        // Comprovem si han pujat el fitxer
        if (is_uploaded_file($_FILES['foto']['tmp_name'])) {
                // Han pujat el fitxer
                // Copio el fitxer a la carpets fotosUsuaris
                if (!copy($_FILES['foto']['tmp_name'],"fotosUsuaris/" . $_FILES['foto']['name']))
                {
                        echo "No he pogut copiar el fitxer";
                        exit();
                }
                // Creem la variable per a inserir a la base de dades
                $fitxer = "fotosUsuaris/" . $_FILES['foto']['name'];
        } else {
                // No han pujat el fitxer
                $fitxer = "";
        }

        // Construim la cadena amb el codi SQL
        $insert = "INSERT INTO usuaris SET nomUsuari='" . $_POST['nom'] . "', “;
        $insert .= "cognomUsuari='" . $_POST['cognoms'] . "', fitxer='$fitxer'";

        // Faig la inserció
        $esCorrecte = mysql_query($insert, $enllac);

        if ($esCorrecte) echo "Dades insertades correctament";
        if (!$esCorrecte) echo "Dades insertades err&ograve;niament";
}
?>
<html>
<head>
        <title>Inserci&oacute; de dades</title>
</head>
<body>
<form name="Dades" method="post" action="gestionaFitxer.php" enctype="multipart/form-data">
<table>
        <tr>
                <td colspan="2"><b>Inserci&oacute; de dades</b></td>
        </tr>
        <tr>
                <td>Nom:</td>
                <td><input type="text" name="nom" size="40" maxlength="40"/></td>
        </tr>
        <tr>
                <td>Cognoms:</td>
                <td><input type="text" name="cognoms" size="40" maxlength="40"/></td>
        </tr>
        <tr>
                <td>Fotografia:</td>
                <td><input type="file" accept="image/*" name="foto" class="txtNormal"></td>
        </tr>
        <tr>
                <td colspan="2"><input type="submit" value="Enviar"/></td>
        </tr>
</table>
</form>
</body>
</html>
```

Com es pot veure, al codi el que fem és comprovar si el fitxer s'ha pujat o no, en el cas que s'hagi pujat el que farem serà copiar-lo a la carpeta `fotosUsuaris` amb el seu nom original. Veieu un exemple:



I després de la inserció:

```
mysql> select * from usuaris;
+----------+-----------+--------------+----------------------------+
| idUsuari | nomUsuari | cognomUsuari | fitxer                     |
+----------+-----------+--------------+----------------------------+
|       13 | David     | L?pez        | fotosUsuaris/PICT0873M.JPG |
+----------+-----------+--------------+----------------------------+
1 row in set (0.00 sec)
```



5.11.3 Gestió completa de fitxers

En aquest apartat millorarem l'exemple anterior creant una carpeta per al registre insertat a on guardar els arxius corresponents i farem un *script php* per a esborrar fitxers.

Per a fer la primera part necessitarem la funció `LAST_INSERT_ID()` de MySQL per tal d'obtenir l'identificador de l'últim registre insertat (sempre que la clau sigui autonumèrica).

Millorar de l'exemple anterior:

```php
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() { ... }

// GESTIONO L'ENVIAMENT DE PARÀMETRES
if (isset($_POST['nom'])) {
        // L'USUARI M'HA ENVIAT LES DADES
        // Construim l'enllaç a la base de dades
        $enllac = connexio();

        // Construim la cadena amb el codi SQL
        $insert = "INSERT INTO usuaris SET nomUsuari='" . $_POST['nom'] . "'";
        $insert .= ", cognomUsuari='" . $_POST['cognoms'] . "'";
        // Faig la inserció
        $esCorrecte = mysql_query($insert, $enllac);

        if ($esCorrecte) {
          // Obtenim l'últim ID
          $select = "SELECT LAST_INSERT_ID() AS ultimID FROM usuaris";
          $resultat = mysql_query($select, $enllac);
          if (!$resultat) {
            echo "Selecci&oacute; err&ograve;nia";
            exit();
          }
          $row = mysql_fetch_array($resultat);



          // Comprovem si han pujat el fitxer
          if (is_uploaded_file($_FILES['foto']['tmp_name'])) {
            // Han pujat el fitxer
          // Creem el directori si no existeix
          if (!is_dir("fotosUsuaris/" . $row['ultimID'])) {
```

```
            mkdir("fotosUsuaris/" . $row['ultimID'],0777);
          }
          // Copio el fitxer a la carpeta
          $nomTempFitxer = $_FILES['foto']['tmp_name'];
          $nomFitxer = $_FILES['foto']['name'];
          if (!copy($nomTempFitxer,"fotosUsuaris/" . $row['ultimID'] . "/" . $nomFitxer)) {
            echo "No he pogut copiar el fitxer";
            exit();
          }
            // Creem la variable per a inserir a la base de dades
            $fitxer = "fotosUsuaris/" . $row['ultimID'] . "/" . $nomFitxer;
          } else {
            // No han pujat el fitxer
            $fitxer = "";
          }

          // Construim la cadena amb el codi SQL
          $update = "UPDATE usuaris SET fitxer='$fitxer' WHERE idUsuari = " . $row['ultimID'];

          // Faig l'actualització
          $esCorrecte = mysql_query($update, $enllac);
          if (!$esCorrecte) {
            echo "Actualitzaci&oacute; incorrecta";
            exit();
          }
        } else {
          echo "Dades insertades err&ograve;niament";
        }
}
```

Ens quedarà:

```
mysql> select * from usuaris;
+----------+-----------+-----------------+-----------------------------+
| idUsuari | nomUsuari | cognomUsuari    | fitxer                      |
+----------+-----------+-----------------+-----------------------------+
|       21 | Jose      | Bueno Rodriguez | fotosUsuaris/21/naranja.jpg |
|       20 | Juan      | Gonzelez Perea  | fotosUsuaris/20/azul.jpg    |
|       19 | David     | Lopez Calder?n  | fotosUsuaris/19/amarillo.jpg |
+----------+-----------+-----------------+-----------------------------+
3 rows in set (0.00 sec)
```

Llistat dels fitxers i opció d'esborrar:

```
<html>
<head>
<title>Selecci&oacute;</title>
</head>
<body>
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() {...}
// Construim l'enllaç a la base de dades
$enllac = connexio();
// Construim la cadena amb el codi SQL
$seleccio = "SELECT * FROM usuaris ORDER BY idUsuari";

// Faig la seleccio
$ResultatConsulta = mysql_query($seleccio, $enllac);

// Si  $ResultatConsulta es FALSE hi ha hagut un error
if (!$ResultatConsulta) {
        echo "Error en la consulta a la base de dades";
        exit();
}

// Calculem el nombre de resultats
$totalRegistres = mysql_num_rows($ResultatConsulta);
?>
Hi ha un total de <?php echo $totalRegistres; ?> registres<br>
<?php

// Mostrem els resultats
```

```php
if ($totalRegistres == 0) {
        ?>No hi ha resultats<?php
} else {
        ?>
        <table>
        <tr><td>Nom</td><td>Cognom</td><td>Foto</td><td>Acci&oacute;</td></tr>
        <?php
        while ($filaResultat = mysql_fetch_array($ResultatConsulta)) {
                ?>
                <tr>
                <td><?php echo $filaResultat['nomUsuari']; ?></td>
                <td><?php echo $filaResultat['cognomUsuari']; ?></td>
                <td><?php
                   if ($filaResultat['fitxer'] != '') {
                      ?><img src="<?php echo $filaResultat['fitxer']; ?>" width="150"><?php
                   } else {
                      echo " ";
                   }
                 ?></td>
                <td><a href="esborrar.php?id=<?php echo $filaResultat['idUsuari']; ?>">
                                                                    Esborrar Fitxer</a></td>
                <tr>
                <?php
        }
        ?>
        </table>
        <?php
}
?>

</body>
</html>
```

Ens dóna com a resultat:

Hi ha un total de 3 registres

Nom	Cognom	Foto	Acció
David	Lopez Calderón		Esborrar Fitxer
Juan	Gonzelez Perea		Esborrar Fitxer
Jose	Bueno Rodriguez		Esborrar Fitxer

Esborrar fitxers:

```php
<?php
// DEFINEIXO LA FUNCIÓ DE CONNEXIÓ
function connexio() {...}
// Obtinc l'identificador de l'usuari que m'han enviat via GET
$identificador = $_GET['id'];

// Construim l'enllaç a la base de dades
$enllac = connexio();

// Construim la cadena amb el codi SQL
$seleccio = "SELECT fitxer FROM usuaris WHERE idUsuari = $identificador";

// Faig la seleccio
$ResultatConsulta = mysql_query($seleccio, $enllac);



// Si  $ResultatConsulta es FALSE hi ha hagut un error
if (!$ResultatConsulta) {
        echo "Error en la consulta a la base de dades";
        exit();
}
$fila = mysql_fetch_array($ResultatConsulta);
$RutaFitxer = $fila['fitxer'];
$esborrarOK = true;
if ($RutaFitxer != "") {
```

```
        // Hi ha un fitxer
        $esborrarOK = unlink($RutaFitxer);
}

if ($esborrarOK) echo "Fitxer esborrat correctament";

// Construim la cadena amb el codi SQL
$update = "UPDATE usuaris SET fitxer='' WHERE idUsuari = $identificador";

// Faig l'actualització
$esCorrecte2 = mysql_query($update, $enllac);
if (!$esCorrecte2) {
        echo "Actualitzaci&oacute; incorrecta";
        exit();
}
?>
```

5.12 Identificació d'usuaris

Per a realitzar la identificació d'usuaris farem servir les sessions. En primer lloc hem de construir una taula d'usuaris registrats en la qual han d'aparèixer, entre d'altres, els camps de:

- login: Identificador únic d'usuari.
- password: Contrasenya d'accés de l'usuari.

Per qüestions de seguretat el més recomanable és NO emmagatzemar la password directament a la base de dades, sinó aplicar-li una funció resumidora. Una de les funcions que més es fa servir és la MD5. Aquesta funció existeix en MySQL i en PHP. Així doncs no emmagatzemarem la contrasenya tal qual sinó l'md5 de la contrasenya que ocupara 32 caràcters SEMPRE.

Així doncs per a identificar un usuari haurem de:

- Tenir una pàgina de login en la qual demanem a l'usuari el login i la password.
- Comprovar si l'usuari és o no a la base de dades.
    - Si l'usuari és a la base de dades crearem una variable de sessió que indiqui que l'usuari que fa ús de la sessió està identificat.
    - Si l'usuari no és a la base de dades o no està identificat tornarem a carregar la pàgina de login.
- Recordeu fer sempre una pàgina de desconnexió del sistema, és molt perillós que una sessió es quedi oberta.