Programació amb HTML i PHP a través d'exemples

PID_00243569

Manel Díaz Llobet

Temps mínim de dedicació recomanat: 3 hores

Universitat Oberta de Catalunya

Manel Díaz Llobet

Índex

Introducció

El contingut d'aquest mòdul didàctic està orientat a aprendre a desenvolupar sistemes web amb interfícies de qualitat i segures per als usuaris.

Es comença mostrant com es defineixen formularis en les pàgines HTML de client. Aquests formularis són la manera d'entrar dades en un context web i es fan servir per tal de proporcionar, entre d'altres, informació al servidor perquè pugui accedir a la base de dades i realitzar així la funcionalitat demanada. Seguidament, es veurà com s'accedeix des del servidor a les dades enviades mitjançant un formulari de client utilitzant PHP. Finalment, es farà una petita consideració sobre la seguretat dels sistemes web i es donarà una sèrie de directrius per a dotar d'un nivell de seguretat adequat les aplicacions web que es desenvolupin.

Abans de començar amb els formularis, però, és necessari explicar què hem de fer a HTML i a PHP per a no tenir problemes amb els potencials caràcters especials que puguin tenir les diferents llengües a què doni suport el nostre web, com per exemple la "ñ" en castellà, la "ç" en català o les vocals accentuades.

1. Caràcters especials

En aquest apartat expliquem els caràcters especials en HTML i PHP.

1) Caràcters especials en HTML. Els caràcters especials, com per exemple la "ç", la "ñ" o les vocals accentuades, no es mostren de forma correcta en HTML per defecte.

Perquè el HTML tingui en compte que s'utilitzaran caràcters especials cal afegir una metadada de tipus "content-type" a la capçalera del fitxer HTML (dintre de l'etiqueta <head>). Un exemple el tenim en el codi següent:

```
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
```

Aquesta línia de codi indica el "content type", o sigui, el conjunt de caràcters a utilitzar per HTML. En aquest cas, el conjunt de caràcters és "utf-8", que conté els caràcters especials necessaris.

2) Caràcters especials en PHP. En el cas de programar fitxers PHP, també es pot indicar el tipus de caràcters que s'utilitzaran fent servir la metadada "content-type". Per a definir-la s'ha d'afegir la següent instrucció al començament del fitxer:

```
header("Content-Type: text/html;charset=utf-8");
```

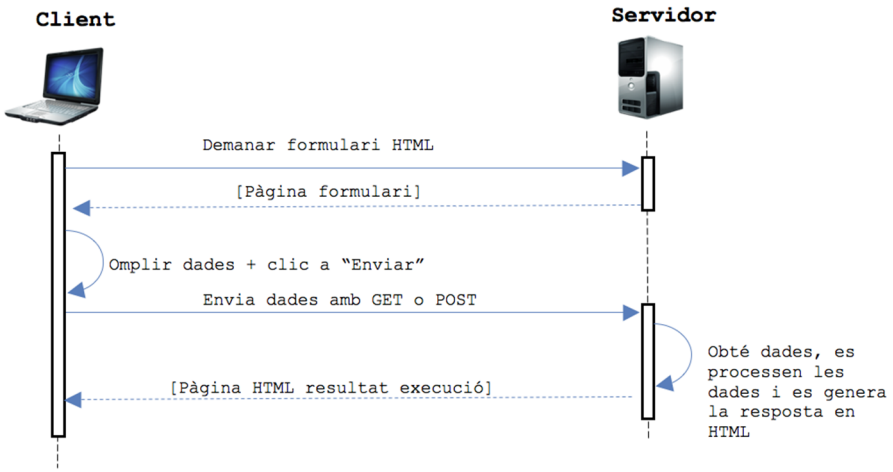
Amb aquestes dues instruccions, garantirem que es mostrin correctament tots els caràcters especials que es facin servir tant a HTML com a PHP.

2. Ús de formularis HTML

Quan es demana l'alta en un sistema de correu electrònic (Gmail o Hotmail, per exemple), cal indicar les dades de l'usuari. Quan es vol accedir via web a una part d'una aplicació web privada, cal introduir la informació d'autenticació d'usuari, sovint amb nom d'usuari i paraula de pas. En aquests dos casos, la tècnica utilitzada són els formularis HTML.

Un formulari HTML es construeix a partir d'un conjunt de sentències HTML i permet a l'usuari introduir una gran quantitat de dades, de diversos formats, per a ser tractades per un sistema. Aquest tractament pot ser molt variat, des d'inserir-les a la base de dades, modificar-les o eliminar-les fins a consultar-les.

La seqüència de passos simplificada que es realitzen en fer servir un formulari web és la següent:



L'ús de formularis HTML/XHTML és molt comú en el desenvolupament de sistemes web. Els formularis HTML/XHTML es basen en l'etiqueta `<form>`.

2.1. Estructura d'un formulari: etiqueta `<form>`

Un formulari en HTML no és més que un conjunt de sentències HTML delimitades per les etiquetes `<form>` i `</form>`. Entre aquestes hi haurà diverses etiquetes `<input>` de diferents tipus, que serviran per a definir les dades que s'esperen recollir en el formulari. Per cada etiqueta <input> es crearà un camp per a recollir les dades associades. Aquest camp podrà ser de diferents tipus, des d'un camp per a entrar text o un camp que permeti seleccionar entre diferents opcions fins a camps que permetin marcar "cert" o "fals" diferents opcions.

Un formulari no serviria de res si, una vegada introduïdes les dades, aquestes no es poden enviar al servidor per a ser processades i generar un resultat per a l'usuari. Per tal de poder realitzar aquest enviament de dades, cal indicar al formulari el seu destí (és a dir, on es troba el programa que en processarà les dades) i el mètode que es vol fer servir per a fer l'enviament de dades, que pot ser de dos tipus: GET o POST.

L'estructura general d'un formulari és la següent:

```
<form action="fitxer_servidor" method="mètode_envia_dades">
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 . . .
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 <input type="submit" value="text_botó"/>
</form>
```

Les diferents parts d'un formulari són les següents:

```
<form action="fitxer_servidor" method="mètode_envia_dades">
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 . . .
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 <input type="submit" value="text_botó"/>
</form>
```

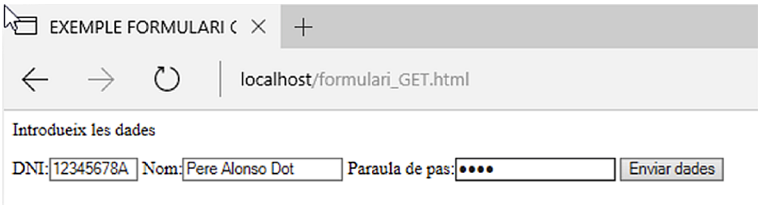
Nom el fitxer al servidor (és un fitxer escrit amb un llenguatge de servidor: PHP, ASP o qualsevol altre) que rebrà la informació del formulari i farà el tractament necessari sobre les dades.

```
<form action="fitxer_servidor" method="mètode_envia_dades">
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 . . .
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 <input type="submit" value="text_botó"/>
</form>
```

Mètode que farà servir el formulari per a enviar les dades al servidor. Hi ha dos mètodes:

1) GET: la informació introduïda per l'usuari al formulari s'envia de forma visible, concatenant parells ***clau=valor*** en la URL de la pàgina de destí. És un mètode que s'ha d'evitar si enviem informació sensible, ja que la informació introduïda per l'usuari es veu al navegador web.

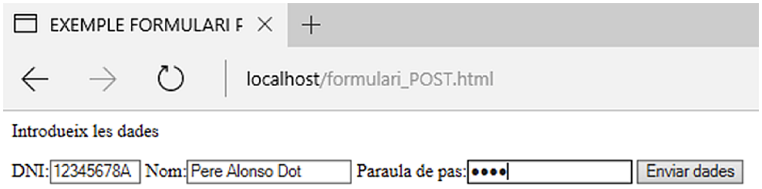
Al següent exemple es veu un formulari que envia les seves dades pel mètode GET:



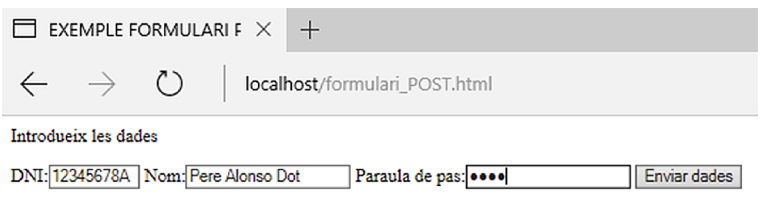
El resultat en fer clic al botó "Enviar dades" és el següent (assumim que el destí és un programa fet en PHP que mostra per pantalla les dades rebudes):

Com es pot veure, a la URL es mostren totes les dades del formulari. Això és molt insegur, perquè qualsevol podria veure-les i modificar-les.

2) POST: la informació introduïda per l'usuari no s'envia a la URL de destí, dintre de la petició HTTP. D'aquesta forma, la informació no és directament visible des de la URL de destí. Al següent exemple es veu un formulari que envia dades pel mètode POST:



El resultat en fer clic al botó "Enviar dades" és el següent:



Com es pot veure, en aquest cas, les dades no es mostren a la URL.

2.2. Entrada de dades simples: etiqueta <input>

Les etiquetes <input> defineixen les parts visibles d'un formulari, és a dir, la seva interfície amb l'usuari. Cada etiqueta <input> representa una dada que es vol obtenir de l'usuari. Hi ha diversos tipus de controls o components gràfics que es poden mostrar en un formulari, depenent del tipus d'etiqueta <input> que es faci servir.

```
<form action="fitxer_servidor" method="mètode_envia_dades">
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 . . .
 <input type="Tipus_entrada_dades" name="Nom_entrada_dades" />
 <input type="submit" value="text_botó"/>
</form>
```

L'atribut més important de l'etiqueta <input> és l'atribut "type" (en el codi anterior marcat en color blau). Aquest atribut és obligatori i indica el tipus de control que s'utilitzarà per a obtenir les dades de l'usuari. Depenent de les característiques de les dades que es demanin, es requerirà un tipus de control o un altre.

El valor d'aquest atribut pot ser un dels següents:

1) `text`. Caixa d'introducció de text, generalment no gaire llarga.

2) `password`. Caixa d'introducció de text, per a paraules de pas. El text introduït s'ofusca usant asteriscos o símbols que no revelen la informació.

3) `checkbox`. Opció seleccionable junt amb altres opcions del mateix tipus. Es poden seleccionar diverses opcions de tipus "checkbox" al mateix formulari.

4) `radio`. Opció seleccionable, mútuament excloent amb altres opcions de tipus radi dintre del mateix control.

5) `submit`. Botó que serveix per a enviar totes les dades introduïdes al formulari, segons el mètode d'enviament definit a l'etiqueta <form> (GET o POST) al fitxer del servidor que també s'ha definit a l'etiqueta <form>. Tot formulari haurà de tenir un control de tipus "submit" o equivalent, que permeti enviar les dades al programa destí.

6) `reset`. Botó que serveix per a netejar totes les dades introduïdes per l'usuari al formulari.

7) `file`. Quadre de text amb un botó associat, que permet triar un fitxer de l'ordinador client per tal d'adjuntar-ho al formulari i enviar-lo cap al servidor. Si es fa servir aquest component en un formulari, és necessari afegir l'atribut "`enctype`" a l'etiqueta <form> amb el valor "`multipart/form-data`".

8) `hidden`. Aquests controls no es mostren a l'usuari. Serveixen per a enviar informació que l'usuari del formulari no cal que conegui. Sol ser informació de control que el servidor necessita i que l'usuari no ha d'introduir.

9) `image`. Serveix per a personalitzar més el formulari, amb imatges carregades des del client.

10) `button`. Amb els botons de "`submit`" i "`reset`", la funcionalitat dels formularis és molt limitada. Es poden definir altres tipus de botons amb aquesta etiqueta. En fer clic a un botó definit d'aquesta manera, el formulari no fa res. És necessari una programació en un llenguatge web de client, com JavaScript, per exemple, en què es programarà la funcionalitat d'aquests botons.

Els altres atributs de l'etiqueta <input> són:

1) name = *[text]*. És el nom del control pel qual es podrà accedir al seu contingut a través de la programació al servidor. El podeu veure en el codi anterior en color vermell.

2) value = *[text]*. És el valor assignat al control. El valor introduït per l'usuari en el formulari s'emmagatzema en aquest camp. El trobareu en color verd en el codi anterior.

3) size = *[mida]*. És la mida inicial del control. Es mesura en caràcters per als controls de tipus text (**"text"** i **"password"**) i en píxels per a la resta de controls.

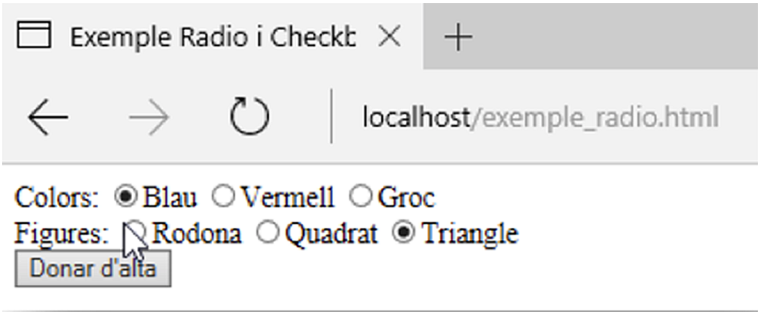
4) maxlength = *[nombre màxim de caràcters]*. És la mida màxima en caràcters dels controls de tipus text (**"`text`"** i **"`password`"**). Tingueu en compte que una cosa és la llargada del control i una altra el nombre de caràcters que s'hi poden entrar.

5) `checked = "checked"`. Per a un control de tipus **"`checkbox`"** o **"`radio`"**, indica que el control està seleccionat. En el cas dels controls de tipus **"`radio`"**, només pot haver-hi una opció amb aquest atribut, dintre del mateix control, tot i que pot haver-hi diversos grups de botons de tipus "radio" al mateix formulari. Per exemple:

```
<form action="fitxer.php" method="post">
       Colors:
 <input type="radio" checked="checked" name="colors" value="Blau" />Blau
 <input type="radio" name="colors" value="Vermell" />Vermell
 <input type="radio" name="colors" value="Groc" />Groc
 <br/>
       Figures:
 <input type="radio" checked="checked" name="figures" value="Rodona" />Rodona
 <input type="radio" name="figures" value="Salat" />Quadrat
 <input type="radio" name="figures" value="Picant" />Triangle
  <br/>

 <input type="submit" value="Donar d'alta"/>
 </form>
```

Aquest codi correspon al següent formulari, en què es poden veure dos grups de botons de radi independents, amb diferents seleccions:



6) disabled ="**disabled**". Si un control té aquest atribut, el control està deshabilitat i no es pot modificar.

7) `readonly = "readonly"`. Si un control té aquest atribut, el control és de només lectura, o sigui, l'usuari no podrà editar ni modificar-ne el contingut.

8) src = *[url]*. Per a les imatges, indica la URL de la imatge que s'ha de carregar com a botó al formulari.

9) alt = *[text]*. És una descripció alternativa del control per qüestions d'accessibilitat.

Un cop vist això, podem entendre el codi del formulari d'exemple amb mètode GET del subapartat 2.1.:

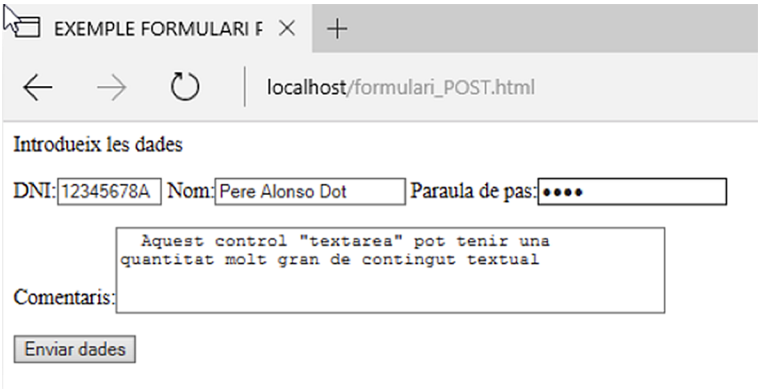
```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-
  8"/>
    <title>EXEMPLE FORMULARI GET</title>
  </head>
  <body>
    <p>Introdueix les dades</p>
    <form action="http://localhost/fitxer_get.php" method="get">
                DNI:<input size="9" maxlength="9" type="text"
name="DNI" />
                Nom:<input type="text" name="nom" />
                Paraula de pas:<input type="password" name="Password"
/>
                <input type="submit" value="Enviar dades"/>
    </form>
  </body>
</html>
```

2.3. Entrada de text extensos: etiqueta `<textarea>`

Amb el control "`text`" que s'ha vist a la secció anterior no n'hi ha prou si es vol introduir una gran quantitat de text. Hi ha un altre control que permet introduir gran quantitat de text, definit mitjançant l'etiqueta `<textarea>`.

La mida d'aquest control es pot definir amb columnes (atribut `cols`) i files (atribut `rows`).

Un exemple d'aquest control és el següent:



Els atributs comuns dels quals disposa aquesta etiqueta són **`name`**, **`disabled`** i **`readonly`**.

Com a atributs específics, l'etiqueta disposa d'aquests:

- **`cols =`** *`[num_columnes]`*. Indica l'amplada inicial en caràcters del component.
- **`rows =`** *`[num_files]`*. Indica l'altura inicial en línies de caràcters del component.

Aquesta etiqueta no té l'atribut **`value`**. Si es vol incloure un valor per defecte al control, s'haurà d'indicar entre la seva etiqueta d'obertura i la de tancament.

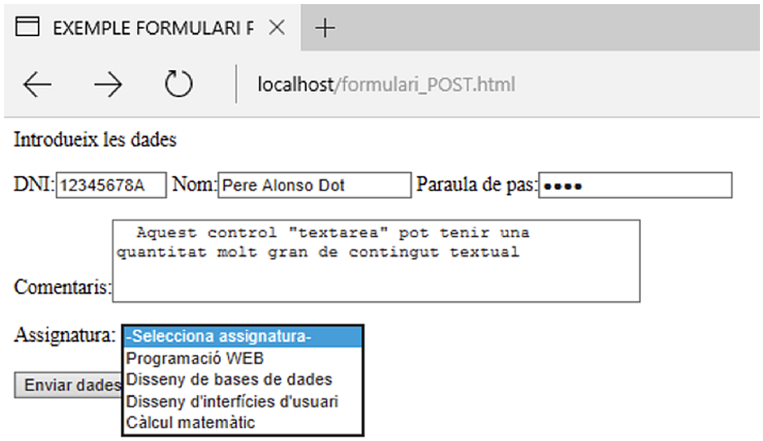
Un exemple d'utilització d'aquesta etiqueta amb el mètode POST d'enviament de dades és:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-
8"/>
    <title>EXEMPLE FORMULARI POST</title>
  </head>
  <body>
   <p>Introdueix les dades</p>
   <form action="http://localhost/fitxer_post.php" method="post">
            DNI:<input size="9" maxlength="9" type="text" name="DNI" />
            Nom:<input type="text" name="nom" />
            Paraula de pas:<input type="password" name="Password" />
            <p>
            Comentaris:<textarea rows="4" cols="50" name="Comentaris">
                  Aquest control "textarea" pot tenir una quantitat
molt gran de contingut textual
            </textarea>
            </p>
            <input type="submit" value="Enviar dades"/>
   </form>
  </body>
</html>
```

2.4. Llistes desplegables d'opcions: etiqueta `<select>`

Una llista desplegable és un control que permet triar una opció d'entre unes quantes, i mostra la llista d'opcions disponibles en fer clic sobre el control. Aquestes llistes es creen mitjançant l'etiqueta `<select>`.

Un exemple d'utilització d'aquest control és:



Com es pot veure, la llista desplegable mostra diverses opcions, i n'hi ha una que està seleccionada per defecte.

Els atributs específics de què disposa aquesta etiqueta són:

1) size = *[num_opcions]*. Indica el nombre d'opcions que es mostraran a la llista. Per defecte és 1.

2) multiple = *"multiple"*. Si hi ha aquest atribut, indica que es pot seleccionar més d'una opció de la llista.

Com a atribut comú, la llista té **`name`**, però no **`value`**. Per tal d'afegir opcions a la llista desplegable, cal fer servir una etiqueta <option> per a cada opció que es vulgui afegir a la llista. Cada opció té els següents atributs:

a) value = *[valor]*. Indica el text que s'envia al servidor si aquesta opció és una de les seleccionades. Si no s'indica el contrari, el valor d'una opció pot ser diferent del text mostrat per l'opció.

b) selected = *"selected"*. Si hi ha aquest atribut, indica que l'opció es mostrarà seleccionada per defecte.

c) Disabled = *"disabled"*. Si hi ha aquest atribut, indica que l'opció es mostrarà deshabilitada, o sigui, que es mostrarà, però no es podrà seleccionar.

El text que es mostrarà per cada opció és el que es posa entre el tag *<option>* i el tag *</option>*. Tal com s'ha avançat, el text que es mostra a la llista i el que s'envia al servidor, si se selecciona l'opció, no ha de ser el mateix. Un es defineix a l'atribut **value**, i l'altre, entre <option> i </option>.

Un exemple d'utilització d'aquesta etiqueta és:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EXEMPLE FORMULARI POST</title>
</head>
<body>
<p>Introdueix les dades</p>
<form action="http://localhost/fitxer_post.php" method="post">
 DNI:<input size="9" maxlength="9" type="text" name="DNI" />
 Nom:<input type="text" name="nom" />
 Paraula de pas:<input type="password" name="Password" />
 <p>
 Comentaris:<textarea rows="4" cols="50" name="Comentaris">
  Aquest control "textarea" pot tenir una quantitat molt gran de
contingut textual
 </textarea>
 </p>
 Assignatura:
 <select name="Assignatura">
     <option value="" selected="selected">
  -Selecciona assignatura-
  </option>
     <option value="Programació WEB">
  Programació WEB
  </option>
     <option value="Disseny de bases de dades">
  Disseny de bases de dades
  </option>
     <option value="Disseny d'interfícies d'usuari">
  Disseny d'interfícies d'usuari
  </option>
     <option value="Càlcul matemàtic" disabled="disabled">
  Càlcul matemàtic
  </option>
</select>
<br>
<br>
 <input type="submit" value="Enviar dades"/>
</form>
</body>
</html>
```

Opció seleccionada per defecte

Valor que s'envia al servidor si se selecciona l'opció

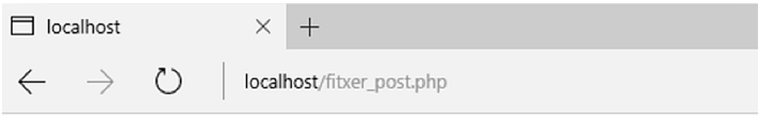
Valor que es mostra per pantalla

Opció deshabilitada, no es pot seleccionar

Hi ha diversos modes de funcionament d'aquesta etiqueta:

1) Llista que mostra una opció per defecte i en fer clic es desplega la llista per a triar una opció: és el mode de funcionament mostrat a l'exemple anterior.

2) Llista que mostra totes les opcions inicialment, amb una opció per defecte i que només permet triar una opció:

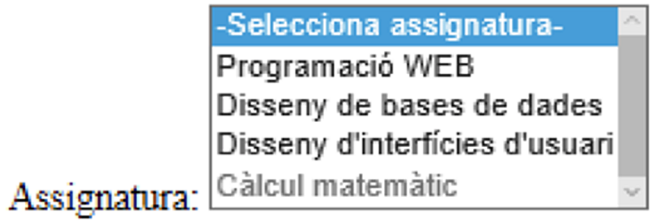


DNI: 12345678A
Nom: Pere Alonso Dot
Password: Hola
Comentari: Aquest control "textarea" pot tenir una quantitat molt gran de contingut textual
Assignatura: Disseny de bases de dades

Només cal afegir-hi l'atribut **`size =`** *`[num_opcions]`*, posant com a valor a *`num_opcions`* el nombre d'opcions totals de la llista:

```
Assignatura:
 <select name="Assignatura" size = 5>
...
 </select>
```

I la llista es mostraria de la següent manera:

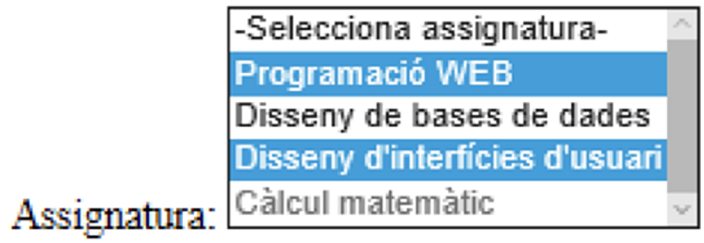


3) Llista que mostra una o totes les opcions, però que permet triar més d'una de les opcions disponibles.

En aquest cas, cal afegir-hi l'atribut **`multiple =`** *`"multiple"`*:

```
Assignatura:
 <select name="Assignatura" size = 5 multiple = "multiple">
...
 </select>
```

I la llista es mostraria de la següent manera:



El mètode per a seleccionar diverses opcions d'una llista és l'habitual; amb les tecles CTRL per a seleccionar opcions alternes i amb SHIFT per a opcions consecutives.

Una llista desplegable d'opcions pot arribar a ser bastant extensa. A més, és possible que a la mateixa llista s'hi agrupin opcions de diferents classes. Per tal de tenir les opcions d'una llista categoritzades (en cas que la llista tingui un gran nombre d'opcions) es fa servir una etiqueta anomenada `<optgroup>`. Aquesta etiqueta és només un separador entre les diferents opcions i no es pot seleccionar com una opció de la llista.

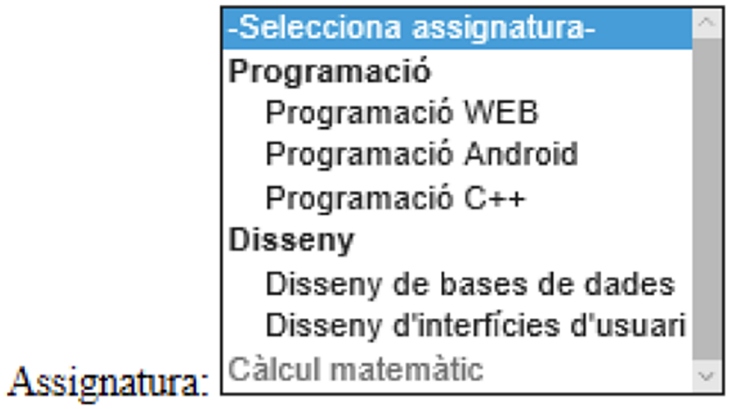
L'etiqueta *<optgroup>* només té un atribut, el nom del grup d'opcions que representa. Aquest atribut és:

label = *[nom_del_grup]*. Mostra el nom del grup a la llista desplegable.

Per exemple:

```
Assignatura:
 <select name="Assignatura" size = 9 multiple = "multiple">
      <option value="" selected="selected">
  -Selecciona assignatura-
  </option>
      <optgroup label = "Programació">
      <option value="Programació WEB">
  Programació WEB
  </option>
      <option value="Programació Android">
  Programació Android
  </option>
      <option value="Programació C++">
  Programació C++
  </option>
      </optgroup>
      <optgroup label = "Disseny">
      <option value="Disseny de bases de dades">
  Disseny de bases de dades
  </option>
      <option value="Disseny d'interfícies d'usuari">
  Disseny d'interfícies d'usuari
  </option>
      </optgroup>
      <option value="Càlcul matemàtic" disabled="disabled">
  Càlcul matemàtic
  </option>
</select>
```

Mostraria la següent llista:



2.5. Estructuració de l'entrada de dades

Per tal de poder estructurar la interfície d'usuari que s'està dissenyant per a una pàgina web, existeixen una sèrie d'etiquetes que donen format i permeten agrupar informacions del mateix tipus en zones concretes de la pàgina web, amb la qual cosa es guanya en claredat i usabilitat.

2.5.1. Etiqueta `<label>`

Aquesta etiqueta es fa servir per a posar nom a diferents parts d'un formulari. Per ara, cada vegada que s'ha fet servir una etiqueta d'entrada de dades (<input>, <textarea> o <select>) s'ha escrit un text abans perquè l'usuari conegui quina mena d'informació se li està demanant en cada cas. Amb l'etiqueta <label>, aquesta tasca es fa de forma més estructurada.

L'únic atribut específic que té aquesta etiqueta és:

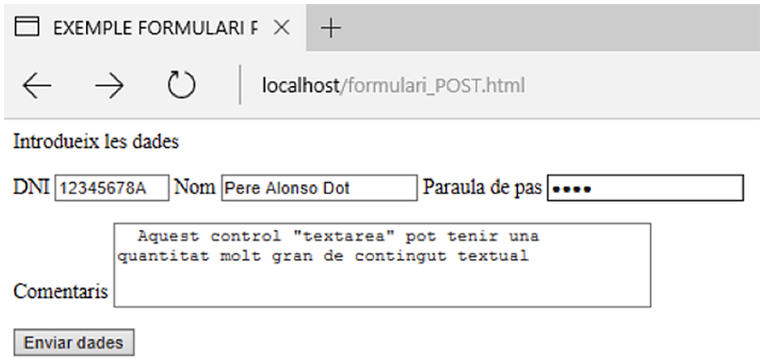
`for =` *`[identificador_etiqueta]`*. Qualsevol altre control al formulari que vulgui fer servir el text d'aquesta etiqueta haurà de fer referència a aquest identificador.

Entre els tags *`<label>`* i *`</label>`* es posa el text que es mostrarà cada vegada que algun altre control vulgui mostrar el text d'aquesta etiqueta.

Un exemple d'utilització d'aquesta etiqueta seria:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EXEMPLE FORMULARI GET</title>
</head>
<body>
<p>Introdueix les dades</p>
<form action="http://localhost/fitxer_post.php" method="post">
<label for="DNI">DNI</label>
<input id="DNI" size="9" maxlength="9" type="text" name="DNI"/>
<label for="nom">Nom</label>
<input id="nom" type="text" name="nom" />
<label for="pwd">Paraula de pas</label>
<input id="pwd" type="password" name="Password" />
<input type="submit" value="Enviar dades"/>
</form>
</body>
</html>
```

I es mostraria així al navegador:



2.5.2. Etiquetes `<fieldset>` i `<legend>`

Aquest parell d'etiquetes serveixen per a agrupar diferents components amb informació relacionada, dintre de la mateixa àrea del formulari, i posar-li un nom a aquella àrea.

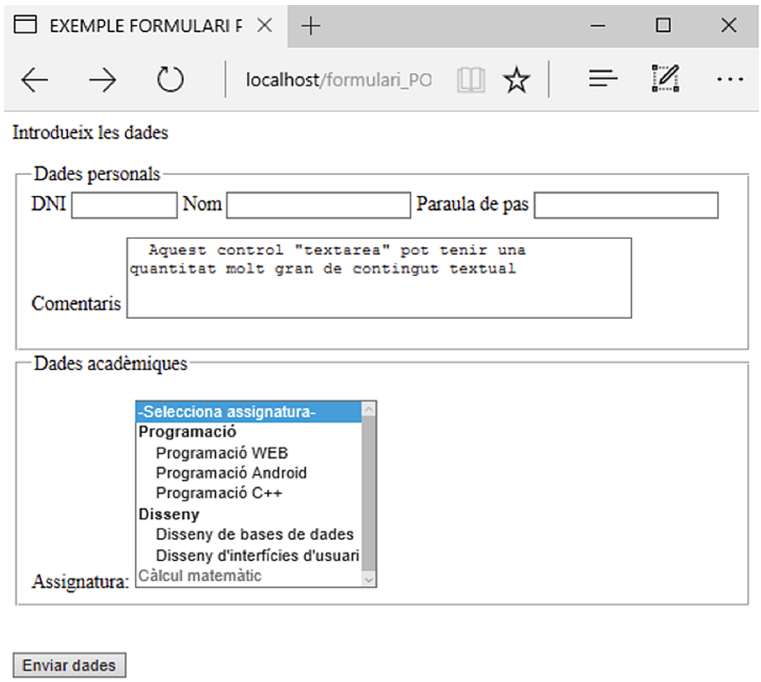
`<fieldset>` no té atributs. Simplement agrupa totes les altres etiquetes que estiguin entre `<fieldset>` i `</fieldset>` dintre de la mateixa àrea de pantalla.

`<legend>` tampoc té atributs. Aquesta etiqueta es posa dintre d'un `<fieldset>` i el text que s'escrigui entre `<legend>` i `</legend>` serà el nom de l'àrea definida pel `<fieldset>`.

Per exemple, el següent codi:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EXEMPLE FORMULARI POST</title>
</head>
<body>
<p>Introdueix les dades</p>
<form action="http://localhost/fitxer_post.php" method="post">
<fieldset>
<legend>Dades personals</legend>
 <label for="DNI">DNI</label>
 <input id="DNI" size="9" maxlength="9" type="text" name="DNI" />
 <label for="nom">Nom</label>
 <input id="nom" type="text" name="nom" />
 <label for="pwd">Paraula de pas</label>
 <input id="pwd" type="password" name="Password" />
 <p>
 <label for="comentaris">Comentaris</label>
 <textarea id="comentaris" rows="4" cols="50" name="Comentaris">
  Aquest control "textarea" pot tenir una quantitat molt gran de
contingut textual
 </textarea>
</fieldset>
<fieldset>
<legend>Dades acadèmiques</legend>
 </p>
 Assignatura:
 <select name="Assignatura" size = 9 multiple = "multiple">
      <option value="" selected="selected">-Selecciona assignatura-
</option>
      <optgroup label = "Programació">
      <option value="Programació WEB">Programació WEB</option>
      <option value="Programació Android">Programació Android</option>
      <option value="Programació C++">Programació C++</option>
      </optgroup>
      <optgroup label = "Disseny">
      <option value="Disseny de bases de dades">Disseny de bases de
dades</option>
      <option value="Disseny d'interfícies d'usuari">Disseny
d'interfícies d'usuari</option>
      </optgroup>
      <option value="Càlcul matemàtic" disabled="disabled">Càlcul
matemàtic</option>
</select>
</fieldset>
<br>
<br>
 <input type="submit" value="Enviar dades"/>
</form>
</body>
</html>
```

mostra el següent resultat:



3. Obtenció de dades d'un formulari amb PHP

Un formulari HTML recopila una sèrie de dades, tot definint una interfície d'usuari amigable, de qualitat i usable. Aquestes dades han de ser tractades al servidor, sovint accedint a la base de dades per a realitzar alguna gestió com pot ser fer-hi una inserció, una modificació, una eliminació o una consulta de dades.

A continuació veurem com es pot accedir mitjançant PHP a les dades d'un formulari.

3.1. Variables predefinides per PHP: $_GET i $_POST

Tal com s'ha comentat, es poden fer servir dos mètodes per a enviar les dades del formulari des del client al servidor: GET i POST.

PHP proveeix de dues variables ($_GET i $POST) per a accedir a la informació enviada per un formulari en HTML. La utilització de l'una o l'altra dependrà del mètode que s'hagi fet servir al formulari per a enviar la informació (atribut `method` de l'etiqueta `<form>`).

Si es té un fitxer PHP, la forma d'accedir a la informació és la següent:

```
$_POST['nom_atribut']
```

```
$_GET['nom_atribut']
```

El '`nom_atribut`' és el valor de l'atribut "`name`", de l'etiqueta `<input>` del fitxer HTML de la qual es vol consultar el valor. Abans de consultar el valor d'un control del formulari és necessari comprovar que s'ha rebut algun valor d'aquest control. Per a fer-ho s'utilitzarà la funció isset(<variable>), que rep la variable com a paràmetre i retorna cert en cas que tingui algun valor.

Així, en el següent exemple, es mostra el fitxer HTML i el fitxer PHP i el traspàs d'informació entre ambdós:

Fitxer HTML que conté el formulari

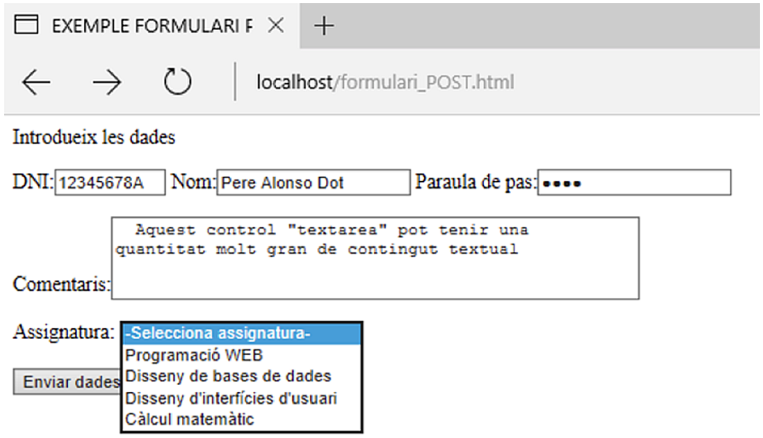
```
...
<form action="http://localhost/fitxer_get.php" method="get">
 DNI:<input size="9" maxlength="9" type="text" name="DNI" />
...
```

Fitxer PHP que gestiona les dades enviades pel formulari

```
<?php
header("Content-Type: text/html;charset=utf-8");
//codi PHP que rep les dades via GET del formulari
$DNI = '';
if (isset($_GET['DNI'])) $DNI = $_GET['DNI'];
echo ('DNI: ' . $DNI);
```

Es pot observar que en el fitxer PHP, abans d'accedir al contingut del paràmetre DNI, es comprova si aquest paràmetre realment té algun valor, amb la instrucció `isset(variable)`. Si efectivament té un valor, la variable $DNI prendrà per valor el valor enviat pel formulari HTML. Si no, la variable $DNI es quedarà amb el valor de la seva inicialització, o sigui, '' (cadena buida).

En cas que l'enviament de la informació des del fitxer HTML cap al fitxer PHP del servidor sigui segons el mètode POST, només caldrà fer servir $_POST en lloc de $_GET.



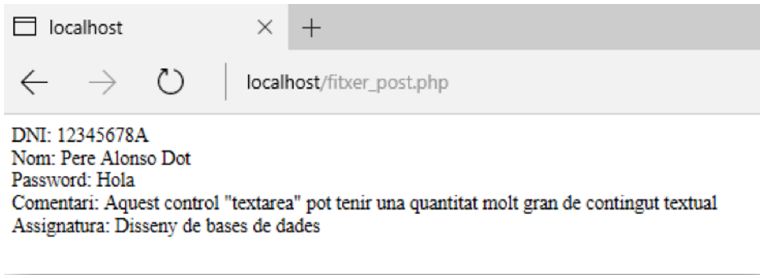
Així, el formulari de la imatge té aquest codi en HTML:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>EXEMPLE FORMULARI POST</title>
</head>
<body>
<p>Introdueix les dades</p>
<form action="http://localhost/fitxer_post.php" method="post">
 DNI:<input size="9" maxlength="9" type="text" name="DNI" />
 Nom:<input type="text" name="nom" />
 Paraula de pas:<input type="password" name="Password" />
 <p>
 Comentaris:<textarea rows="4" cols="50" name="Comentaris">
  Aquest control "textarea" pot tenir una quantitat molt gran de
contingut textual
 </textarea>
 </p>
 Assignatura:
 <select name="Assignatura">
      <option value="" selected="selected">
  -Selecciona assignatura-
  </option>
      <option value="Programació WEB">
  Programació WEB
  </option>
      <option value="Disseny de bases de dades">
  Disseny de bases de dades
  </option>
      <option value="Disseny d'interfícies d'usuari">
  Disseny d'interfícies d'usuari
  </option>
      <option value="Càlcul matemàtic" disabled="disabled">
  Càlcul matemàtic
  </option>
</select>
<br>
<br>
 <input type="submit" value="Enviar dades"/>
</form>
</body>
</html>
```

I el fitxer PHP que gestiona les dades proporcionades per l'anterior formulari seria aquest:

```
<?php
header("Content-Type: text/html;charset=utf-8");
//codi PHP que rep les dades via POST del formulari
$DNI = '';
$nom = '';
$pass = '';
$comentaris = '';
$assignatura = '';
if (isset($_POST['DNI'])) $DNI = $_POST['DNI'];
if (isset($_POST['nom'])) $nom = $_POST['nom'];
if (isset($_POST['Password'])) $pass = $_POST['Password'];
if (isset($_POST['Comentaris'])) $comentaris = $_POST['Comentaris'];
if (isset($_POST['Assignatura'])) $assignatura =
$_POST['Assignatura'];
echo ('DNI: ' . $DNI); echo ('<br>');
echo ('Nom: ' . $nom); echo ('<br>');
echo ('Password: ' . $pass); echo ('<br>');
echo ('Comentari: ' . $comentaris); echo ('<br>');
echo ('Assignatura: ' . $assignatura);
?>
```

El resultat de l'execució del fitxer PHP és el següent:


localhost
localhost/fitxer_post.php
DNI: 12345678A
Nom: Pere Alonso Dot
Password: Hola
Comentari: Aquest control "textarea" pot tenir una quantitat molt gran de contingut textual
Assignatura: Disseny de bases de dades

4. Seguretat en l'entrada de dades: protecció contra atacs SQL-Injection

Els atacs informàtics a través de pàgines web són un problema molt comú. Una part d'aquests atacs es basen en l'entrada de dades a través d'un formulari. A continuació es discuteixen els problemes de seguretat que poden comportar una manca de control de les dades introduïdes. En el següent punt, es veuran possibles solucions per a implementar controls que evitin els problemes presentats.

4.1. El problema: l'entrada de dades

Si un sistema no està ben protegit, amb atacs basats en l'entrada de dades, un atacant pot, entre d'altres:

- Conèixer, modificar i/o eliminar dades personals d'usuaris del sistema.
- Obtenir drets d'administració del sistema.
- Obtenir el control de tot el servidor en què estigui allotjada la pàgina web atacada.
- Veure el codi intern de qualsevol base de dades del sistema.
- Executar comandes en el sistema operatiu del servidor atacat.

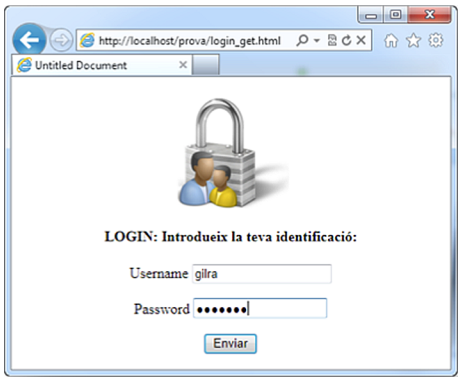
Així, és responsabilitat del dissenyador i del programador web implementar un sistema de seguretat per a evitar, en la mesura del possible, aquests atacs. I no només perquè no se'n vegi afectada la informació pròpia, sinó també perquè un atacant no pugui aconseguir prendre el control de *tot* el servidor atacat a través del seu sistema web.

Una de les lleis bàsiques de la seguretat informàtica (i també de la seguretat en general) és que **la seguretat màxima d'un sistema és igual a la seguretat del seu punt més feble**. Així, si en un servidor hi ha allotjats diversos sistemes web, però només un d'aquests sistemes no té una seguretat adequada, *tots* els sistemes poden veure compromesa la seva seguretat.

Un dels atacs més comuns d'entrada de dades és l'atac d'injecció de codi (SQL-Injection, en anglès).

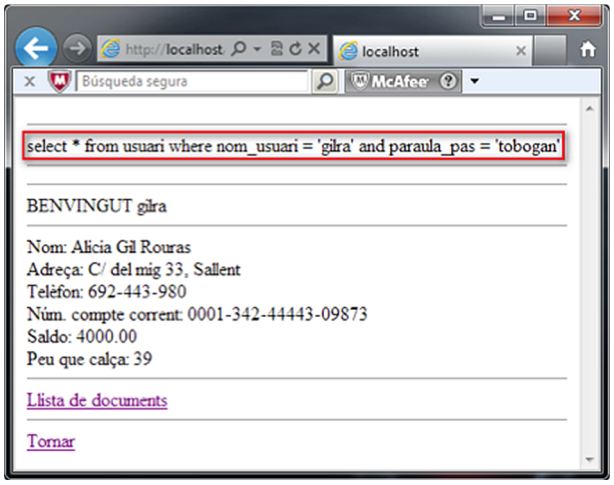
El concepte principal d'un atac de SQL-Injection és inserir una sentència SQL mitjançant la "injecció" de codi SQL als camps d'entrada d'un formulari, o a la barra d'adreces del navegador, que després avaluarà el codi PHP de l'aplicació web.

Per exemple, un formulari típic de validació d'usuari a un sistema web, amb el nom d'usuari i la contrasenya:



Al botó "Enviar" (el "submit" del formulari) hi ha una crida a un fitxer PHP, en què es validarà si el nom d'usuari i la contrasenya són correctes, per saber si dóna un missatge d'error, o bé, per donar accés a l'usuari. En aquest cas, el formulari enviarà la informació al fitxer PHP mitjançant el mètode GET.

En fer clic al botó "Enviar", es carrega la següent pàgina:



Per tal que es vegi més clarament com actua la injecció de codi SQL, es mostra a dalt de la pàgina quina és la instrucció SQL que ha generat el sistema.

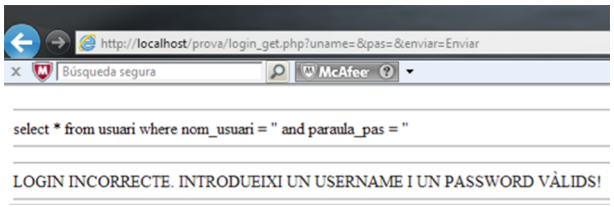
En aquesta pàgina hi ha tota la informació de l'usuari Alicia Gil Rouras. A la barra de navegació del navegador, es pot veure que l'adreça generada per l'anterior formulari ha estat aquesta:

La primera errada *greu* és fer servir el mètode GET per a enviar dades secretes, com pot ser la contrasenya de l'usuari. En aquest cas, només mirant la barra de navegació, es pot veure que la contrasenya d'aquest usuari és "tobogan". És clar que un atacant no sap, d'entrada, aquesta informació, però aquest sistema

dóna peu a dos atacs sobre la pàgina: l'atac de "**força bruta**", o sigui, tenint en compte un usuari, anar provant contrasenyes contingudes en un diccionari (hi ha programes que automatitzen aquesta tasca) i l'atac de "**mirar de dalt a baix**", o sigui, que en el moment d'introduir la informació per part de l'usuari (sobretot en espais públics) hi hagi algú mirant la pantalla i ho pugui veure.

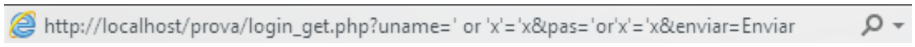
De totes maneres, aquests dos atacs no tenen res a veure amb SQL-injection, però dóna una mostra de com pot arribar a ser de vulnerable una pàgina web mal dissenyada i programada.

El que sí que pot fer un atacant amb aquesta pàgina, sense conèixer ni el nom d'usuari ni la paraula de pas, és, primer de tot, fer clic en el botó "Enviar" amb un nom d'usuari i una contrasenya en blanc. Es mostrarà la següent pàgina:

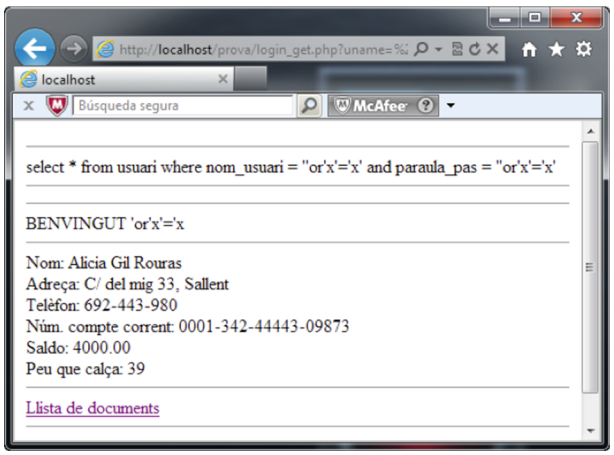


Com es pot veure, la pàgina dóna un error de validació. Però això és tot el que necessita saber l'atacant per a poder entrar al sistema: a la **barra de navegació** es pot veure quins paràmetres passa (fent servir el perillós mètode GET) la pàgina de validació d'usuari a la pàgina PHP; aquests paràmetres són "uname" (que té pinta de ser el nom d'usuari) i "pas" (que molt probablement deu ser la contrasenya).

Tot el que ha de fer l'atacant és canviar l'adreça que es mostra a la barra d'adreces del navegador, de la següent manera:



L'atacant ha posat com a username: `'or'x'='x` i la mateixa contrasenya, amb el qual el codi PHP, en obtenir el valor d'aquests dos paràmetres amb $_GET, genera la següent pàgina:



Aquesta és la informació *personal* d'Alicia Gil Rouras!

Com és possible? L'atacant ha canviat el nom de l'usuari i la contrasenya per `'or'x'='x`. Quina consulta SQL acaba executant el fitxer PHP?

```
Select * from usuari where nom_usuari = ''or'x'='x' and
paraula_pas = ''or'x'='x'
```

Això significa que MySQL rep una consulta que li està dient: "Dóna'm tota la informació dels usuaris el nom d'usuari dels quals sigui `''or'x'='x'` i que tinguin com a contrasenya `''or'x'='x'`".

Aquesta consulta, una vegada avaluada la condició (la part "where" de la consulta), queda de la següent manera ('x' = 'x' sempre és CERT):

```
Select * from usuari where nom_usuari = '' or CERT and paraula_pas =
''or CERT
```

`'' or CERT = CERT`

```
Select * from usuari where nom_usuari = CERT and paraula_pas = CERT
```

`CERT and CERT = CERT`

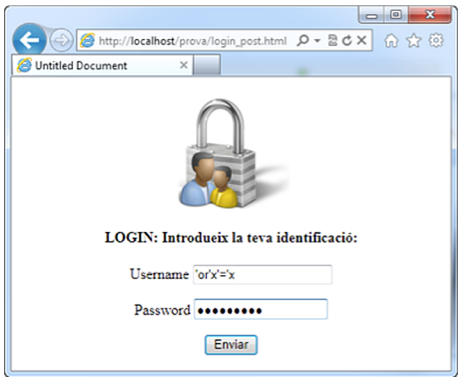
```
Select * from usuari where CERT
```

Com que la condició sempre es compleix, es fa la part del select..from, i dóna el primer usuari de la llista d'usuaris!

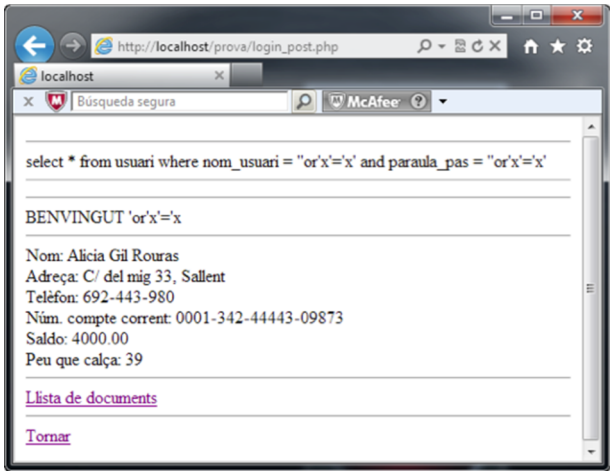
Cal dir que amb altres cadenes d'injecció de SQL es pot arribar a treure la informació de *tots* els usuaris de la base de dades, fins i tot entrar al sistema com a administrador!

Aquesta és l'essència d'un atac de SQL-Injection: canviar la instrucció SQL que s'executa en el servidor per tal que retorni informació reservada o bé que executi instruccions de modificació o esborrament a la base de dades.

Fer servir el mètode POST no soluciona res, ja que la cadena d'injecció de SQL es pot posar als `<input>` del formulari, de la següent manera:



I com es pot veure, el resultat és exactament el mateix que amb el mètode GET:



La conclusió és que si no es prenen altres tipus de mesures, tant li fa que es faci servir GET com POST. L'atacant podrà extreure informació de la base de dades sense gaire dificultat.

Cal dir que aquest no és ni de bon tros l'únic atac de SQL-injection que es pot fer a un sistema web. De fet, és el més senzill. Dintre de la tècnica de SQL-injection hi ha moltes subtècniques diferents per a realitzar atacs a sistemes web.

4.2. La solució: controlar l'entrada de dades

No hi ha cap solució definitiva per a evitar *tots* els possibles atacs de SQL-injection a un sistema web. El que sí que es pot fer és posar-hi mesures per a evitar la gran majoria d'aquests atacs.

La primera mesura a prendre és ***no fer servir el mètode GET***, o si es fa servir, almenys, enviar la informació xifrada, tant en el cas dels valors com en el cas dels noms dels camps. No és que el mètode POST eviti aquests tipus d'atacs, però, com a mínim, no és tan evident com el mètode GET.

La segona mesura és ***validar* sempre *l'entrada de dades***. No es pot permetre que l'usuari de la pàgina web (que en un moment donat pot ser un atacant) pugui posar qualsevol dada als `<input>` d'un formulari.

Això implica diverses coses:

1) Controlar la longitud de les entrades. Fixar una longitud màxima tant per als camps d'entrada numèrics com textuals, per a evitar errors de desbordament.

2) Gestionar els errors des del programa. No deixar que sigui el sistema qui doni els missatges d'error, perquè el sistema pot donar informació sensible en el text dels seus errors, com per exemple el nom de la base de dades, de l'usuari d'accés o de taules o camps de taules.

3) Validar el contingut de $_GET o $_POST. No concatenar *mai* les variables $_GET o $_POST en una consulta SQL sense haver-ne validat primer el contingut. A MySQL existeix una funció que "neteja" (s'anomena "escapar" en argot tècnic) de caràcters perillosos una cadena de text que s'hi passi. Aquesta instrucció és:

```
mysqli_real_escape_string (connexió, cadena);
```

S'hi passa la connexió actual amb la base de dades, i la cadena de text que es vol comprovar, i retorna la mateixa cadena, lliure de caràcters perillosos per un atac de SQL-Injection.

Un exemple d'utilització d'aquesta instrucció seria:

```
//validació login
$instruccio = "select * from usuari where nom_usuari =
mysqli_real_escape_string ($con, $_POST['uname']) and paraula_pas =
mysqli_real_escape_string ($con, $_POST['pas']);

$res = mysqli_query($con, $instruccio);
if (!$res) //si error es finalitza
{
      die("Validació incorrecta");
}
else
{
      echo "Validació correcta";
}
```

4) Donar els permisos adients als usuaris de la base de dades per tal que només puguin accedir a les àrees del sistema gestor de bases de dades que realment necessitin per a realitzar la seva feina.

En qualsevol cas, tot i que no es pot evitar al 100% aquest tipus d'atac, sí que, seguint aquestes directrius, es poden posar les coses més difícils a possibles atacants.