Bases de Dades

Practica PHP

Francesc Xavier Bullich Parra

U1939691

Gurp 2 – Dijous 8-10

Índex

Consideracions inicials

M'he trobat amb algun problema alhora d'utilitzar alguns dels controls html, com segurament s'haurà trobat la majoria de companys ja que cada navegador va una mica a la seva. Per això he decidit centrar-me en que funciones be almenys en un navegador.

Així dons he triat el Mozilla Firefox i he fet que funciones correctament amb aquest. D'altra banda al arreglar coses en aquest navegador el més probable es que no funcionin del tot igual en un altre, però he donat prioritat en que funciones la part que implica la base de dades en comptes dels navegadors. Soc conscient però que hauria de funcionar be en tots els casos.

Control d'accés

Per l'accés d'un usuari al lloc web s'ha utilitzat el mètode que es va donar a classe: l'ús de sessions emmagatzemades. Cada cop que s'inicia la navegació (mentre el navegador ho recordi) es demanaran les credencials.

S'ha aplicat el control a tot el lloc web de manera que si s'intenta accedir a qualsevol pàgina que no sigui la inicial es redirigirà automàticament a la pantalla de login.

```
<?php
        if (empty($_SESSION['user']) or empty($_SESSION['pwd'])){
                header('Location: practica_PHP.html');
                exit;// no seguir carregan la pagina
        }
?>
```

Si no hi ha guardats ni l'usuari ni la contrasenya s'encertarà una línia al header que farà que es redirigeixi a la pantalla inicial (login). A més s'hi posa l'exit per que no segueixi carregant la pàgina, de tal forma que no es veurà res de la pagina a la que es volia accedir (quedaria estrany veure com es comença a carregar una pàgina i de cop canvia).

També s'ha aplicat un control per validar les dades de connexió. De fet l'únic que fa és intentar crear una connexió amb l'oracle. De manera que si no pot accedir torni a demanar les dades d'accés.

```
<?php
        $conn = oci_connect($_SESSION['user'], $_SESSION['pwd'],
                                'oracleps');
        if(!$conn){
                header('Location: practica_PHP.html');
                exit;// no seguir carregan la pagina
?
```

El funcionament es el mateix que abans, si no es pot accedir posa una línia a la capçalera que fa que redirigeixi automàticament al login i deixa de carregar la pàgina.

Aquest control es a totes les pagines de lloc web, per tant només es podrà accedir passant pel login.

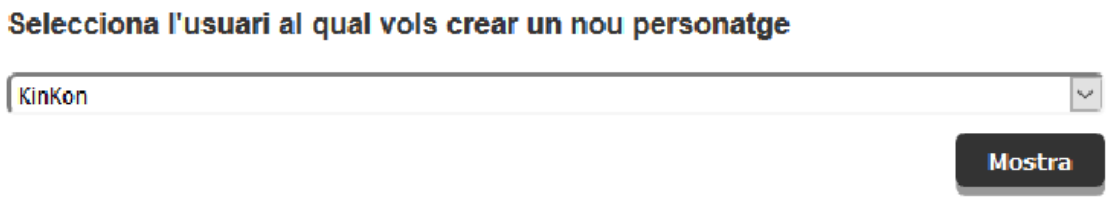
Estructura de les pàgines

S'utilitza una estructura de menú simple. Cada una de les opcions de l'enunciat està representada per una opció en una barra de menú horitzontal.

A la pàgina inicial hi ha un resum del que fa cada apartat (tot i que son totes les opcions seguides el nom es diferent).

He decidit utilitzar només una pàgina per a cada opció. Totes son php de forma que pugui carregar tot el que necessito en cada un dels passos de cada opció.

El primer que es troba a cada una de les pàgines és un selector que permet triar una cursa, un usuari, etc. Per exemple la opció per crear un nou personatge primer demanarà quin es l'usuari al que pertany.



Aquest mètode permet:

- Assegurar-se que s'entri un valor vàlid. Donat que es un selector no dona opció a entrar valors diferents als existents.
- Restringir els valors possibles. En algun cas hi ha combinacions que serien errors i s'haurien de tractar posteriorment. Per exemple inscriure un personatge i vehicle. Si primer seleccionem usuari i cursa, restringirem els valors possibles de personatges i de vehicles als del usuari triat, de manera que es mes senzill no equivocar-se i produir errors involuntaris.

Nota: Tot i això en altres casos no seria necessari demanar-ho al principi d'aquesta manera queden totes les pàgines estructurades de forma similar.

Formularis

Com he comentat abans, les pàgines son úniques per opció. De manera que els formularis redirigeixen a la mateixa pagina, passant els paràmetres per POST.

Es fa un control inicial amb PHP per saber quin apartat toca mostrar segons els paràmetres POST que s'han rebut. Per exemple:

La pàgina de crear personatge primer mira si s'ha passat el codi d'usuari. Si no s'ha passat mostra un selector per aquest element. Si pel contrari ha rebut l'usuari mostra la pantalla d'entrar dades de personatge. Aquesta última, un cop fa la crida per inserir, també valida que hi hagi les dades de personatge i llavors entra en la opció d'actualització de la BD.

Validació de dades

Per evitar varis errors basics, s'utilitzen selectors de manera que les dades ja siguin vàlides.

Per les altres dades s'utilitzen varies formes.

Camps de tipus text:

- S'utilitzen inputs del tipus text.
- Es validen les llargades de les cadenes utilitzant l'atribut "maxlenght" del control.
- Es validen els camps obligatoris amb l'atribut "required".

Camps de tipus número:

- Es controla amb els inptus de tipus "number".
- Amb el "max" es pot indicar la 'llargada màxima' del número.
- Amb l'atribut "step" es pot indicar si es seguirà un valor enter (indicant un número) o si permetrà decimals (valor=any).
- Permet que el camp sigui obligatori amb "required"
- Amb "value" se li donar un valor per defecte.

Camps de tipus data:

- S'utiliten inputs del tipus date.
    - El Firefox no permet aquets tipus de input. El funcionament original es mostrar un calendari de forma que es pugui seleccionar còmodament una data. Per aconseguir aquesta funcionalitat s'ha afegit un javascript al header, però en algun cas no s'acaba de mostrar en el lloc correcte.
    - També s'ha de dir que el Chrome si funciona be el date però crec que al posar l'script nou fa que hi hagi comportaments estranys.
- El format de la data es MM/DD/AAAA degut al script per mostrar el calendari, també modifica com es guarda la data. Es deixa així per no tenir mes problemes alhora de recollir el valor i actualitzar la BD.

Camps de tipus hora:

- S'utilitzen inputs del tipus text. (Hi ha un tipus time però igual que la data no funciona amb Fifefox.
- Donat que es tipus text s'aplica un placeholder per indicar quin ha de ser el format del contingut HH:MM:SS. Ja que només es una ajuda visual (no valida res), s'afageix també una validació per expressió regular "pattern". D'aquesta manera no es podrà continuar si s'entra una cadena que no sigui en aquest format ja que la expressió donarà el resultat com a invàlid.

Utilitzant aquestes validacions pròpies del formulari s'aconsegueix que no es pugui avançar fins que totes les dades siguin correctes.

Codificació de les pàgines

Exceptuant la pàgina d'alta de personatge, totes les pàgines tenen una codificació similar. Contenen una codificació amb html i parts de php.

La estructura de la pagina esta amb tags html mentre que el comportament que ha de tenir ve regida per els blocs php.

Contenen vairs blocs "div" que es mostren o s'oculten depenen del pas on estiguin (els paràmetres POST indiquen si ja s'ha entrat un primer codi per filtrar possibles valors o si estan a la "pantalla inicial" de la opció).

Els blocs PHP també contenen les sentencies insert/Update així com els selects necessaris.

La pàgina d'alta de personatge però es diferent. Tota la pàgina es en php de manera que l'html resultat es genera tot dins del bloc php. No es per cap motiu en especial. La primera pàgina que vaig fer era aquesta i al veure que quedava tot una mica confús vaig decidir fer les altres barrejant html i php.

Encara que potser seria millor haver-ho fet tot en un únic bloc php que controli el resultat html final. Ja que no hi ha blocs ocults que podrien portar problemes en determinats casos.

Funcions

He afegit un mòdul a part de funcions que em permet cridar-les des de qualsevol pàgina. La idea era afegir tantes coses com pogués aquí per que el php final fos més simple. Tot i que al final només m'han calgut 2 funcions.

OmpleCombo:

```
<?php
function ompleCombo($select,$postName,$id,$descrip=false,$required=false){
//Pre: $select es una consulta ben construida
//Post: construieix un select-option amb nom $postName i els valor del
resultat de $select. amb opcions de obligatori($required)
//               i mostrar el nom o descripcio ($descrip)


        echo "<select name=\"".$postName."\" id=\"".$id."\"
".($required?'required="required"':'')." >\n";
        echo "<option ".($required?'':'value="blank"')." > </option>\n";
        while ($row = oci_fetch_array($select, OCI_BOTH)) {
                echo "<option value=\"".$row[0]."\">".$row[($descrip ? 1 :
0)]."</option>\n";

        }
        echo "</select>";

}
?>
```

Rep una $select executada i omple un control de tipus selector per un formulari tipus POST. Li posa el "name" un "id" i els valors que conte la select.

Te dos paràmetres opcionals:

- $descrip indica si la select conte un camp que serà el que es mostrara, en cas negatiu es mostrara el codi.
- $required indica si el serà un input obligatori, si es així el primer element que es mostra sempre en blanc no tindra valor, de forma que si no es posa cap altre valor no validarà.

Exist:

```
<?php
function exist($select,$nom){
//Pre: select es una consulta correcte
//Post: retorna cert si $nom existeix a la base de dades

        $existeixCodi=false;
        //nom: camp que es selecciona
        //num: variable que contidra el valor del camp seleccionat $nom
        oci_define_by_name($select, $nom, $num);
        oci_execute($select);

        while (oci_fetch($select)){
                if($num>0)
                        $existeixCodi=true;
        }

        return $existeixCodi;

}
?>
```

Es una funció simple que retorna cert si el valor que se li passa a $nom ja existeix a la base de dades. Utilitza la sentencia $select que ja ha d'estar executada.

Actualitzacions de la BD

Quan els controls han validat que totes les dades siguin correctes i al fer clic al boto del formulari s'entra en l'apartat de modificació de la base de dades (en el cas de les opcions que puguin modificar).

Per totes les pàgines es recullen les dades del POST i es munta la sentencia insert/Update corresponent.

En algunes però, es fan verificacions de control de primary keys. Es mira si els camps que s'han enviat que son claus primaria no existeixin ja a la Base de dades. També el cas de inscripció que un personatge o vehicle no s'hagi inscrit ja a la cursa (aquesta particularitat no l'he trobat en l'enunciat però ho trobo correcta. Per exemple: si un personatge ja ha estat inscrit en una cursa no hauria de poder-ho fer amb un altre vehicle per exemple).

En aquests casos s'informa amb un missatge d'error i es torna a carregar la pàgina de modificació de dades per tal que es pugui tornar entrar la informació correcta.

Entrada de temps

Tot i que no calia entrar els temps de cop (es deia que podien ser d'un en un) comentaré el mètode que he utilitzat per poder entrar varis temps alhora.

Es tracta d'utilitzar una matriu per el paràmetre POST muntada de la seguent forma (tot i que hi ha més formes de fer-ho).

```
<?php
$query="select u.alias as nom, p.alias, c.vehicle ".
        "from usuaris u inner join personatges p  on u.alias=p.usuari".
        " inner join participantsCurses c on p.alias=c.personatge ".
        "where c.cursa='".$_POST['cursa']."' and c.temps is null";

$select=oci_parse($conn,$query);
oci_execute($select);

$i=0;
while(($row=oci_fetch_array($select,OCI_ASSOC+OCI_RETURN_NULLS)) !==false){
        if($i%2==0)
                $classname="evenRow";
        else
                $classname="oddRow";
        echo '<tr class="'.$classname.'" >';
        $valor=$_POST['cursa'];
        foreach ($row as $item) {
                echo "<td>".($item !== null ? htmlentities($item,
                                ENT_QUOTES) : " ")."</td>\n";
                $valor=$valor.";".($item !== null ? htmlentities($item,
                                        ENT_QUOTES) : " ");
        }

        echo '<td><input type="text" name="temps['.$valor.']"
        pattern="(?:[01]|2(?![4-9])){1}\d{1}:[0-5]{1}\d{1}:[0-
        5]{1}\d{1}" placeholder="hh:mm:ss" /> </td>';
        echo "</tr>\n";
        $i++;
}
?>
```

Amb un doble bucle per emplenar columnes i files guardo primer a $valor l'id de la cursa.

Dins del bucle que munta la fila (td) es concatenen els altres valors: usuari, personatge i vehicle separats per ';'.

Un cop $valor te tots els identificadors es posa com a "name" del paràmetre POST però no de forma normal, sinó utilitzant una array→name="temps[$valor]".

Això permet llavors recuperar els tots els valors com una array i fer un bucle per a cada "participant" de la cursa.

```php
<?php
while( list($key, $value) = each($_POST["temps"])){
        list($cursa, $usuari, $personatge, $vehicle)=split(";",$key);
   /*
  Resta de linies
        /*
}
?>
```

Amb els mètodes list i each podem obtenir la "clau" de la tupla "codis:temps" i, fent un Split per ';' que abans hem posat obtenir tots els valors codi que identifiquen el registre.

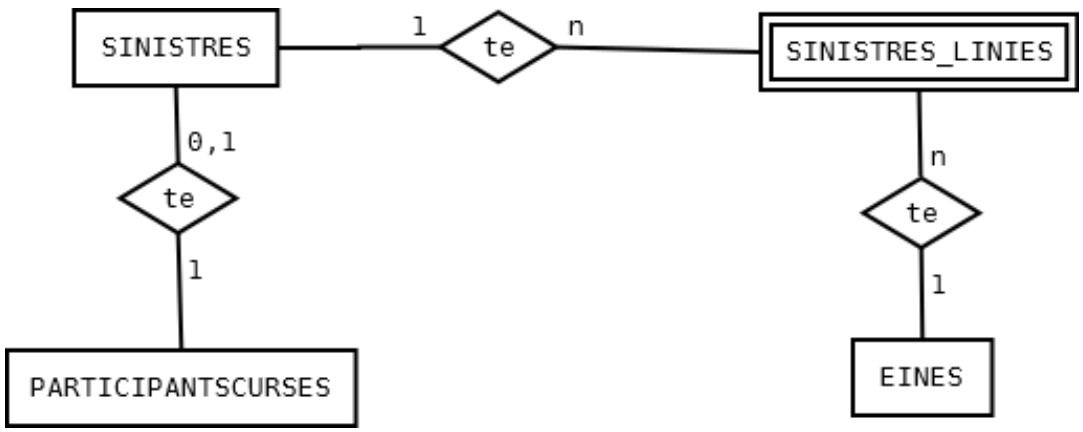
Gestió de sinistres

He escollit utilitzar dues taules per la informació dels sinistres:

Sinistres: Conte la informació de la cursa, personatge, vehicle i gravetat del sinistre. La clau principal es un codi autonumèric gestionat per una seqüencia.

Sinistres_linies: una entitat feble que depèn de sinistres conte informació de l'eina i el cost que s'han utilitzat per reparar els vehicle (cada eina en una línia diferent: 1,2,3.. de fet nomes fins a tres per a cada sinistre ja que es el màxim segons la gravetat).

Entitat relació



Model relacional

SINISTRES(**Codi**,vehicle, cursa, personatge, dataReparacio, gravetat)

SINISTRES_LINIES(**Sinistre,Linia**, eina, cost, compatibilitat)

Actualització

La primera opció era fer un bucle en php per a cada participant de la cursa corresponent i anar actualitzant cada un dels vehicles abandonats. Inserint-los primer a la taula de sinistres, assignant-los la gravetat corresponent. I inserir les eines necessàries per reparar el vehicle, actualitzant també la taula de einesVehicles on apareixen les eines ja utilitzades.

Però, ja que estem fen BD i SQL he pensat que seria mes interessant construir consultes que ho fessin tot això. (les sentencies queden mal tabulades al document pero no hi ha intros, de manera que si es copia en un notepad surt be).

Taula Sinistres:

```
insert into sinistres(codi,cursa,vehicle,personatge,dataReparacio,gravetat)
select sinistres_seq.nextval, p.cursa, p.vehicle, p.personatge, sysdate as
dataReparacio,
                        case when s.totalSinistre is null then 1 when
s.totalSinistre=1 then 2 else 3 end as gravetat
from participantsCurses p left outer join

        (select vehicle, count(vehicle) as totalSinistre

        from sinistres

        group by vehicle) s
        on p.vehicle=s.vehicle
where p.cursa='codi de cursa aquí' and p.temps is null;
```

Selecciona els registres de la taula particpiantsCurses que no tinguin temps i fa un join amb una taula (sinistres) que conte el nombre de vegades que un vehicle ha estat sinistrat. A partir d'aquest número assigna la gravetat.

Taula Sinistres_Linies:

```
insert into sinistres_linies(sinistre, linia,eina,cost, compatibilitat)
select sinis, linia, eina, preureparacio, compatibilitatDefecte
from(
        select s.codi sinis, comp.* , gravetat ,preuhoralloguer,  preuhoralloguer*3
preureparacio from (
                select codi,  cursa, propietari, eg.eina, grup,
compatibilitatDefecte, nvl(horesusada,0) ,
                                dense_rank() over (partition by codi order by
eg.compatibilitatDefecte desc, nvl(horesusada,0), eg.eina) linia
                from einesGrupVehicles eg inner join vehicles v on
eg.grup=v.grupVehicle
                        inner join participantsCurses p on p.vehicle=v.codi
                        left outer join einesVehicles ev on p.vehicle=ev.vehicle
and eg.eina=ev.eina
                where cursa='Codi de cursa aqui' and temps is null
        ) comp inner join sinistres s on comp.codi=s.vehicle and comp.cursa=s.cursa
                inner join eines e on comp.eina=e.codi
        where linia<=gravetat
) linies
order by sinis, linia
```

Primer de tot, hi ha un nivell de select que no seria necessari però per comprovar que treu esta be, sense haver de tornar-la a muntar.

La select mes interna fa els joins necessaris per treure les eines compatibles de cada vehicle de la cursa que s'està gestionant. Aquí s'obtenen totes les dades necessàries per llavors agafar les eines més compatibles. Amb la funció **dense_rank()** obtenim el ranking d'eines compatibles: l'eina mes compatible tindrà un 1 la segona mes compatible un 2 i així successivament. El fet d'usar el dense_rank() en comptes del rank() es perquè el dense_rank() no se salta números en cas d'empat (tot i així al order by del partition hi he afegit el codi de l'eina perquè no pugui passar).

El seguent nivell de select fa el join amb sinistres i obté nomes les línies necesaries segons la gravetat: si tenia gravetat 1 només agafarà la eina mes compatible, si tenia 2 les dues mes compatibles i el mateix per 3.

Taula EinesVehicle

```
Merge into einesVehicles ev
using (  select s.codi, s.cursa, s.vehicle, s.personatge, sl.eina, sl.linia,
sl.cost, s.gravetat, sl.compatibilitat
                from sinistres s inner join sinistres_linies sl on
s.codi=sl.sinistre
                where s.cursa='codi de cursa aquí'
           ) linies
        on (ev.eina=linies.eina and ev.vehicle=linies.vehicle)
        when matched then Update
                set ev.horesUsada=horesUsada+3,
                        ev.compatibilitat=linies.compatibilitat
        when not matched then insert (vehicle, eina,
compatibilitat,horesUsada)
                values(linies.vehicle,linies.eina,linies.compatibilitat,3);
```

Es un merge normal i corrent fent el match per codi d'eina i vehicle.