

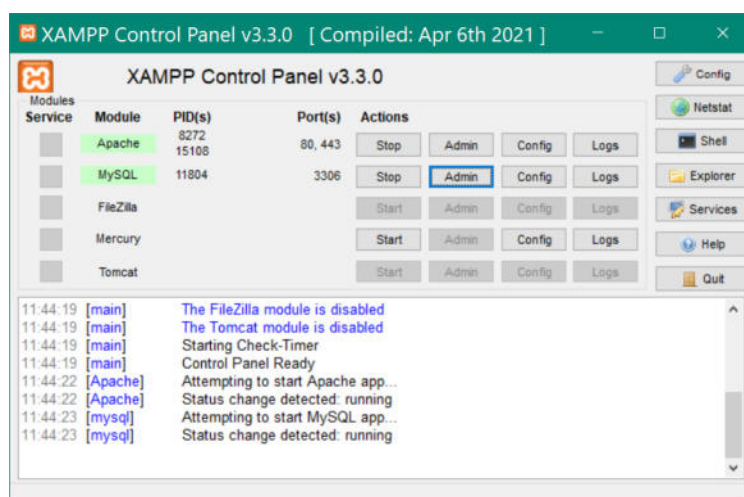
# Informatică Aplicată 1 – Îndrumar de Laborator

## Capitolul 5 – Lucrul cu Baze de Date în PHP

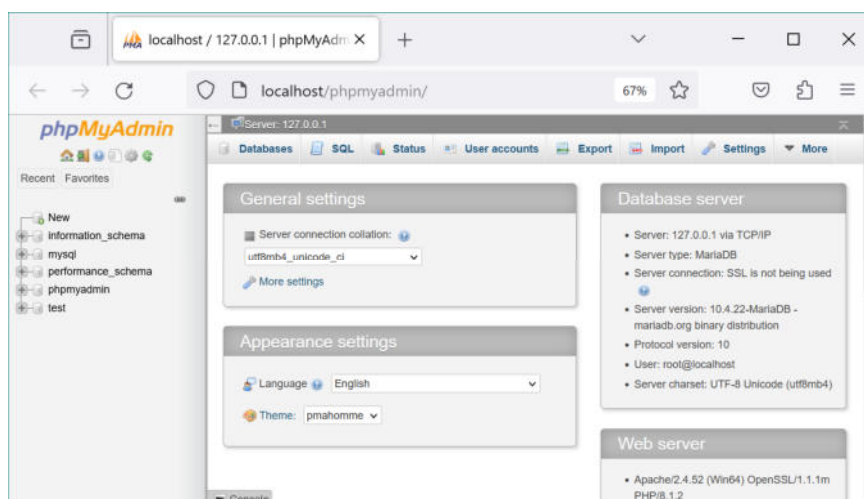
### 1) Utilizarea phpMyAdmin

În acest capitol vom vedea cum putem avea acces la datele stocate pe server într-o bază de date. Bazele de date sunt prezente în majoritatea activităților cotidiene, precum rezervarea unui bilet de tren sau avion, căutarea unor informații online, extragerea unei sume de bani de la bancomat, gestiunea studenților într-o aplicație online a universității etc. Desigur, pentru domeniul vast al bazelor de date este prevăzut un curs separat. Aici vom testa doar câteva exemple simple din domeniul bazelor de date accesibile prin Internet.

Pentru început, porniți XAMPP Control Panel și apăsați pe butoanele “Start” din dreptul Apache și MySQL. Modulele ar trebui să apară cu verde ca în figura următoare. MySQL este un server de baze de date, adică un pachet software care se ocupă cu gestiunea bazelor de date. Pentru a putea rula exemplele următoare, atât serverul de web Apache, cât și serverul MySQL trebuie să ruleze.

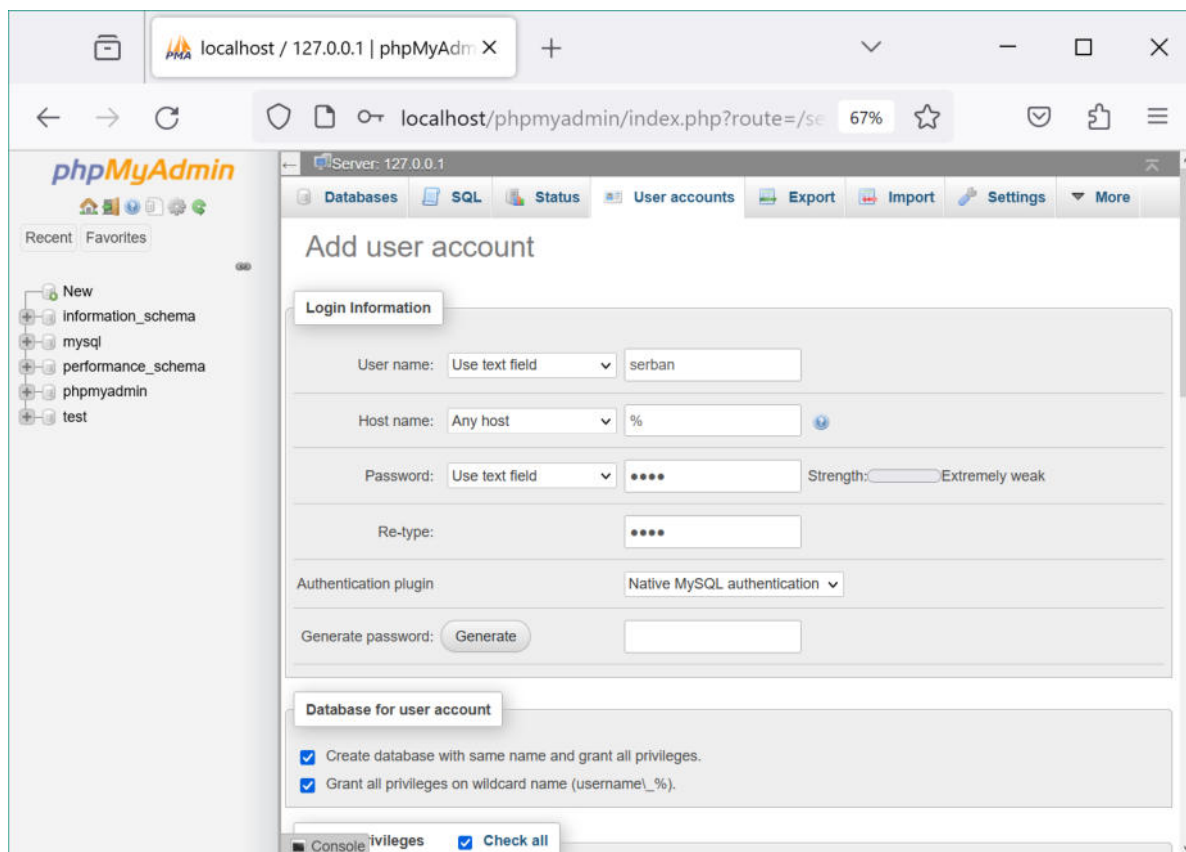


Apăsați pe butonul „Admin” din dreptul MySQL pentru a accesa *phpMyAdmin*.



Vom folosi aplicația phpMyAdmin pentru a accesa în mod grafic o bază de date, a crea o tabelă și a insera linii în aceasta. Fiind un capitol introductiv, nu vom aprofunda aici comenzile SQL necesare pentru a efectua aceste operații, ci ne vom baza pe interfața grafică a programului.

**Primul pas:** creați un nou cont de utilizator apăsând pe link-ul „User accounts” și apoi „Add user account”.

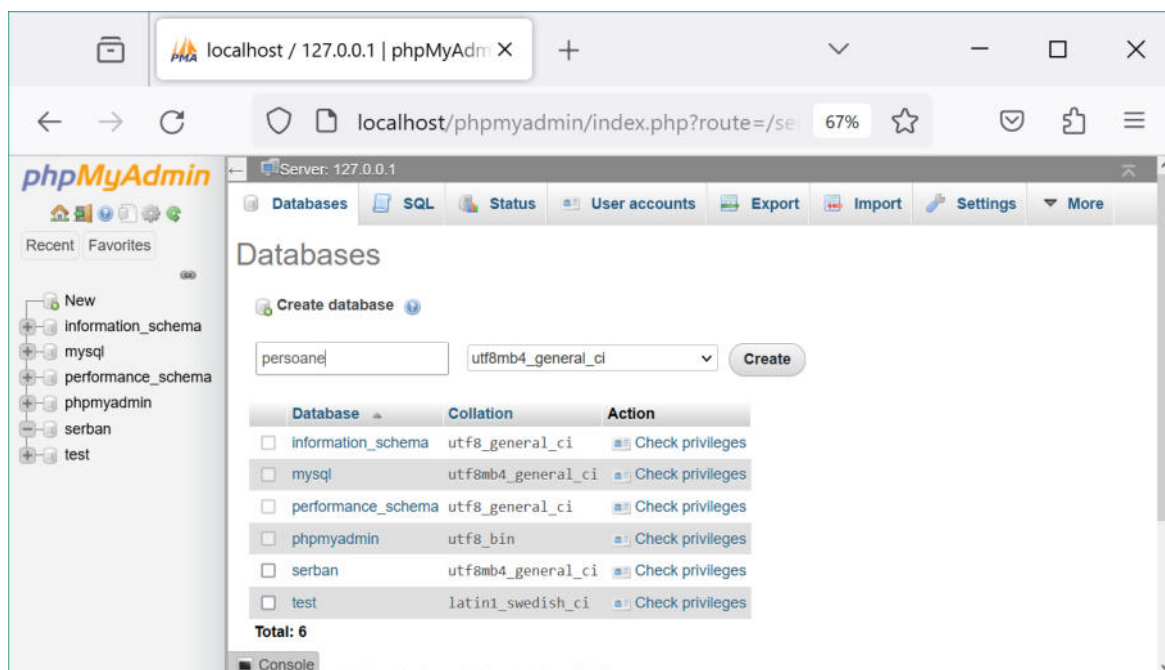


The screenshot shows the 'Add user account' form in phpMyAdmin. The 'Login Information' section includes fields for 'User name' (set to 'serban'), 'Host name' (set to 'Any host' and '%'), 'Password' (masked with dots), and 'Re-type' (also masked). The 'Authentication plugin' is set to 'Native MySQL authentication'. There are checkboxes for 'Create database with same name and grant all privileges' and 'Grant all privileges on wildcard name (username\\_%)'. A 'Generate password' button is also present. The 'Database for user account' section has the same two checkboxes checked. At the bottom, there is a 'Check all' button under the 'Privileges' section.

Alegeți un nume de utilizator și o parolă. Bifați opțiunile „Create database with same name and grant all privileges”, „Grant all privileges on wildcard name (username\\_%)” precum și „Global privileges Check all”. Apăsați pe „Go”.

**Pasul al doilea:** creați o bază de date și o tabelă.

Apăsați pe link-ul „Databases”, alegeți un nume pentru baza de date, de exemplu „persoane” ca în figura următoare, și apoi apăsați butonul „Create”.



The screenshot shows the 'Databases' page in phpMyAdmin. At the top, there is a 'Create database' section with a text input field containing 'persoane' and a dropdown menu set to 'utf8mb4\_general\_ci'. A 'Create' button is next to it. Below this is a table listing existing databases with columns for 'Database', 'Collation', and 'Action'.

Database	Collation	Action
<input type="checkbox"/> information_schema	utf8_general_ci	<a href="#">Check privileges</a>
<input type="checkbox"/> mysql	utf8mb4_general_ci	<a href="#">Check privileges</a>
<input type="checkbox"/> performance_schema	utf8_general_ci	<a href="#">Check privileges</a>
<input type="checkbox"/> phpmyadmin	utf8_bin	<a href="#">Check privileges</a>
<input type="checkbox"/> serban	utf8mb4_general_ci	<a href="#">Check privileges</a>
<input type="checkbox"/> test	latin1_swedish_ci	<a href="#">Check privileges</a>

Total: 6

Sistemul va trece automat în fereastra în care puteți adăuga o tabelă la baza de date „persoane” creată. Alegeți de exemplu numele tabeli „Agenda” și lăsați numărul de coloane egal cu 4, apoi apăsați butonul „Go”. Va apărea o fereastră precum următoarea:

The screenshot shows the phpMyAdmin interface with the 'Create Table' dialog open. The table name is 'Agenda' and it is located in the 'persoane' database. The dialog shows 4 columns to be added, each with a default type of 'INT' and a length of 1. The 'Go' button is visible.

Aici putem alege denumirea celor patru coloane ale tabeli, precum și tipul de date al fiecărei coloane.

Alegeți următoarele coloane, în ordine: ID, Nume, Prenume, Telefon. Cu tipurile de date: INT pentru ID, VARCHAR pentru Nume, Prenume și Telefon. Câmpul „Length” semnifică numărul maxim de cifre sau caractere permis pentru fiecare coloană. Un exemplu de completare este următorul. Bifați căsuța „A\_I” (AUTO\_INCREMENT) pentru coloana ID.

The screenshot shows the phpMyAdmin interface with the 'Create Table' dialog open. The table name is 'Agenda' and it is located in the 'persoane' database. The dialog shows 4 columns to be added:

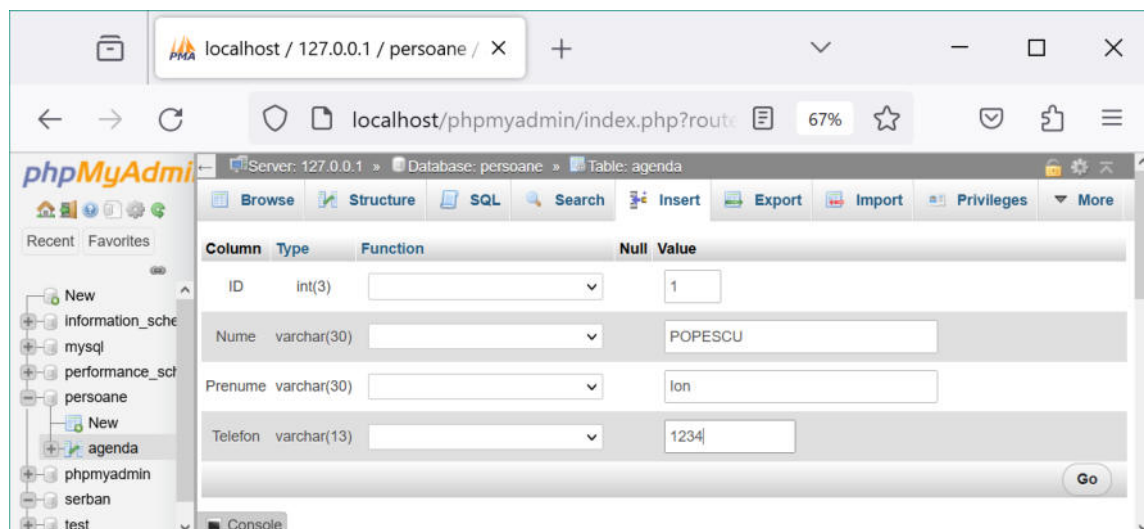
Name	Type	Length/Values	Default	Attributes
ID	INT	3	None	<input checked="" type="checkbox"/> A_I
Nume	VARCHAR	30	None	
Prenume	VARCHAR	30	None	
Telefon	VARCHAR	13	None	

The 'Go' button is visible.

Lăsați neschimbate celelalte opțiuni și apăsați pe butonul „Save” pentru a crea tabela. În acest moment am creat tabela „Agenda” în baza de date „persoane”, dar tabela nu conține nici o linie.

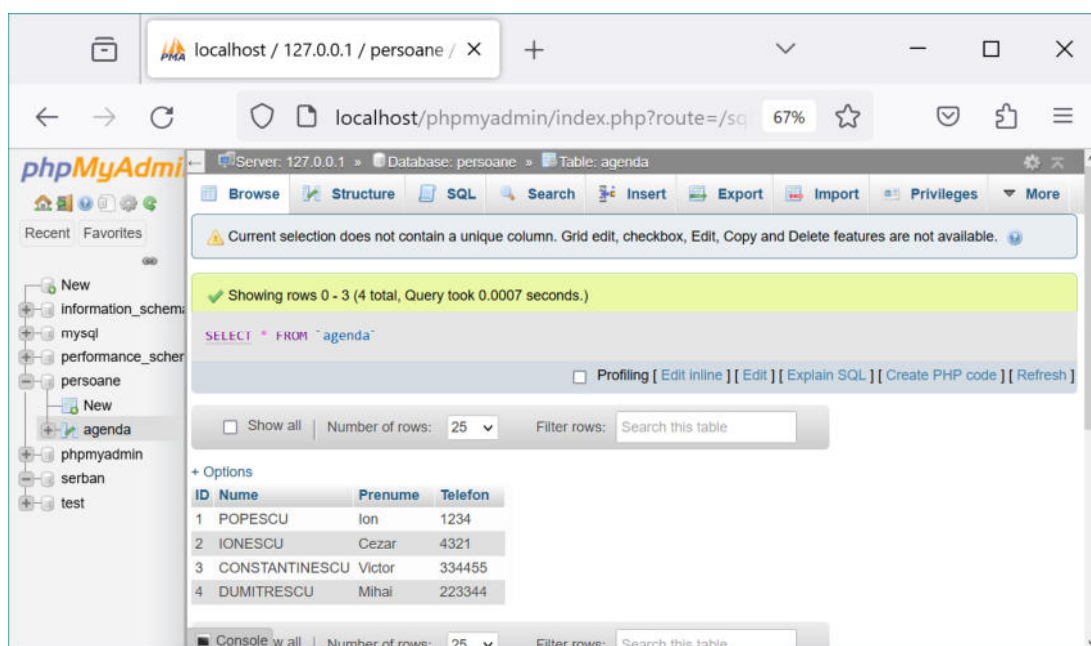
**Pasul al treilea:** inserați câteva linii în tabela „Agenda”.

Pentru a insera linii în tabela „Agenda” apăsați tab-ul „Insert” precum în figura următoare.



Inserați datele la rubrica „Value”, apoi apăsați butonul „Go” pentru a insera linia. Inserați măcar 4-5 linii în tabelă.

Pentru a vedea apoi conținutul tabelii, apăsați pe tab-ul „Browse”. Vom obține o pagină precum cea din figura următoare.



## 2) Accesarea unei baze de date din PHP

PHP include funcții pentru accesul la diverse sisteme de gestiune a bazelor de date, cel mai utilizat de programatori fiind MySQL.

Începând cu versiunea PHP5 extensia „mysql” ce permite comunicarea cu serverul MySQL a fost îmbunătățită, noua extensie având numele „mysqli” (*MySQL improved*). Aceasta este extensia recomandată în prezent, extensia veche (mysql) urmând a fi eliminată din distribuțiile viitoare.

Printre îmbunătățirile aduse de `mysqli` menționăm: suportul pentru programarea obiect-orientată, suport pentru operații avansate cu bazele de date precum trimiterea de interogări multiple, interogări pregătite, proceduri stocate, tranzacții, capacități extinse de depanare etc.

Principalele etape în lucrul cu o bază de date MySQL sunt:

- Deschiderea unei conexiuni cu serverul MySQL.
- Selectarea bazei de date cu care se va lucra.
- Transmiterea de interogări SQL pentru definirea/manipularea datelor.
- Închiderea conexiunii cu serverul MySQL.

Toate aceste patru etape trebuie parcurse în ordine de fiecare dată când dorim să lucrăm cu o bază de date.

### *Conectarea la serverul MySQL*

Se face apelând funcția `mysqli_connect` ce are următoarea sintaxă:

`mysqli_connect(ume_server, ume_utilizator, parola, ume_baza_date);`

Exemplu:

```
$db = mysqli_connect("localhost", "serban", "1234", "persoane");
if (!$db) {
    exit('Conectare esuata: ' . mysqli_connect_error());
}
echo 'Conectare reusita';
```

În cazul unei conectări nereușite, variabila `$db` va primi valoarea FALSE. În această situație programul se termină cu un mesaj de eroare. `localhost` este numele serverului MySQL pe calculatorul local. Parola este cea stabilită înainte pentru utilizatorul creat.

### *Selectarea bazei de date*

Se realizează cu funcția `mysqli_select_db`(legătură, ume\_baza\_de\_date).

În mod normal această funcție nu mai este necesară decât în cazul în care dorim schimbarea bazei de date specificate ca al patrulea parametru în funcția `mysqli_connect`.

Exemplu:

```
$succes = mysqli_select_db($db, "persoane");
if (!$succes) {
    exit('<br>Nu se poate selecta baza de date: ' . mysqli_error($db));
}
```

Variabila booleană `$succes` permite detectarea erorilor. După deschiderea unei conexiuni la baza de date, se poate testa apariția unei erori și cu: `if(mysqli_errno($db))`.

Închiderea conexiunii cu baza de date se face apelând funcția `mysqli_close($db)`;

### *Executarea de interogări SQL*

Funcția folosită pentru a trimite o interogare la server este `mysqli_query(interogare)`.

Următorul exemplu arată modalitatea de preluare a rezultatului unei interogări:

```
$interogare = "SELECT * FROM Agenda";
$raspuns = mysqli_query($db, $interogare);
if (mysqli_errno($db))
    exit('<br>'.mysqli_errno($db).": ".mysqli_error($db)."<br>");
$N = mysqli_affected_rows($db);
echo "<br>Interogarea a returnat $N linii : <br>";
```

```

for($i=0; $i<$N; $i++){
    $linie = mysqli_fetch_row($raspuns);
    echo "<br>", $linie[0], " ", $linie[1], " ", $linie[2], " ", $linie[3];
}

```

Comanda SQL „SELECT \* FROM Agenda” scrisă mai sus determină serverul de baze de date să returneze tot conținutul tabelului Agenda, adică toate coloanele și toate liniile.

Variabila \$interogare este folosită ca parametru al funcției *mysqli\_query*, care va returna o referință ce va fi stocată în variabila de tip „resursă” \$raspuns. Tipul resursă este un tip de date special introdus în PHP4 pentru a păstra o legătură (sau o referință) către o resursă externă precum baze de date, fișiere etc. Astfel \$raspuns va indica zona de memorie în care se află rezultatul rulării interogării, rezultat la care putem avea acces folosind diferite funcții.

Una din funcțiile uzuale este *mysqli\_affected\_rows()* care va returna numărul de linii al rezultatului. Funcția care preia fiecare linie din rezultat este *mysqli\_fetch\_row()*. Variabila \$linie este un vector ce are ca elemente valorile fiecărei coloane din rezultat, de exemplu: ID, Nume, Prenume și Telefon.

Opțional, dacă dorim eliberarea zonei de memorie utilizată pentru stocarea rezultatului interogării, putem apela funcția: *mysqli\_free\_result(\$raspuns)*;

Pentru a afla numărul de coloane din rezultat putem folosi funcția *mysqli\_num\_fields*, de exemplu:

```
$ncol = mysqli_num_fields($raspuns);
```

### Exercițiu:

Modificați programul anterior pentru a afișa doar primele 3 linii într-un tabel HTML cu bordura vizibilă, având capul de tabel: ID, Nume, Prenume, Telefon.

O variantă simplificată de a afișa rezultatul interogării de mai sus este următoarea:

```

while($linie = mysqli_fetch_row($raspuns)){
    foreach ($linie as $scoloana)
        echo " $scoloana ";
    echo "<br>";
}

```

Bucula *while* se va termina când *mysqli\_fetch\_row* va returna FALSE (nu mai există linii).

Instrucțiunea *foreach* are efect doar pentru variabile de tip vector; în exemplul de mai sus, variabila \$scoloana va primi pe rând fiecare element al vectorului \$linie.

Dacă în loc de indici numerici (0,1,2,...) pentru coloane se dorește utilizarea indicilor de tip șir de caractere conținând numele coloanelor din tabelă, putem folosi funcția *mysqli\_fetch\_array*.

### Exemplu:

```

while($linie = mysqli_fetch_array($raspuns, MYSQLI_ASSOC)){
    foreach ($linie as $nume => $scoloana)
        echo " $nume => $scoloana ";
    echo "<br>";
}

```

Parametrul MYSQLI\_ASSOC determină ca fiecare linie din tabel să fie preluată ca un vector asociativ, adică având drept cheie (indice) numele coloanei.

Observați de asemenea a doua modalitate de utilizare a instrucțiunii *foreach* cu sintaxa *... as \$cheie => \$valoare\_element*.

### 3) Interacțiunea cu baza de date prin formulare HTML

O aplicație foarte utilă și larg răspândită este utilizarea formulelor HTML pentru a introduce date într-o baza de date, sau pentru a efectua diverse interogări (căutarea unei persoane, ștergerea unor linii din tabele, actualizări etc.).

#### Exemplu:

Salvați următorul cod HTML într-un fișier text cu numele *agenda.html*

```
<HTML>
<HEAD> <TITLE> Agenda telefonica </TITLE> </HEAD>
<BODY>
<h3>Introduceti datele</h3>
<FORM METHOD="GET" ACTION="agenda.php">
  Nume:<BR>
  <INPUT TYPE=TEXT SIZE=15 MAXLENGTH=20 NAME="nume">
  <BR>
  Prenume:<BR>
  <INPUT TYPE=TEXT SIZE=15 MAXLENGTH=20 NAME="prenume">
  <BR>
  Telefon:<BR>
  <INPUT TYPE="TEXT" SIZE=15 MAXLENGTH=15 NAME="telefon">
  <BR>
  <BR>
  <INPUT TYPE="SUBMIT" VALUE="OK"> <INPUT TYPE="RESET">
</FORM>
</BODY>
</HTML>
```

Formularul HTML de mai sus va trimite datele către scriptul din fișierul *agenda.php* următor:

```
<?php
$nume = $_GET["nume"]; $prenume = $_GET["prenume"]; $telefon = $_GET["telefon"];

$db = mysqli_connect("localhost", "serban", "1234", "persoane");
if (!$db) {
    exit('Conectare esuata: ' . mysqli_connect_error ());
}
echo 'Conectare reusita';

$interogare = "INSERT INTO Agenda (Nume, Prenume, Telefon)
                VALUES ('$nume' , '$prenume' , '$telefon' )";
mysqli_query($db,$interogare);

if (mysqli_errno($db))
    exit('<br>Adaugare esuata: '.mysqli_errno($db).": ".mysqli_error($db)."<br>");

echo "<br> $prenume $nume a fost adaugat in agenda.";

mysqli_close($db);
?>
```

Verificați în phpMyAdmin faptul că linia cu datele din formular a fost inserată în tabelă.

Notă: formularul și scriptul apelat de formular de mai sus folosesc metoda GET (doar ca exemplu, deoarece metoda recomandată este POST). Observați transmiterea perechilor (nume\_variabilă, valoare\_variabilă) în cazul metodei GET, prin intermediul adresei URL, de exemplu:  
*http://localhost/agenda.php?nume=Pop&prenume=Ion&telefon=123*

#### Câteva funcții MySQL suplimentare:

- *\$reusit = mysqli\_data\_seek(\$raspuns, \$i);*  
Funcția *mysqli\_data\_seek* permite poziționarea pointerului curent care indică linia din rezultatul obținut cu funcția *mysql\_query*. Adică, putem indica ce linie din rezultat dorim să extragem prin apelul funcției *mysql\_fetch\_row*. Numărul primei linii este 1. Variabila *\$reusit* întoarce TRUE dacă s-a reușit setarea pointerului și FALSE altfel (pointer invalid).
- *\$baze\_de\_date = mysqli\_list\_dbs();*  
În variabila de tip resursă *\$baze\_de\_date* vom obține lista tuturor bazelor de date. De fapt funcția are un comportament similar cu *mysql\_query*, adică rezultatul poate fi obținut apelând funcțiile *mysql\_num\_rows* etc.
- *\$tabele = mysqli\_list\_tables(„test”);*  
Vom obține lista tabelelor dintr-o bază de date, de exemplu „test”. Variabila *\$tabele* este de același tip cu *\$baze\_de\_date* de mai sus.

#### 4) Încărcarea de imagini pe server

Pentru a putea încărca imagini în baza de date trebuie întâi să realizăm o interfață de încărcare (upload) a unui fișier (de exemplu de tip imagine) într-un folder pe server.

Principiul este următorul: utilizatorul, sau clientul, aflat la distanță, are la dispoziție un formular prin care poate trimite un fișier de pe calculatorul local către server.

##### Exemplu:

Creați un fișier text cu numele *upload.html* și următorul conținut [1]:

```
<HTML>
<HEAD>
  <TITLE> Incarcarea unui fisier pe server </TITLE>
</HEAD>
<BODY>
  <h3>File upload</h3>
  <FORM METHOD="POST" ACTION="upload.php" enctype="multipart/form-data">
    Nume fisier:<BR>
    <INPUT TYPE=FILE NAME=nume_fisier>
    <BR>
    <INPUT TYPE="SUBMIT" VALUE="OK">
  </FORM>
</BODY>
</HTML>
```

Formularul de mai sus conține un control de tip FILE (fișier).

Apoi, creați în folderul *htdocs* un nou folder cu numele *imagini*.

Creați un nou fișier text cu numele *upload.php* și următorul conținut [1]:

```
<?PHP
if ($_FILES["nume_fisier"]["error"] > 0)
  echo "Eroare: " , $_FILES["nume_fisier"]["error"] , "<br>";
else
{
```



```

echo "Incarcare: " . $_FILES["nume_fisier"]["name"] . "<br>";
echo "Tip: " . $_FILES["nume_fisier"]["type"] . "<br>";
echo "Dimensiune: " . ($_FILES["nume_fisier"]["size"] / 1024) . " kB<br>";
echo "Temporar salvat in: " . $_FILES["nume_fisier"]["tmp_name"];
}
echo "<BR>";
if (file_exists("imagini/" . $_FILES["nume_fisier"]["name"]))
    echo "Fisierul " . $_FILES["nume_fisier"]["name"] . " exista deja. ";
else
{
    move_uploaded_file($_FILES["nume_fisier"]["tmp_name"],
        "imagini/" . $_FILES["nume_fisier"]["name"]);
    echo "Fisierul a fost salvat in: " . "imagini/" . $_FILES["nume_fisier"]["name"];
}
?>

```

Fișierul specificat în formularul upload.html va fi trimis către scriptul upload.php, care îl preia prin intermediul vectorului predefinit `$_FILES`. Observați modul de accesare a numelui, tipului, dimensiunii fișierului, precum și a locației temporare în care a fost stocat fișierul.

Fișierul este mutat în folderul imagini apelând funcția *move\_uploaded\_file(sursă, destinație)*.

Putem de asemenea restricționa (cu un *if*), în scriptul upload.php, tipul fișierului (de exemplu la „image/jpeg”) și dimensiunea, pentru a nu permite încărcarea oricărui fișier.

### Probleme propuse

- 1) Realizați un formular HTML conținând 2 controale de tip text: cerere\_sql și BD. Primul control va prelua o cerere SQL, de exemplu „SELECT \* FROM Agenda”, iar al doilea va prelua un nume de bază de date, de exemplu „Test”. Realizați scriptul PHP apelat de formular care să: afișeze dacă s-a realizat cu succes conectarea la BD, dacă cererea a fost executată, apoi numărul de linii rezultate și afișarea acestora, dacă cererea a fost un SELECT.

### *Referințe*

[1] [http://www.w3schools.com/php/php\\_file\\_upload.asp](http://www.w3schools.com/php/php_file_upload.asp)