CĂTĂLINA ENESCU

APLICAȚII WEB PAS CU PAS

PHP, MYSQL, MATERIAL DESIGN

EDITURA EVOMIND
2019

APLICAŢII WEB PAS CU PAS

PHP&MYSQL, MATERIAL DESIGN

Autor: Cătălina Enescu

ISBN 978-606-94762-7-7 (EPUB)

Editura Evomind, 2019

https://evomind.org/

CUPRINS

1 ARGUMENT

“Dacă tu ai un măr şi eu am un măr şi facem schimb de mere, atunci tu şi eu vom avea în continuarea un măr. Dar dacă tu ai o idee şi eu am o idee şi le schimbăm între ele atunci fiecare dintre noi va avea doua idei” – George Bernard Shaw

Culegerea prezintă pas cu pas modul în care se pot dezvolta mici aplicaţii web cu baze de date. Nivelul este de la simplu la complex, fiecare tutorial exersând principii introduse în tutorialul anterior şi introducând elemente noi de complexitate. Aplicaţiile sunt dezvoltate independent, dar, în acelaşi timp pot fi folosite ca nişte bucăţi de puzzle în dezvoltarea altor aplicaţii.

Astfel, această culegere se doreşte a fi un sprijin pentru cei care doresc să înveţe să dezvolte aplicaţii web cu baze de date să folosească tehnologii moderne, dar, în acelaşi timp, să se axeze pe principii general valabile, care nu depind de tehnologia folosită. Se poate folosi în activitatea de predare/ învăţare/ dezvoltare a elevilor/ profesorilor de liceu, studenţilor, tuturor celor care au o experienţă anterioară cu un limbaj de programare şi vor să treacă la programare web.

Pentru a putea folosi eficient aceste tutoriale, recomand:

- Noţiuni minime de baze de date, HTML, CSS, PHP, JavaScript sau disponibilitatea de a le acumula pe parcurs
- Existenţa unei infrastructuri care să permită implementarea şi testarea aplicaţiilor:
    - Apache + MySQL + PHP sau
    - WampServer - http://www.wampserver.com/en/ sau
    - XAMPP - https://www.apachefriends.org/index.html

2 NOMENCLATOR JUDEŢE ŞI LOCALITĂŢI. $_POST

Un nomenclator se defineşte ca fiind o “listă sau culegere care cuprinde nomenclatura unui domeniu de activitate, sistematizată după anumite criterii”[1].

În cazul nostru nomenclatorul va fi lista judeţelor din România, respectiv lista localităţilor din fiecare judeţ.

[1] Dicţionarul explicativ al limbii române

Ne propunem să creăm o bază de date care va memora aceste liste şi o interfaţă web prin intermediul căreia vom interoga baza de date.

2.1 Obiective

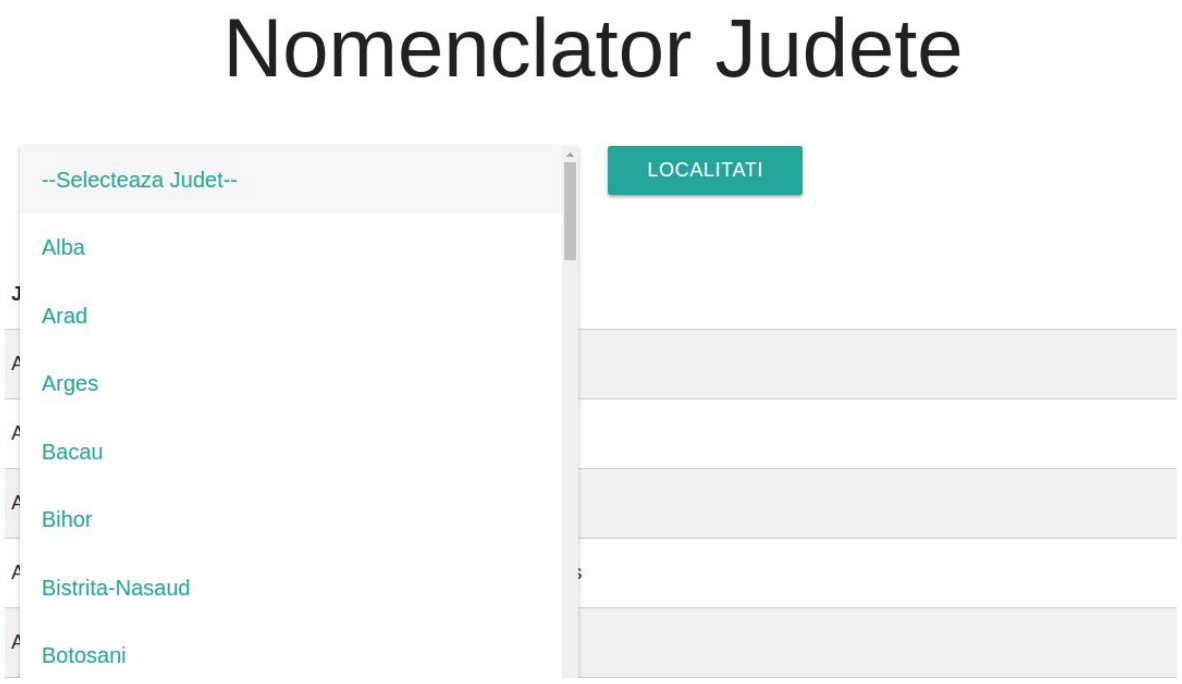
Parcurgând acest tutorial, vom putea face următoarele:

- Să proiectăm o bază de date cu două tabele
- Să importăm o bază de date MySQL în PHPMyAdmin
- Să creăm un formular HTML din care să preluăm datele cu PHP prin intermediul variabilei globale $_POST
- Să selectăm date din baza de date, să le prelucrăm cu PHP şi să afişăm rezultatele în browser cu ajutorul limbajului HTML

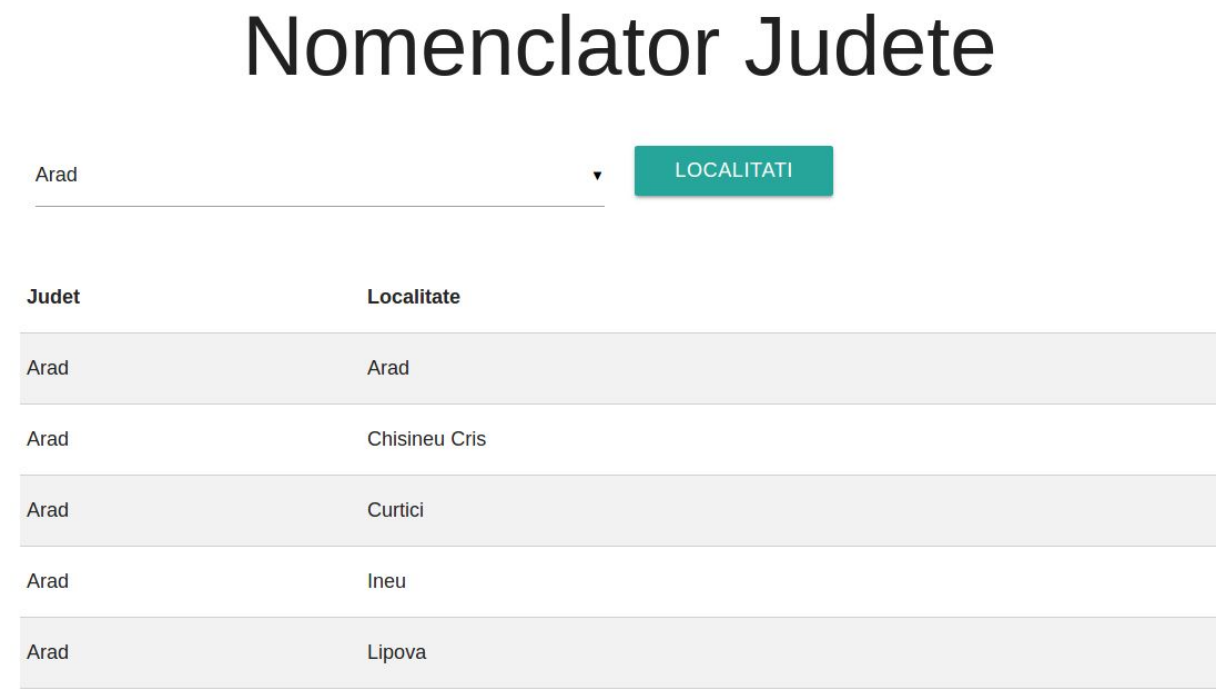
2.2 Rezultatul aşteptat

Aplicaţia pe care o vom realiza îi va permite utilizatorului următoarele:

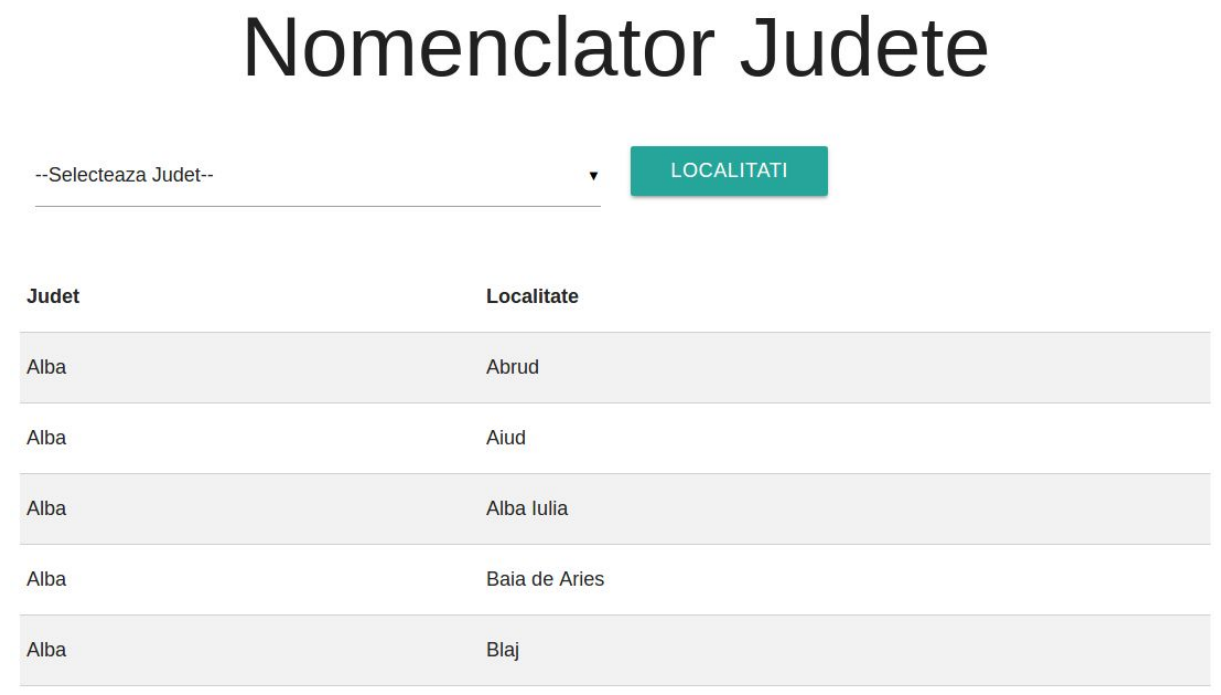
- Să selecteze un judeţ dintr-o listă de selecţie
- La click pe butonul Localităţi, va vizualiza lista localităţilor din judeţul selectat
- Dacă selecţia rămâne pe --*Selectează Judet*--, se vor afişa toate localităţile



Lista de selecţie conţine toate judeţele din România ordonate alfabetic.



La selecţia unui judeţ, după apăsarea butonului *Localităţi,* se vor afişa toate localităţile judeţului.



Dacă selecţia rămâne pe *--Selecteaza Judet--*, la apăsarea butonuli *Localităţi*, se vor afişa toate localităţile din ţară, grupate pe judeţe.

2.3 Proiectarea bazei de date

Pentru proiectarea bazei de date, vom parcurge următoarele etape:

1. Stabilirea entităţilor - obiectele de interes general
2. Stabilirea atributelor fiecărei entităţi
3. Stabilirea relaţiilor între entităţi şi a cardinalităţii fiecărei relaţii
4. Transformarea entităţilor în tabele
5. Transformarea relaţiilor
    a. În cheie străină dacă este o relaţie de cardinalitate *1 la 1* sau *1 la n*
    b. În tabel asociativ dacă este o relaţie de cardinalitate *n la n*

Entităţi: *judeţ, localitate*.

Atributele entităţilor

- *judet(id, nume, indicativ)*

Id este numărul de ordine al judeţului, ordinea fiind cea alfabetică.

Nume este denumirea judeţului. Exemplu: Arad.

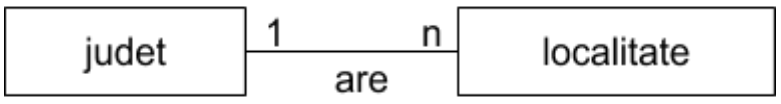
Indicativ este indicativul rutier al judeţului. Exemplu: AR pentru Arad.

- *localitate(id, nume, oras, resedinta)*

Oras are valoarea 1 dacă localitatea are statut de oraş, 0 în caz contrar.

Resedinta are valoarea 1 dacă localitatea are statut de reşedinţă de judeţ, 0 în caz contrar.

Relaţii: un judeţ are mai multe localităţi, o localitate este situată într-un singur judeţ. Deci, rezultă o relaţie *1:n (unu la n)* între entităţile *judeţ* şi *localitate* (1 la judeţ, n la localitate).



Transformarea entităţilor şi relaţiilor

Entitatile devin tabele:

Entitatea *judeţ* devine tabelul *nom_judete.*

Entitatea *localitate* devine tabelul *nom_localitati*.

Relaţia *1:n* devine cheie străină (câmpul *părinte*) în tabelul din partea *n*, deci în tabelul *nom_localitati*.

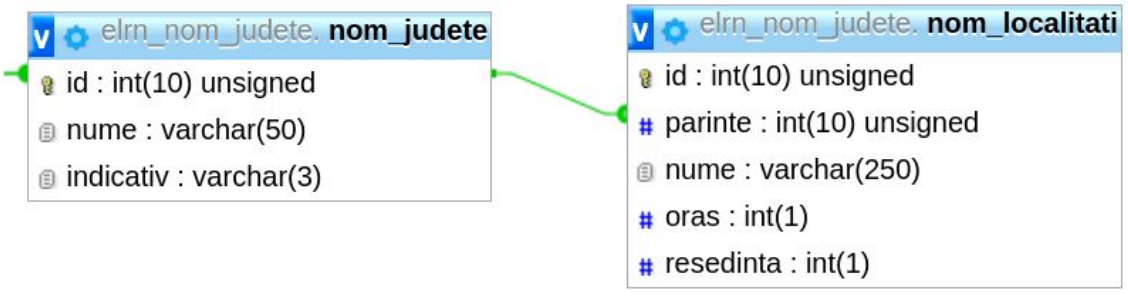


Diagrama este generată de componenta *Designer* a aplicaţiei *PHPMyAdmin.*

2.4 Structura aplicaţiei

Structura trebuie gândită astfel încât să permită o întreţinere cât mai uşoară a aplicaţiei, precum şi reutilizarea codului în rezolvarea altor probleme similare.

Varianta prezentată aici este doar un exemplu şi are la bază gruparea fişierelor în funcţie de tipul şi specificul acestora:

- În *config* avem fişierul *db.php* care conţine datele de configurare a bazei de date şi codul care realizează conexiunea cu baza de date.
- În *css* avem fişierele *.css* folosite pentru stilizarea paginii web. *Materialize.css* conţine foile de stil ale bibliotecii *materialize*, iar *style.css* va conţine regulile de stil personalizat.
- În *js* avem fişierele care conţin cod javaScript cu ajutorul căruia creăm elemente dinamice sau interactive ale paginii web.
- În *img* vom salva imaginile folosite în pagina web. Se recomandă ca imaginile să fie prelucrate în prealabil pentru web folosind un editor de imagine.
- În *sql vom avea:*
    - codul sql folosit pentru crearea iniţială a bazei de date

    - fişiere *diff.data.sql* care vor conţine modificările ulterioare asupra bazei de date; ex: *diff.2017.09.21.sql* conţine modificările făcute asupra bazei de date pe 21.09.2017.
- În *libs* am păstrat librăriile externe utilizate în aplicaţie: *materialize* şi *jquery*.

2.5 Etape de lucru

2.5.1 Importul bazei de date

Când lucrăm cu seturi mari de date aşa cum sunt şi nomenclatoarele, una dintre modalităţile de a construi baza de date este să importăm datele din fişiere *.csv* sau *.xml* sau chiar dintr-o bază de date existentă.

Pentru aplicaţia noastră, codul mySQL pentru crearea bazei de date se află în fişierul *sql/create.sql*. Mai jos este prezentat doar o parte din cod, volumul de date inserate în baza de date fiind foarte mare.

Fişierul se poate descărca accesând link-ul specificat în anexa *Fişiere*.

sql/create.sql

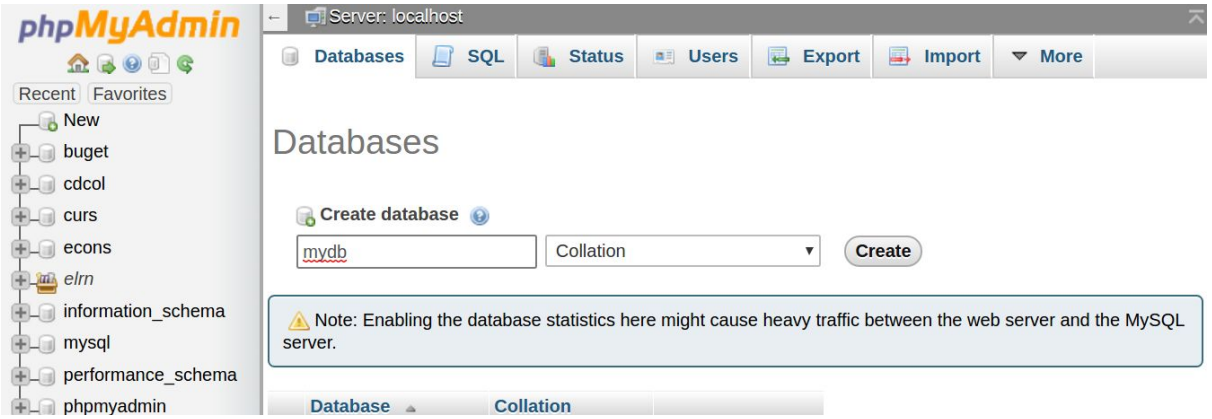
```
--
-- Table structure for table `nom_judete`
--

CREATE TABLE IF NOT EXISTS `nom_judete` (
`id` int(10) unsigned NOT NULL,
  `nume` varchar(50) NOT NULL,
  `indicativ` varchar(3) NOT NULL
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=54 ;

--
-- Table structure for table `nom_localitati`
--

CREATE TABLE IF NOT EXISTS `nom_localitati` (
`id` int(10) unsigned NOT NULL,
  `parinte` int(10) unsigned NOT NULL,
  `nume` varchar(250) NOT NULL,
  `oras` int(1) NOT NULL DEFAULT '0' COMMENT '1 dc e oras, 0 altfel',
  `resedinta` int(1) DEFAULT '0' COMMENT '1-reseinta de judet, 0 altfel'
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=3188 ;

--
-- Dumping data for table `nom_judete`
--
```

```sql
INSERT INTO `nom_judete` (`id`, `nume`, `indicativ`) VALUES
(1, 'Alba', 'AB'),
(2, 'Arad', 'AR'),
(3, 'Arges', 'AG'),
(4, 'Bacau', 'BC'),
(5, 'Bihor', 'BH'),
(6, 'Bistrita-Nasaud', 'BN'),
(7, 'Botosani', 'BT'),
(8, 'Braila', 'BR'),
(9, 'Brasov', 'BV'),
(53, 'BUCURESTI', 'B'),
(10, 'Buzau', 'BZ'),
(11, 'Calarasi', 'CL'),
(12, 'Caras-Severin', 'CS'),
(13, 'Cluj', 'CJ'),
(14, 'Constanta', 'CT'),
(15, 'Covasna', 'CV'),
(16, 'Dambovita', 'DB'),
(17, 'Dolj', 'DJ'),
(18, 'Galati', 'GL'),
(19, 'Giurgiu', 'GR'),
(20, 'Gorj', 'GJ'),
(21, 'Harghita', 'HR'),
(22, 'Hunedoara', 'HD'),
(23, 'Ialomita', 'IL'),
(24, 'Iasi', 'IS'),
(25, 'Ilfov', 'IF'),
(26, 'Maramures', 'MM'),
(27, 'Mehedinti', 'MH'),
(28, 'Mures', 'MS'),
(29, 'Neamt', 'NT'),
(30, 'Olt', 'OT'),
(31, 'Prahova', 'PH'),
(32, 'Salaj', 'SJ'),
(33, 'Satu Mare', 'SM'),
(34, 'Sibiu', 'SB'),
(35, 'Suceava', 'SV'),
(36, 'Teleorman', 'TR'),
(37, 'Timis', 'TM'),
(38, 'Tulcea', 'TL'),
(39, 'Valcea', 'VL'),
(40, 'Vaslui', 'VS'),
(41, 'Vrancea', 'VN');

--
-- Dumping data for table `nom_localitati`
--

INSERT INTO `nom_localitati` (`id`, `parinte`, `nume`, `oras`, `resedinta`)
VALUES
(1, 2, 'Arad', 1, 1),
(2, 2, 'Chisineu Cris', 1, 0),
(3, 2, 'Curtici', 1, 0),
(4, 2, 'Ineu', 1, 0),
(5, 2, 'Lipova', 1, 0),
(6, 2, 'Nadlac', 1, 0),
(7, 2, 'Pancota', 1, 0),
(8, 2, 'Pecica', 1, 0),
```

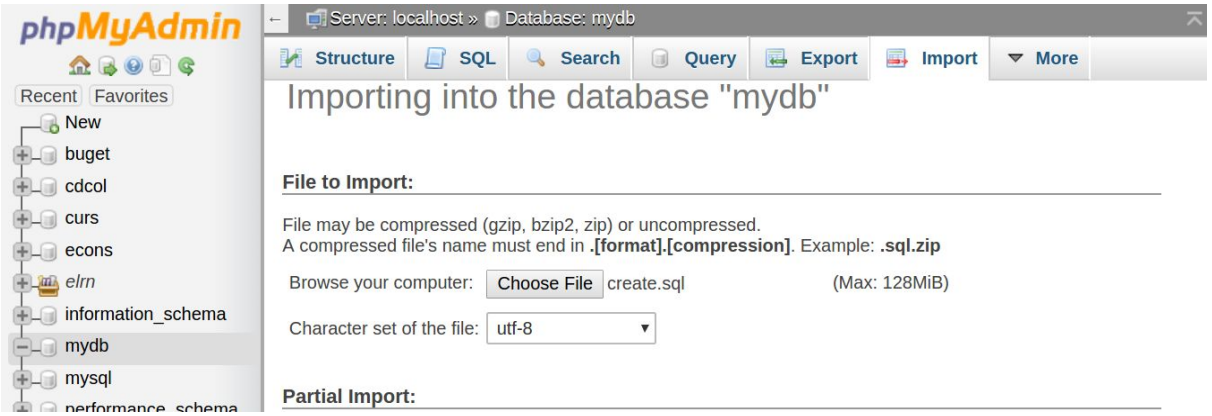
```
(9, 2, 'Santana', 1, 0),
(10, 2, 'Sebis', 1, 0),
(11, 2, 'Almas', 0, 0),
(12, 2, 'Apateu', 0, 0),
.....
```

Cum importăm baza de date?

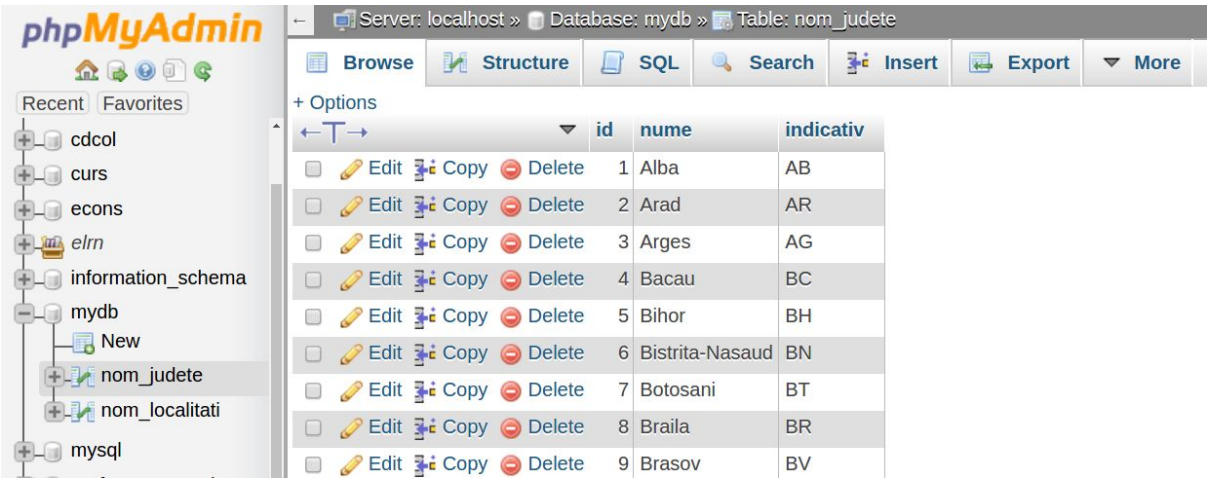
1. Accesăm *phpMyAdmin*, accesăm tab-ul *Databases* şi creăm o bază de date nouă



2. Selectăm baza de date, accesăm tab-ul *Import* , încărcăm fişierul *.sql* şi dăm click pe butonul *Go.*



Pentru a vedea datele, selectăm baza de date, apoi tabelul dorit şi accesăm tab-ul *Browse*.



Având construită baza de date, pasul următor este să facem conexiunea cu aceasta prin intermediul codului PHP.

2.5.2 Conexiune cu baza de date

Conexiunea cu baza de date o vom face în fişierul config/db.php folosind extensia PDO (PHP Data Object).

De ce folosim PDO? Pentru avantajele pe care le oferă faţă de extensiile *mysql*, respectiv *mysqli*:

- **Portabilitate.** Lucrează cu mai multe SGBD-uri (Sistem de Gestiune a Bazelor de Date): MySQL, PostgreSQL, SQLite, Oracle, Microsoft SQL Server, etc.
- **Flexibilitate.** Dacă am folosit PDO într-un proiect şi dorim să schimbăm SGBD-ul, putem face asta schimbând o singură linie de cod.
- **Viteza.** PDO este o librărie scrisă in C/C++, limbaj compilat, nu interpretat precum PHP.

Vom folosi PDO şi pe parcurs, ori de câte ori vom avea de interogat baza de date.

Dacă avem nevoie de mai multe detalii, accesăm manualul PHP - http://php.net/manual/en/class.pdo.php.

PHP. Fisierul *db.php*

```
<?php
// date de acces la baza de date
```

```php
$host = "localhost";
$db_name = "elrn_nom_judete";
$username = "root";
$password = "";

// incearca conexiunea la baza de date
try {
	$db = new PDO("mysql:host={$host};dbname={$db_name}", $username,
	$password);
} catch(PDOException $exception){// arata eroarea in caz de esec
	echo "Eroare la conectare: " . $exception->getMessage();
}
?>
```

Observaţii asupra codului:

Daca avem date de acces la baza de date diferite faţă de cele din exemplu, adaptăm fişierul *db.php.*

2.5.3 Structura fişierului index.php

Fişierul index.php este fişierul principal al aplicaţiei. Aici vom descrie structura (limbaj HTML) şi funcţionalitatea aplicaţiei (limbaj PHP).

Încă de la început vom încerca, pe cât posibil, să respectăm principiul separării codului responsabil de structură, formă, respectiv funcţionalitate.

Pentru a descrie structura aplicaţiei, vom folosi cod HTML. Pentru preluarea şi prelucrarea datelor, avem nevoie de limbaj de programare, deci vom folosi limbajul PHP. La realizarea unei forme elegante, ne va ajuta biblioteca *materialize.css.* Pentru stil personalizat, vom scrie cod CSS în fişierul *style.css.*

Astfel, structura fişierului *index.php* va fi:

Index.php - structura iniţială

```php
<!DOCTYPE HTML>
<html>
<head>
	<title>Nomenclator Judete</title>
	<!--Import Google Icon Font-->
	<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
	rel="stylesheet">
	<!--Includ materialize.css-->
	<link type="text/css" rel="stylesheet" href="css/materialize.css" />
	<!--Includ custom css-->
	<link type="text/css" rel="stylesheet" href="css/style.css" />
	<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
	<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"
	/>
```

```
</head>
<body>
<div class="container">
	<h1 class="header center">Nomenclator Judete</h1>
	<div class="row" id="judete">
		<!-- Judetele -->
	</div>
	<div class="row" id="localitati">
		<!-- Localitatile asociate judetului selectat-->
	</div>
</div> <!-- end .container -->

<!--Scripturi JS-->
<!--Import jQuery inainte de materialize.js-->
<script type="text/javascript" src="js/jquery-3.2.1.min.js"></script>
<script type="text/javascript" src="js/materialize.js"></script>
</body>
</html>
```

Acum ne vom ocupa de partea de conţinut a paginii. Ne propunem:

1. să selectăm toate judeţele din baza de date şi să le vizualizăm în browser într-o listă de selecţie.
2. Când selectăm un judeţ, la click pe un buton, să aducem din baza de date toate localităţile asociate judeţului şi să le vizualizăm într-un tabel HTML.

Pentru că dorim ca rezultatele să apară repede şi să ajungem pas cu pas la ceea ce ne propunem, vom introduce o etapa auxiliară: selectarea tuturor localităţilor din baza de date şi vizualizarea acestora într-un tabel HTML.

2.5.4 HTML. Tabel cu toate localităţile din baza de date

Ne propunem să selectăm localităţile din baza de date şi să le vizualizăm în browser într-un tabel HTML, ca în imaginea următoare.

Nomenclator Judete

Judet	Localitate
Alba	Abrud
Alba	Aiud
Alba	Alba Iulia
Alba	Baia de Aries
Alba	Blaj
Alba	Campeni
Alba	Cugir

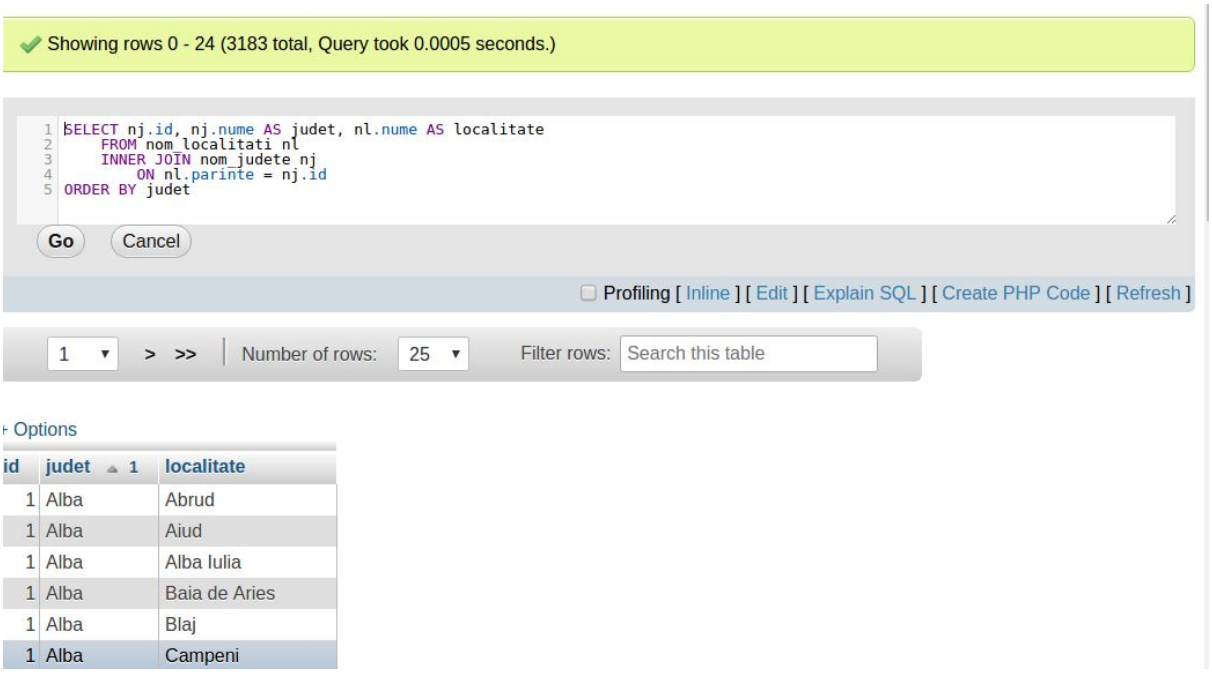
Prima etapă este să creăm structura tabelului folosind limbajul HTML. Pentru stil, apelăm la clasele predefinite în *materialize.css.* De asemenea, am asociat *id-uri* casetelor principale pentru a face mai uşor referinţă la acestea.

HTML. Tabelul pentru vizualizarea localităţilor

```
<div class="row"  id="localitati">
      <div class="col s12 center">
            <table class="bordered striped">
                  <thead><tr><th>Judet</th><th>Localitate</th></tr></thead>
                  <tbody  id="date">
                  <!-- Aici actualizez cu datele aduse din baza de date -->
                  </tbody>
            </table>
      </div>
</div>
```

Localităţile pe care le vrem în tabel sunt memorate în baza de date, aşadar, pasul următor este să facem un SELECT în baza de date şi să generăm automat liniile tabelului HTML folosind cod PHP.

Recomandare: testăm mai întâi SELECT-ul în PHPMyAdmin. Astfel evităm să avem erori de sintaxă şi să consumăm timp cu depanarea codului.



Având convingerea că SELECT-ul generează rezultatul aşteptat, putem scrie codul PHP care va realiza următoarele:

1. Conexiunea cu baza de date
2. Executarea SELECT-ului care extrage din baza de date judeţele şi localităţile asociate (un JOIN pe două tabele)
3. Generarea automată a unei linii de tabel HTML pentru fiecare înregistrare extrasă din baza de date

PHP. SELECT în baza de date. Generarea liniilor de date din tabelul HTML

```
<tbody  id="date">
<!-- Aici actualizez cu datele aduse din baza de date -->
<?php
include "config/db.php"; // conexiunea cu baza de date
$sql = "SELECT nl.id, nl.nume AS localitate, nj.nume AS judet
             FROM nom_localitati nl
                    INNER JOIN nom_judete nj
                    ON nl.parinte = nj.id
             ORDER BY judet";
$q = $db->prepare($sql);
$q->execute();
// determin numarul de linii returnate
$num = $q->rowCount();
if($num > 0){// daca am cel putin o inregistrare
while ($row = $q->fetch(PDO::FETCH_ASSOC)){
       // creez linie noua in tabel pentru fiecare inregistrare
       echo
                    '<tr>
```

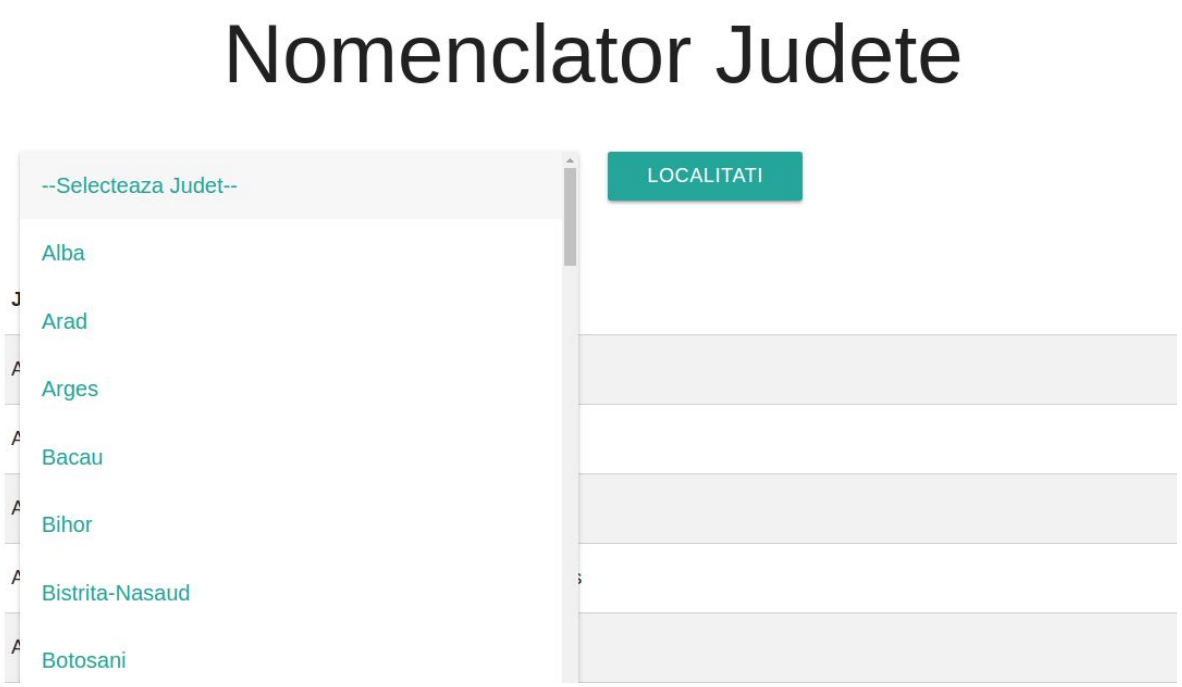
```
                        <td>'.$row['judet'].'</td>
                        <td>'.$row['localitate'].'</td>
                    </tr>';
        }// end while
}// end if
?>
</tbody>
```

Aplică noţiunile învăţate:

1. Adaugă codul de mai sus în fişierul *index.php*. Salvează şi deschide în browser.
2. Modifică codul astfel încât, în tabel, să apară şi coloana cu id-ul judeţului.
3. Pentru un judeţ, sunt afişate mai întâi oraşele, apoi celelalte localităţi. Modifică codul astfel încât, la nivel de judeţ, localităţile să fie afişate în ordine alfabetică.

2.5.5 Lista de selecţie pentru judeţe şi butonul de tip submit

Acum vom extrage judeţele din baza de date şi vom crea o listă de selecţie HTML. Butonul de tip submit ne va permite să transmitem id-ul judeţului selectat către codul PHP, astfel încât, în tabelul HTML, să fie vizibile doar localităţile asociate judeţului.



Etapele pe care le avem de parcurs vor fi:

1. Crearea formularului HTML care va conţine un element de tip *select* şi un buton de tip *submit;* datele din formular vor fi transmise către PHP prin variabila globală *$_POST*
2. Crearea comenzii SELECT pentru extragerea judeţelor din baza de date
3. Codul PHP care va executa comanda SELECT şi va genera în lista de selecţie câte un element <*option*> pentru fiecare înregistrare generată de SELECT.
4. Actualizarea comenzii SELECT care extrage localităţile din baza de date astfel încât să returneze doar localităţile asociate judeţului selectat.

2.5.6 HTML şi JQuery. Formularul pentru lista de selecţie

HTML şi JQuery. Formularul pentru lista de selecţie

```
<div class="row"  id="judete">
	<form class="col s12" action="" method="post">
		<div class="input-field col s6" id="div_judet">
			<select id="judet">
				<option value="">--Selecteaza Judet--</option>
				<!--Aici actualizez cu datele aduse din baza de
				date-->
			</select>
			<div class="input-field col s6">
				<button class="btn waves-effect waves-light"
				type="submit">Localitati</button>
			</div>
		</div>
	</form>
</div>
...
<!--la finalul documentului, dupa includerea bibliotecilor .js -->
<script type="text/javascript">
	$(document).ready(function(){
		$('select').material_select();//initializez elementul select
	});
</script>
```

Observaţii asupra codului:

Dacă testăm în browser fără codul JQuery de la final, lista de selecţie nu va fi vizibilă;

Explicaţia: *Materialize* transformă elemntul <*select*> în <*ul*> iar elementele <*option*> în elemente <*li*> cu scopul de a oferi posibilităţi de stilizare mai facile; dar

poate face asta prin intermediul funcţiei/metodei *material_select()* din biblioteca *materialize.js.*

Soluţii:

1. adăugăm clasa *browser-default* elementuli *<select>*: *<select id="judet" class="browser-default">*; astfel, elementul select nu va mai fi transformat, dar nici nu va mai beneficia de elemente de stil oferite de biblioteca *materialize;*
2. la încărcarea documentului, pentru fiecare element select, apelăm metoda *material_select()*:

```
<script type="text/javascript">
$(document).ready(function(){
      $('select').material_select();//initializez
elementul select
      });
</script>
```

2.5.7 MySQL. Selectarea judeţelor din baza de date

MySQL. Selectarea judeţelor din baza de date

```
SELECT id, nume AS judet FROM nom_judete
ORDER BY judet;
```

Observaţii asupra codului:

1. Cu ajutorul cuvântului cheie *AS*, am asociat alias-ul *judet* câmpului nume. Astfel, putem referi acest câmp şi prin intermediul alias-ului.
2. Apelăm la alias doar dacă avem avantaje de pe urma acestuia: ne oferă mai multă claritate şi/sau ne permite să diferenţiem mai uşor câmpuri cu denumiri similare.

2.5.8 PHP. Generarea elementelor <option> pentru judeţe

PHP. Generarea elementelor *<option> pentru judeţe*

```
<select name="judet" id="judet">
      <option value="">--Selecteaza Judet--</option>
      <!--Aici actualizez cu datele aduse din baza de date-->
      <?php
            $sql = "SELECT id, nume AS judet FROM nom_judete
                        ORDER BY judet";
            $q = $db->prepare($sql);
```

```
            $q->execute();
            // determin numarul de linii returnate
            $num = $q->rowCount();
            if($num > 0){// daca am cel putin o inregistrare
                  while ($row = $q->fetch(PDO::FETCH_ASSOC)){
                        // creez optiune noua pentru fiecare inregistrare
                        echo
                              '<option value="'.$row['id'].'">'
                                    .$row['judet'].
                              '</option>';
                  }// end while
            }// end if
      ?>
</select>
```

Observaţii asupra codului:

Etapele de lucru sunt aceleaşi pe care le-am parcurs şi la generarea tabelului de localităţi.

Aplică noţiunile învăţate:

1. Actualizează fişierul *index.php.* Salvează şi deschide în browser.
2. Lista ta de selecţie conţine toate judeţele din baza de date?

2.5.9 MySQL şi PHP. Localităţile asociate judeţului selectat

MySQL şi PHP. Localităţile asociate judeţului selectat

```
Anterior:
$sql = "SELECT nl.id, nl.nume AS localitate, nj.nume AS judet
            FROM nom_localitati nl
                  INNER JOIN nom_judete nj
                  ON nl.parinte = nj.id
            ORDER BY judet";
$q = $db->prepare($sql);
$q->execute();

Acum:
$sql = "SELECT nl.id, nl.nume AS localitate, nj.nume AS judet
            FROM nom_localitati nl
            INNER JOIN nom_judete nj
                  ON nl.parinte = nj.id";
if(isset($_POST['judet']) && $_POST['judet']!==""){
      $sql .= " WHERE nj.id =".$_POST['judet'];
}
$sql .= " ORDER BY judet";
$q = $db->prepare($sql);
```

```
$q->execute();
```

Observaţii asupra codului:

1. Modificarea codului o vom face în blocul *<tbody id="date">*
2. Logica codului şi funcţionalitatea aşteptată:
    a. Selectăm un judeţ;
    b. Facem click pe butonul *Localităţi* şi id-ul judeţului va fi memorat în variabila globală *$_POST['judet']*; dacă selecţia a rămas pe *--Selectează judeţ--, $_POST['judet'] va fi* `"";`
    c. Astfel, în comanda SELECT vom avea o clauză WHERE doar dacă există *$_POST['judet']* şi valoarea sa este diferită de `"";`
    d. În tabel vizualizăm doar localităţile din judeţul selectat; dacă selecţia a rămas pe *--Selectează judeţ--*, vizualizăm toate localităţile din baza de date
3. De ce *$_POST['judet']*? Pentru că elementului *<select>* i-am asociat atributul *name="judet"*: *<select name="judet" id="judet">*; dacă am fi avut *name="id_judet",* ar fi rezultat *$_POST['id_judet']*
4. De ce *$_POST['judet']* va avea drept valoare id-ul judeţului? Pentru că *$_POST['judet']* va fi egal cu valoarea atributului *value* asociat elementului *<option>;* elementul *<option>* asociat judeţului are atributul *value* iniţializat cu id-ul judeţului

```
<select name="judet" id="judet" class="text-red initialized" data-select-id="f11bba69-204c-0cf1-c9dc-63b81eaaa707">
  <option value>--Selecteaza Judet--</option>
  <!--Aici actualizez cu datele aduse din baza de date-->
  <option value="1">Alba</option>
  <option value="2">Arad</option>
  <option value="3">Arges</option>
  <option value="4">Bacau</option>
  <option value="5">Bihor</option>
  <option value="6">Bistrita-Nasaud</option>
  <option value="7">Botosani</option>
```

2.6 Aplicaţii propuse

Vizualizare date. Să se adauge aplicaţiei descrise în acest tutorial următoarea funcţionalitate: la selecţia unui judeţ, tabelul de date va mai conţine încă două coloane: *Oras (Da/Nu), Resedinta (Da/Nu)*.

Judet	Localitate	Oras (Da/Nu)	Resedinta Judet (Da/Nu)
Brasov	Codlea	DA	NU

Indicaţie: Verifică structura bazei de date pentru a adapta comanda SELECT.

2.7 Concluzii

În acest moment avem o aplicaţie funcţională care se poate adapta uşor pentru alte situaţii similare: categorii şi produse, edituri şi cărţi, clase şi elevi, facultăţi şi domenii de licenţă, etc.

3 NOMENCLATOR JUDEŢE ŞI LOCALITĂŢI. AJAX

Deseori, în aplicaţii reale, apare necesitatea ca o listă de selecţie să se actualizeze automat în funcţie de ceea ce s-a selctat într-o altă listă de selecţie. Sunt aşa numitele liste de selecţie dependente.

Una dintre modalităţile prin care putem rezolva o astfel de situaţie este să folosim AJAX.

AJAX este acronim de la Asynchronous JavaScript and XML.

Folosind AJAX, evităm reîncărcarea completă a paginii la fiecare click pe butoanele din conţinutul acesteia. Astfel, aplicaţia va fi mai rapidă, utilizatorul ceva mai fericit.

Ajax este de fapt un grup de tehnologii. Poate include HTML, CSS, JavaScript/JQuery, scripturi PHP.

Pentru a exemplifica lucrul cu AJAX, rămânem la aceeaşi situaţie cu care ne-am obişnuit deja: judeţe şi localităţi. Vom avea aceeaşi bază de date şi aceeaşi structură ca şi în aplicaţia anterioară.

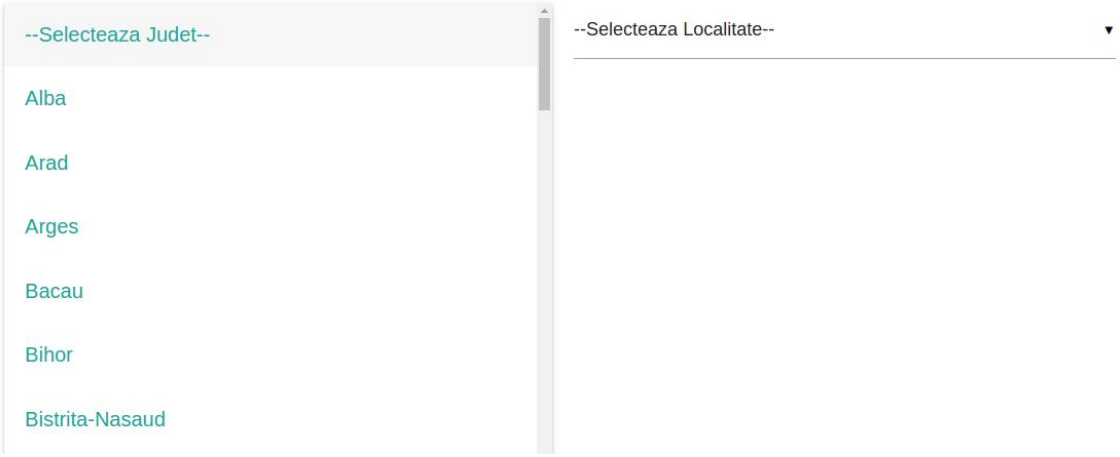
3.1 Obiective

Parcurgând acest tutorial, în plus faţă de aplicaţia anterioară, vom putea face următoarele:

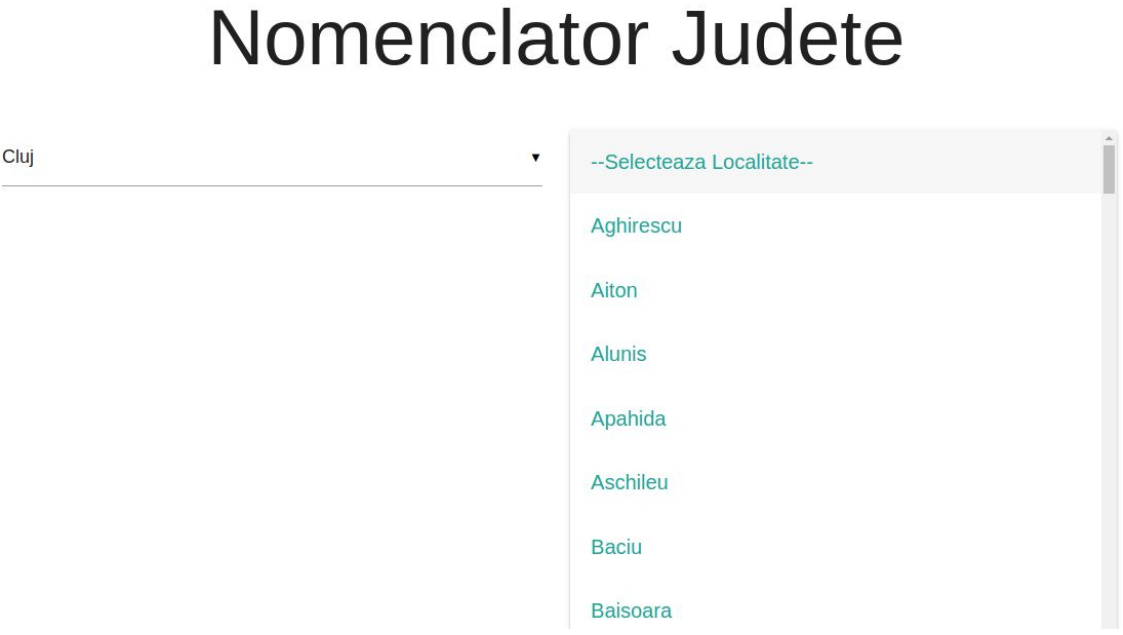
- Să utilizăm funcţie JQuery, *$.ajax()* pentru:
    - Transmiterea de date prin AJAX
    - Actualizarea codului HTML în funcţie de rezultatele unor interogări ale bazei de date
- Să facem debug folosind extensia AJAX Debugger din Google Chrome.

3.2 Rezultatul aşteptat

Iniţial, utilizatorul are posibilitatea de a selecta un judeţ din lista de selecţie care conţine toate judeţele din România, ordonate alfabetic.



La selecţia unui judeţ, lista de selecţie asociată localităţilor se va actualiza automat şi va conţine toate localităţile din judeţul respectiv, ordonate alfabetic.



Dacă selecţia rămâne pe *--Selecteaza Judet--*, lista de selecţie asociată localităţilor va conţine doar mesajul *“Selecteaza mai intai Judetul”.*

3.3 Baza de date

Baza de date este cea utilizată în aplicaţia anterioară. Deci şi fişierul de configuraţie *config/db.php* va rămâne neschimbat.

3.4 Structura aplicaţiei

3.5 Etape de lucru

Vom porni de la aplicaţia anterioară. În fişierul *index.php* vom descrie mai întâi structura paginii folosind cod HTML.

Folosind cod JQuery, în fişierul *index.php* vom transmite prin AJAX datele necesare interogării bazei de date.

În fişierul *ajax.php* vom interoga baza de date şi vom face prelucrările necesare. Rezultatatele se vor întoarce către *index.php* actualizând astfel codul HTML.

3.5.1 Conexiune cu baza de date

Baza de date este aceeaşi ca şi la aplicaţia anterioară, deci şi fişierul de conexiune la baza de date va fi acelaşi.

PHP. Fisierul *db.php*

```php
<?php
// date de acces la baza de date
$host = "localhost";
$db_name = "elrn_nom_judete";
$username = "root";
$password = "";

// incearca conexiunea la baza de date
try {
	$db = new PDO("mysql:host={$host};dbname={$db_name}", $username,
	$password);
} catch(PDOException $exception){// arata eroarea in caz de esec
	echo "Eroare la conectare: " . $exception->getMessage();
}
?>
```

3.5.3 Structura fişierului index.php

Ne mai amintim de structura iniţială a fişieruluil *index.php* din aplicaţia anterioară? Ne întoarcem la ea şi o reutilizăm (o găsim într-un capitol anterior).

În div-ul *.container*, descriem structura celor două liste de selecţie: una pentru judeţe, cealaltă pentru localităţi.

HTML. Listele de selecţie

```html
<div class="container">
	<h1 class="header center">Nomenclator Judete</h1>
	<div class="row">
		<div class="input-field col s6" id="div-judet">
			<select name="judet" id="judet">
				<option value="">--Selecteaza Judet--</option>
				<!--Aici actualizez cu datele aduse din baza de
				date-->
			</select>
		</div>
```

```
            <div class="input-field col s6" id="div-localitate">
                  <select name="localitate" id="localitate">
                        <option value="">--Selecteaza Localitate--</option>
                        <!--Aici actualizez cu datele transmise prin AJAX-->
                  </select>
            </div>
      </div>
</div><!-- end .container -->
```

Observaţii asupra codului:

Pentru că nu mai transmitem datele prin *$_POST*, nu vom mai avea nevoie de buton; de aceea nu am mai folosit nici formularul.

3.5.4 PHP. Lista de selecţie pentru judeţe

Pentru a nu complica prea mult de la început, lista de selecție pentru județe o vom genera prin cod PHP, ca și în aplicația anterioară.

PHP. Lista de selecţie pentru judeţe

```
<select name="judet" id="judet">
      <option value="">--Selecteaza Judet--</option>
      <!--Aici actualizez cu datele aduse din baza de date-->
      <?php
      include "config/db.php";
      $sql = "SELECT id, nume AS judet FROM nom_judete ORDER BY judet";
      $q = $db->prepare($sql);
      $q->execute();
      // determin numarul de linii returnate
      $num = $q->rowCount();
      if($num > 0){// daca am cel putin o inregistrare
            while ($row = $q->fetch(PDO::FETCH_ASSOC)){
                  // creez optiune noua pentru fiecare inregistrare
                  echo
                  '<option value="' .$row['id']. '">'
                        .$row['judet'].
                  '</option>';
            }// end while
      }// end if
      ?>
</select>
```

3.5.5 JQuery. Id-ului judeţului transmis prin AJAX

Creăm fișierul *ajax.php* asupra căruia vom reveni ulterior.

Ne întoarcem în *index.php* și vom scrie codul JQuery care ne va ajuta să transmitem id-ul județului selectat către *ajax.php*.

Zona rezervată codului JavaScript/JQuery este la finalul fişierului, în blocul *<script type="text/javascript">*.

Pentru a se executa codul imediat ce se încarcă pagina, îl vom scrie în interiorul funcţiei *$(document).ready(function(){})*.

```
    <!--Scripturi JS-->
    <!--Import jQuery inainte de materialize.js-->
    <script type="text/javascript" src="js/jquery-3.2.1.min.js"></script>
    <script type="text/javascript" src="js/materialize.js"></script>

    <script type="text/javascript">

    $(document).ready(function(){
        $('select').material_select();//initializez elementul select

        $('#judet').on('change',function(){
            // cand schimb optiunea din #judet, generez selectul #localitate asociat
        })// end $('#judet').on('change',function(){})
    });
    </script>
</body>
</html>
```

Pentru a transmite datele, vom folosi funcţia JQuery *$.ajax()*. Mai multe detalii putem găsi pe http://api.jquery.com/jquery.ajax/.

JQuery. Transmiterea id-ului judeţului prin AJAX

```
$('#judet').on('change',function(){
	var judetID =$(this).val();
	if(judetID){ // daca am selectat judet, transmit id-ul catre ajax.php
		$.ajax({
			type:'POST',
			url:'ajax.php',
			dataType: 'html',
			data: {
				action : 'localitati',
				ajax : 1,
				judet_id : judetID,
			},
			success:function(data){
				//console.log(data); // pentru debug
				$('#localitate').html(data);
				$('#localitate').material_select();// re-initializez
				material-select;
			}
		});// end ajax
	}else{
		$('#localitate').html('<option value="">Selecteaza mai intai
		Judetul</option>');
		$('#localitate').material_select();
```

```
    }
})
```

Observaţii asupra codului:

1. Dacă am schimbat selecţia în lista de judeţe, memorăm valoarea asociată în variabila JavaScript *judetID*:
    - `var judetID =$(this).val();`
2. *judetID* va memora id-ul unui judeţ sau valoarea “” dacă selecţia a rămas pe *--Selectează judet--*
3. Dacă în *judetID* avem un id de judeţ, îl transmitem către *ajax.php* prin variabila globală *$_POST*. Pentru a verifica mai uşor dacă datele au ajuns în *ajax.php*, transmitem şi o variabilă *action* si o variabilă “semafor” - *ajax : 1* (vor fi utile mai ales atunci când vom transmite mai multe date)*:*

```
type:'POST',
url:'ajax.php',
data: {
      action : 'localitati',
      ajax : 1,
      judet_id : judetID,
},
```

4. În caz de succes, codul HTML generat în *ajax.php* (ceea ce se transmite cu *echo*) se va adăuga elementului *<select name="localitate" id="localitate">*. Vom obţine astfel lista localităţilor asociate judeţului selectat

```
success:function(data){
      $('#localitate').html(data);
      $('#localitate').material_select();
}
```

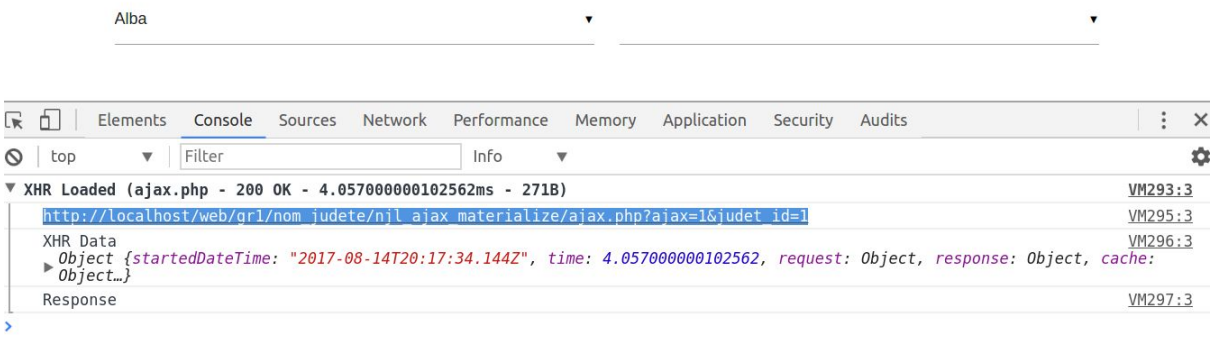
5. Pentru că lucrăm cu biblioteca *materialize*, este necesar să reiniţializăm elementul *select*; altfel nu se va actualiza:

```
$('#localitate').material_select();
```

6. Dacă în *judetID* nu avem un id de judeţ, transmitem mesajul “*Selecteaza mai intai Judetul”*:

```
else{
      $('#localitate').html('<option value="">Selecteaza mai
      intai Judetul</option>');
      $('#localitate').material_select();
}// end if
```

Pentru a trece mai departe, trebuie să verificăm dacă ceea ce am făcut până acum funcţionează.



În Google Chrome adăugăm extensia *AJAX Debugger* (More Tools → Extensions).

Deschidem fereastra *Inspect* (click dreapta → Inspect) şi selectăm tab-ul *Console.*

Selectăm primul judeţ din lista de selecţie şi vedem că se transmite către *ajax.php ajax=1&judet_id=1.* Deci datele se transmit corect către *ajax.php*.

3.5.6 PHP. Fişierul ajax.php

În fişierul *ajax.php* preluăm id-ul judeţului selectat şi generăm lista de selecţie cu localităţile asociate. Tot ceea ce se transmite cu *echo* este cod HTML care va actualiza elementul *#localitate*.

Fişierul *ajax.php*

```
<?php
include 'config/db.php'; // includ conexiunea la baza de date
if($_POST['ajax']){
$action = trim($_POST['action']);
switch($action){
	case 'localitati':
		$judet_id = trim($_POST['judet_id']);
		// selectez localitatile asociate judetului
		$sql = "SELECT id, nume FROM nom_localitati
			WHERE parinte = :judet_id
			ORDER BY nume";
		$q = $db->prepare($sql);
		$q->bindValue(':judet_id', $judet_id, PDO::PARAM_INT);
		$q->execute();
		$num = $q->rowCount();// determin numarul de linii returnate
		echo '<option value="">';
			echo "--Selecteaza Localitate--";
		echo '</option>';
```

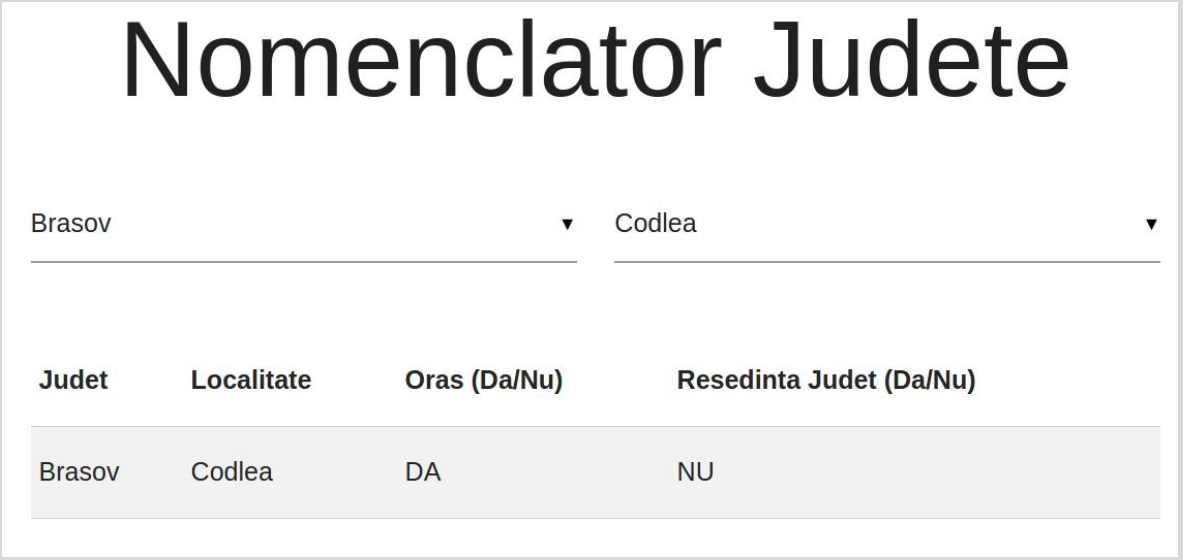
```
            if($num > 0){// daca am gasit cel putin o inregistrare
                  while ($row = $q->fetch(PDO::FETCH_ASSOC)){
                        echo '<option value="'.$row['id'].'">';
                              echo $row['nume'];
                        echo '</option>';
                  }
            }else{// daca nu am gasit inregistrari
                echo '<option value="">Judetul nu este valid</option>';
            }
      break;
}// end switch
}
?>
```

Observaţii asupra codului:

Etapele de lucru sunt aceleaşi pe care le-am parcurs şi la generarea listei de selecţie pentru judeţe.

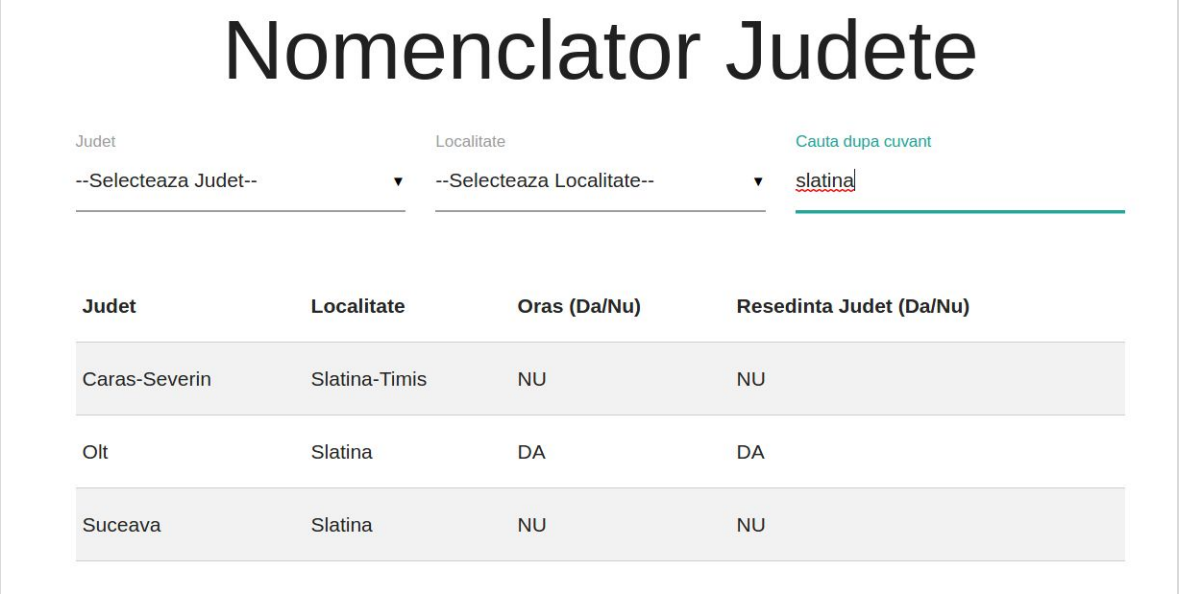
3.6 Aplicaţii propuse

1. **Vizualizare date.** Să se adauge aplicaţiei descrise în acest tutorial următoarea funcţionalitate: la selecţia unei localităţi se vor vizualiza într-un tabel HTML datele asociate acesteia: numele localităţii, judeţul, dacă este oraş, dacă este reşedinţă de judeţ.



2. **Căutare după cuvânt.** Se va adăuga un câmp de căutare care va permite căutarea după un cuvânt. După tastarea a cel puţin trei caractere, se vor

afişa într-un tabel cu structura descrisă mai sus, toate localităţile care se potrivesc cuvântului de căutare.



3.6.1 Soluţie aplicaţie 1:

Pasul 1: în *index.php* creăm structura tabelului de date

```
<div class="row"  id="localitati">
      <div class="col s12 center">
            <table class="bordered striped" id="date">
            <!-- Aici actualizez cu datele transmise prin AJAX -->
            </table>
      </div>
</div>
```

Pasul 2: JQuery. Transmitem id-ul localităţii prin AJAX

```
$('#localitate').on('change',function(){
      var localitateID =$(this).val();
      $.ajax({
            type:'POST',
            url:'ajax.php',
            dataType: 'html',
            data: {
                  action : 'date',
                  ajax : 1,
                  localitate_id : localitateID,
            },
            success:function(data){
```

```
                    $('#date').html(data);
                }
        });// end ajax
})// end $('#localitate').on('change',function(){})
```

Pasul 3: actualizăm fisierului *ajax.php*

```
case 'date':
        $localitate_id = trim($_POST['localitate_id']);
        // selectez localitatile asociate judetului
        $sql =
        "SELECT nl.id, nl.nume AS localitate, oras, resedinta, nj.nume AS judet
                FROM nom_localitati nl
                INNER JOIN nom_judete nj
                        ON nl.parinte = nj.id
                WHERE nl.id = :localitate_id
        ";
        $q = $db->prepare($sql);
        $q->bindValue(':localitate_id', $localitate_id, PDO::PARAM_INT);
        $q->execute();
        $num = $q->rowCount();// determin numarul de linii returnate
        echo '<thead>
                <tr>
                        <th>Judet</th>
                        <th>Localitate</th>
                        <th>Oras (Da/Nu)</th>
                        <th>Resedinta Judet (Da/Nu)</th>
                </tr>
                </thead>';
        if($num > 0){// daca am gasit cel putin o inregistrare
                echo '<tbody>';
                while ($row = $q->fetch(PDO::FETCH_ASSOC)){
                        $oras        = $row['oras'] == 1 ? 'DA' : 'NU';
                        $resedinta   = $row['resedinta'] == 1 ? 'DA' : 'NU';
                        echo '<tr>
                                        <td>' .$row['judet']. '</td>
                                        <td>' .$row['localitate']. '</td>
                                        <td>' .$oras. '</td>
                                        <td>' .$resedinta. '</td>
                                </tr>';
                }// end while
                echo '</tbody>';
        }// end if
        break;
```

3.6.2 Soluţie aplicaţie 2:

Pasul 1: în *index.php* creăm câmpul de căutare

```
<div class="input-field col s4" id="div-search">
	<input id="search" type="text" class="" placeholder="minim 3 caractere"/>
	<label for="search">Cauta dupa cuvant</label>
</div>
```

Pasul 2: JQuery. Transmitem cuvântul de căutare prin AJAX

Transmitem cuvântul de căutare după tastarea a cel puțin 3 caractere.

Variabila *action* va ramâne tot cu valoarea *date* pentru că, în urma transmiterii cuvântului de căutare, vom genera codul HTML pentru zona de date, ca și la selecția localității.

```
$('#search').keyup(function() {
	var word = $('#search').val();
	if (word.length > 2) {
		$('#date').html('');
		$.ajax({
			type: "POST",
			url: 'ajax.php',
			dataType: 'html',
			data: {
				ajax: 1,
				action:'date',
				word: word
			},
			success: function(data) {
				$('#date').html(data);
			}
		});// end ajax
	}// end if
});// end $('#search').keyup()
```

Pasul 3: actualizăm fisierului *ajax.php*

Vom interveni doar asupra comenzii SQL pentru a separa cele două situații mutual exclusive: transmiterea id-ului localității sau transmiterea cuvântului de căutare.

Anterior:

```
case 'date':
	$localitate_id = trim($_POST['localitate_id']);
	// selectez localitatile asociate judetului
	$sql =
	"SELECT nl.id, nl.nume AS localitate, oras, resedinta, nj.nume AS judet
		FROM nom_localitati nl
```

```
            INNER JOIN nom_judete nj
                  ON nl.parinte = nj.id
            WHERE nl.id = :localitate_id
      ";
      $q = $db->prepare($sql);
      $q->bindValue(':localitate_id', $localitate_id, PDO::PARAM_INT);
      $q->execute();
      ...
      break;
```

Acum:

```
case 'date':
      $localitate_id =
      isset($_POST['localitate_id']) ? trim($_POST['localitate_id']) : NULL;
      $word = isset($_POST['word']) ? trim($_POST['word']) : NULL;
      $sql =
      "SELECT nl.id, nl.nume AS localitate, oras, resedinta, nj.nume AS judet
      FROM nom_localitati nl
      INNER JOIN nom_judete nj
            ON nl.parinte = nj.id";
      if($localitate_id){
            $sql .= " WHERE nl.id = :localitate_id";
      }
      if($word){
            $sql .= " WHERE nl.nume LIKE '%" .$word. "%'";
      }
      $q = $db->prepare($sql);
      if($localitate_id){
            $q->bindValue(':localitate_id', $localitate_id, PDO::PARAM_INT);
      }
      $q->execute();
      ...
      break;
```

4 NOMENCLATOR COR. STRUCTURĂ ARBORESCENTĂ DE DATE

4.1 Obiective

Parcurgând acest tutorial vom putea face următoarele:

- Să creăm o bază de date MySQL pornind de la un fişier *.csv.*
- Să interogăm o bază de date în care datele sunt relaţionate ierarhic.
- Să utilizăm funcţie JQuery, *$.ajax()* pentru:
    - Transmiterea de date prin AJAX în format ***JSON***
    - Actualizarea codului HTML în funcţie de rezultatele unor interogări ale bazei de date
- Să creăm elemente dinamice în pagină folosind *matrialize.js*.
- Să facem debug folosind extensia AJAX Debugger din Google Chrome

4.2 Rezultatul aşteptat

Acest tutorial vine în completarea celor anterioare, nivelul de complexitate fiind ceva mai ridicat.

COR este Clasificarea Ocupaţiilor din România. Fiind nomenclatorul ocupaţiilor cu care se operează pe piaţa muncii, o aplicaţie web cu o astfel de bază de date poate fi utilă viitorilor angajaţi, actualilor angajaţi, angajatorilor, celor care operează în departamente de resurse umane dar şi celor din mediul educaţional (elevi, studenţi, cursanţi, profesori, traineri) care, cunoscând aceste detalii, pot orienta sau se pot orienta mai uşor în proiectarea unei cariere potrivite.

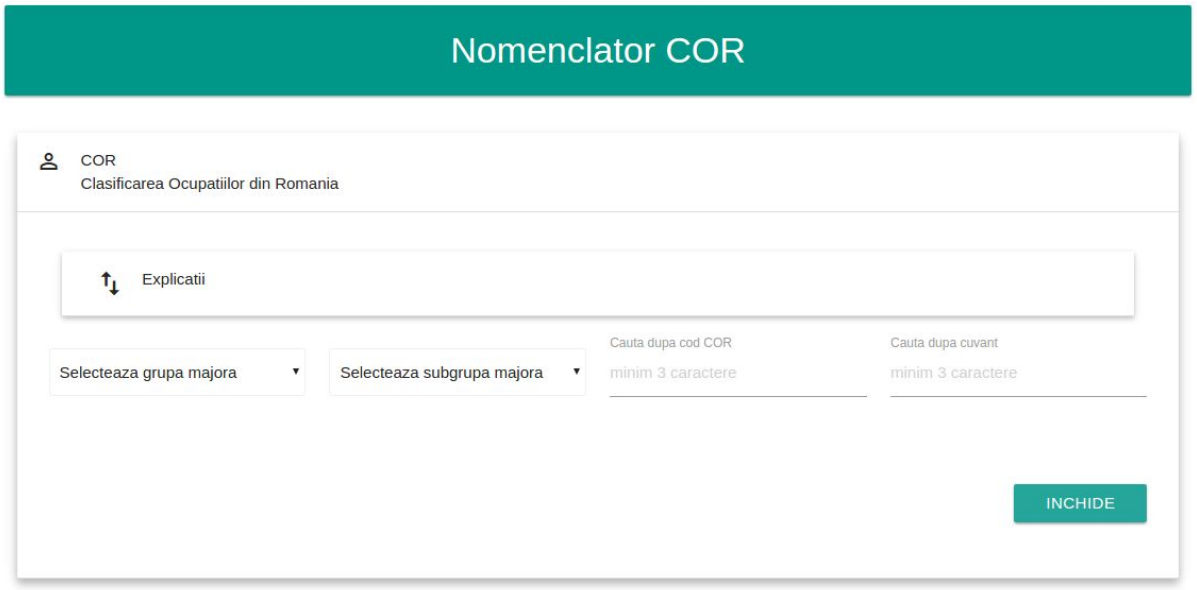
Codul COR este compus din 6 cifre. În structura acestuia regăsim următoarele:

- Grupa majoră reprezintă prima cifră din cod
- Subgrupa majoră reprezintă a doua cifră din cod
- Grupa minoră reprezintă a treia cifră din cod
- Grupa de bază reprezintă a patra cifră din cod

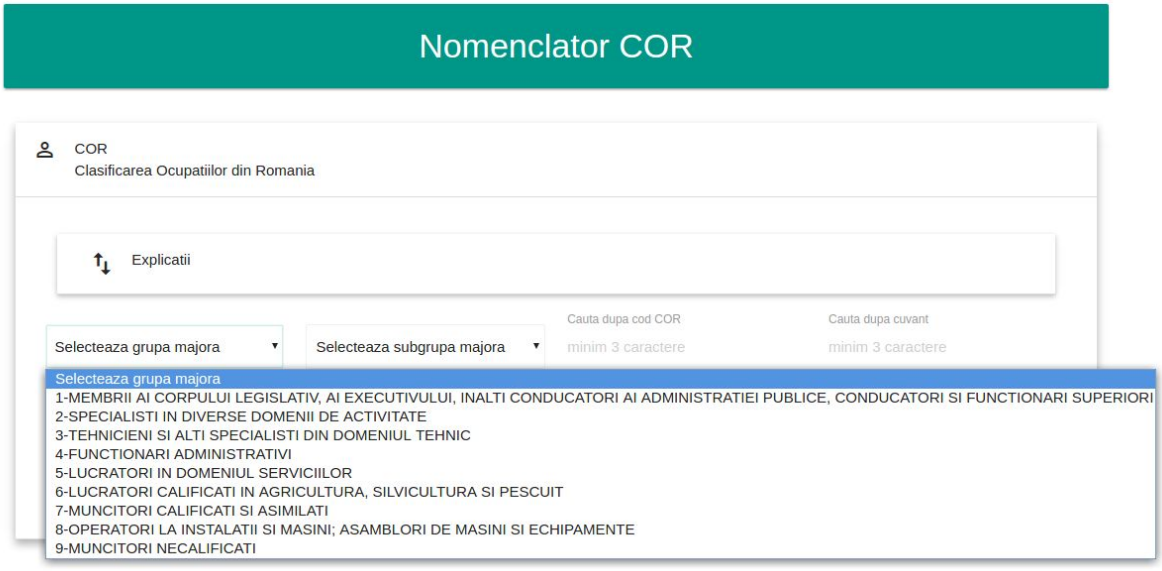
Astfel, rezultă o structură arborescentă pe cinci nivele, ultimul fiind cel cu codurile COR şi denumirile ocupaţiilor existente pe piaţa muncii.

Prin aplicaţia pe care o vom dezvolta în continuare, dorim să facilităm căutarea datelor în nomenclatorul COR în funcţie de nevoile existente în practica curentă:

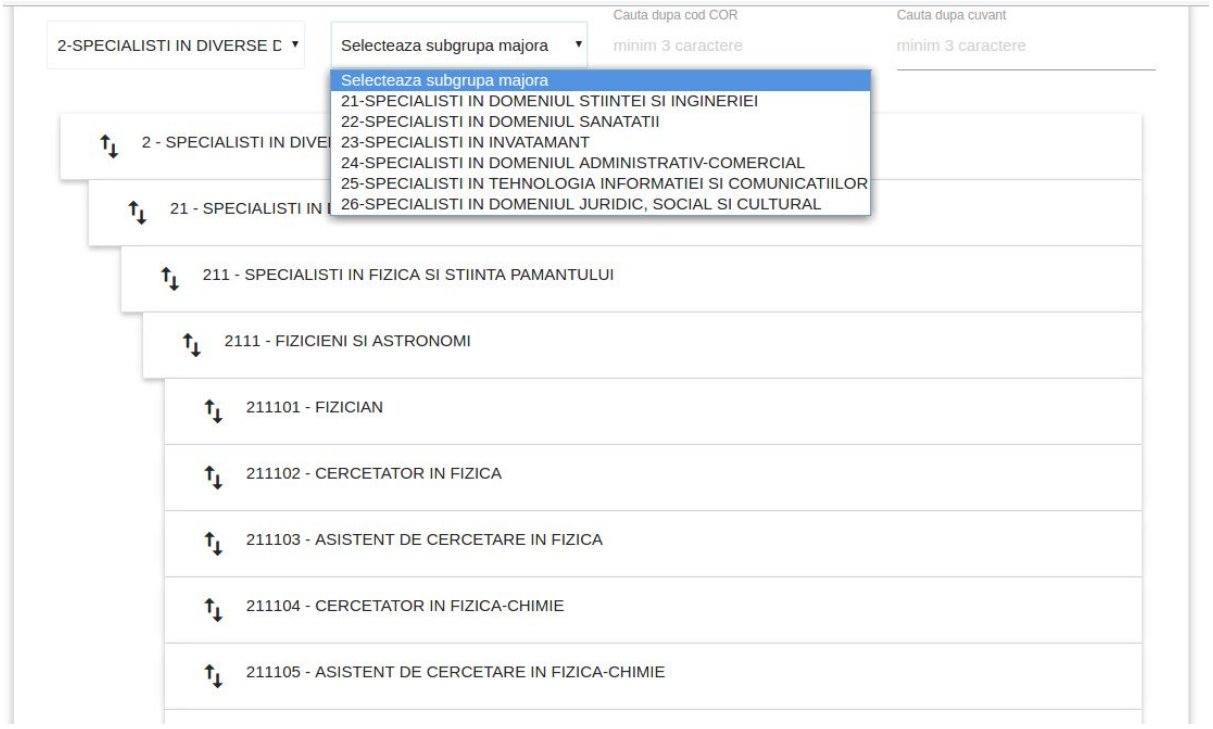
- Căutare după grupa majoră
- Căutare după subgrupa majoră
- Căutarea după codul COR (se impune tastarea a cel puţin trei caractere din cod). Căutarea se face doar pe ultimul nivel.
- Căutarea după un cuvânt de căutare (se impune tastarea a cel puţin trei caractere din cuvânt). Căutarea se face doar pe ultimul nivel, atât în denumirea ocupaţiei, cât şi în descrierea acesteia.



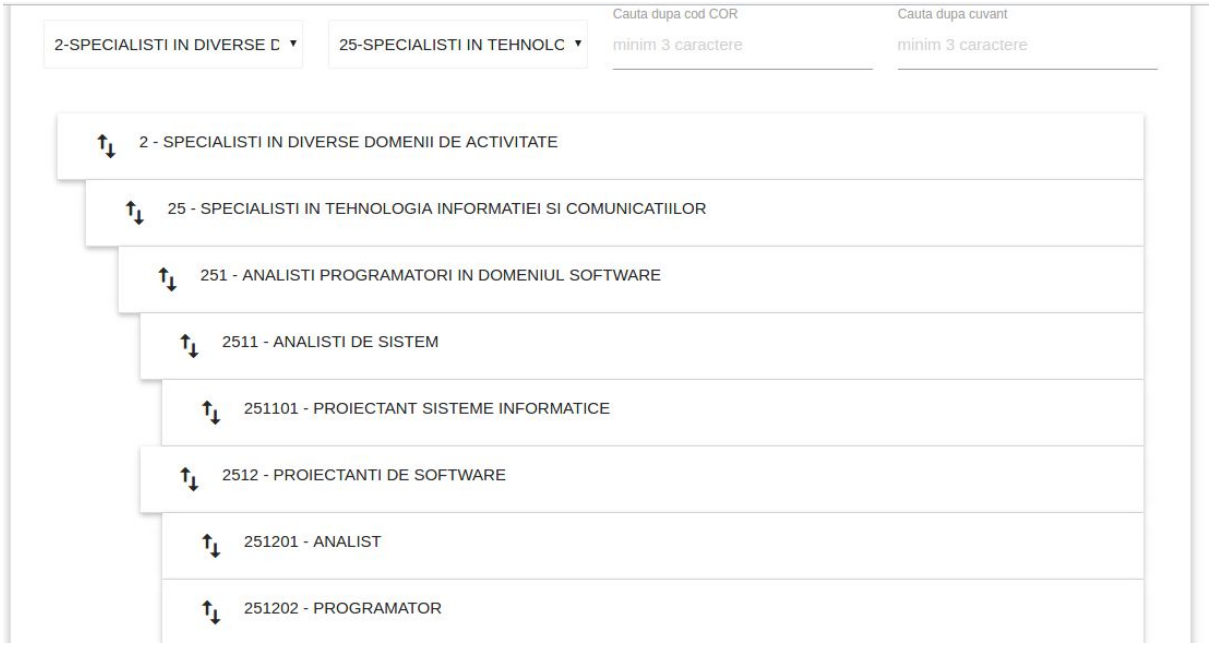
La deschiderea aplicaţiei, utilizatorul poate vedea descrierea nomenclatorului COR (descrierea apare la click pe caseta *Explicaţii)* sau poate face o căutare după unul dintre criteriile prezentate mai sus.



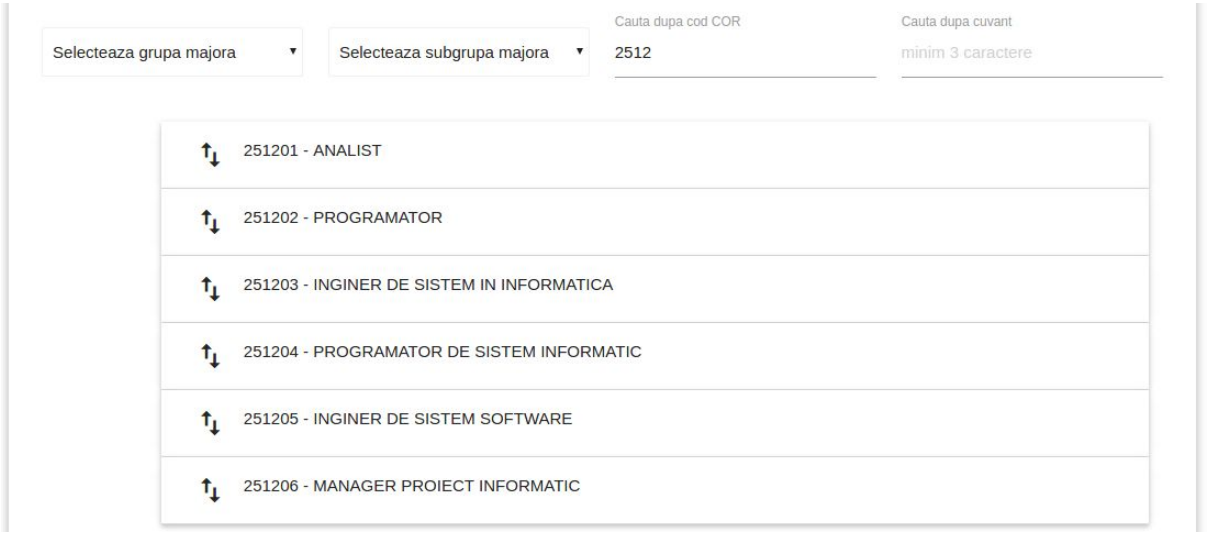
Prima listă de selecţie conţine cele nouă grupe majore.



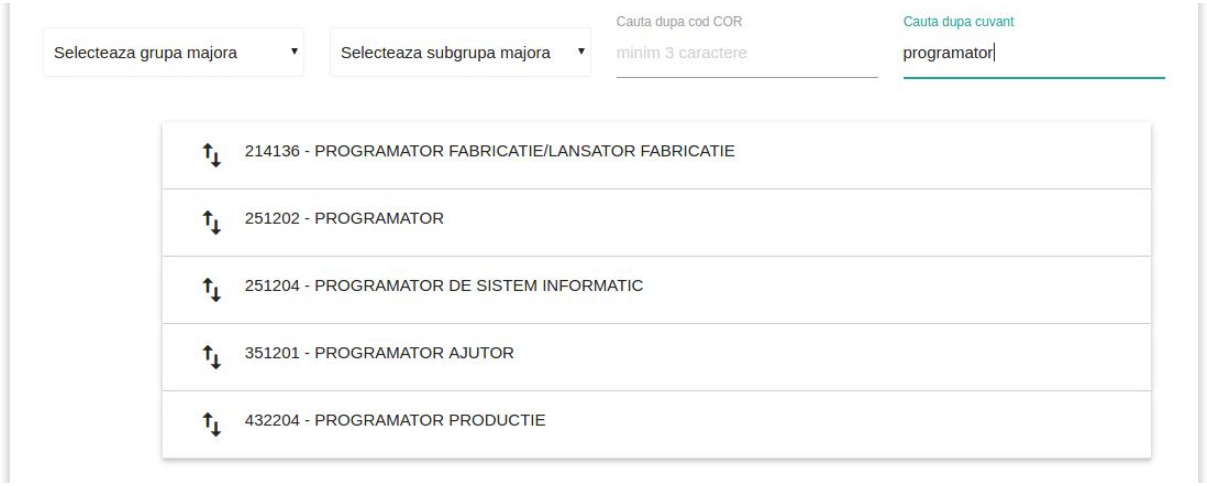
La selecţia unei grupe majore, se afişează toată arborescenţa acesteia (în imagine este vizibilă doar o parte a arborescenţei, volumul de date fiind foarte mare). De asemenea, lista de selecţie cu subgrupele majore se actualizează automat, conţinând numai subgrupele asociate grupei selectate.



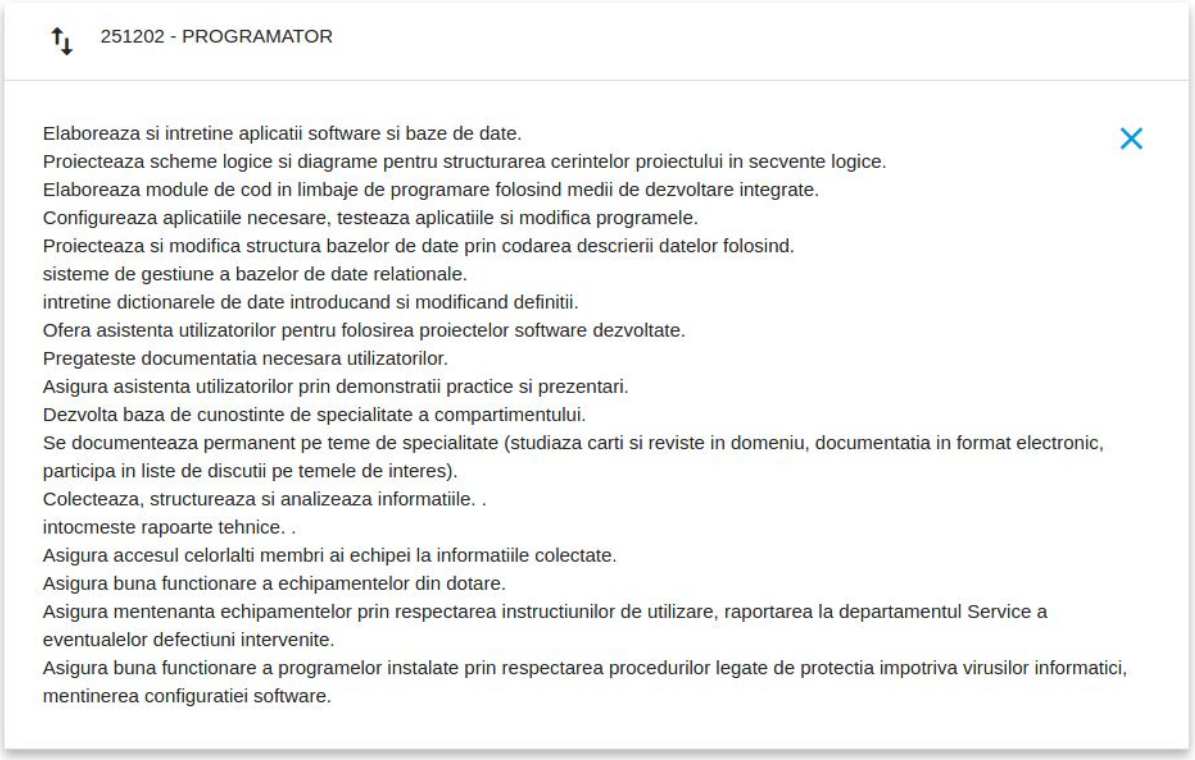
La selecţia succesivă a unei grupe majore, respectiv a unei subgrupe, se va afişa grupa, subgrupa şi numai arborescenţa subgrupei selectate.



La tastarea a cel puţin trei caractere dintr-un cod COR, se vor afişa toate ocupaţiile care conţin în codul COR secvenţa tastată.



La tastarea a cel puţin trei caractere din cuvântul de căutare, se vor afişa toate ocupaţiile care conţin în denumire secvenţa tastată.



Fiecare element al arborescenţei are asociată şi o descriere, aceasta devenind vizibilă la click pe elementul respectiv.

4.3 Baza de date

Baza de dat o vom crea importând fişierul *cor.csv* care se poate descărca folosind link-ul din anexa *Fişiere*.

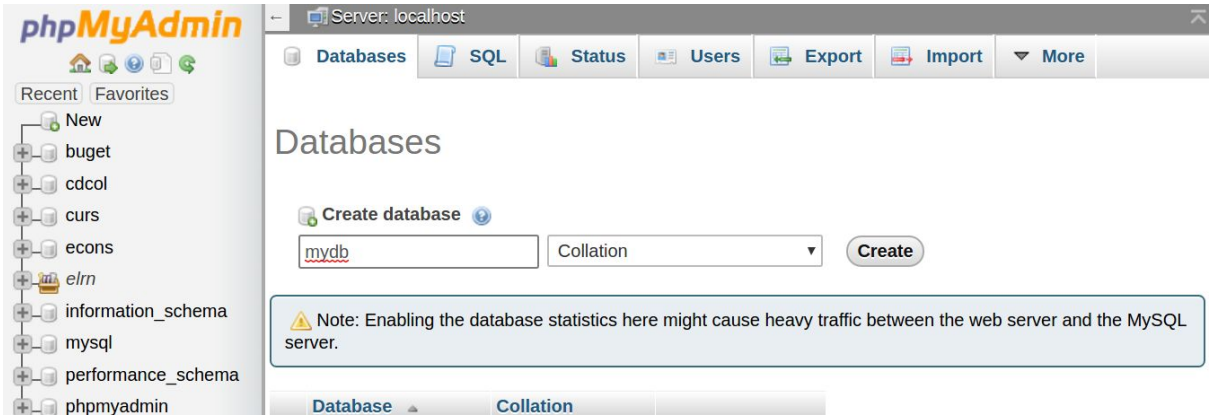
Sursa datelor este site-ul Ministerului Muncii (fiind în format *.pdf*, a necesitat prelucrări ulterioare):

http://www.mmuncii.ro/j33/index.php/ro/2014-domenii/munca/c-o-r

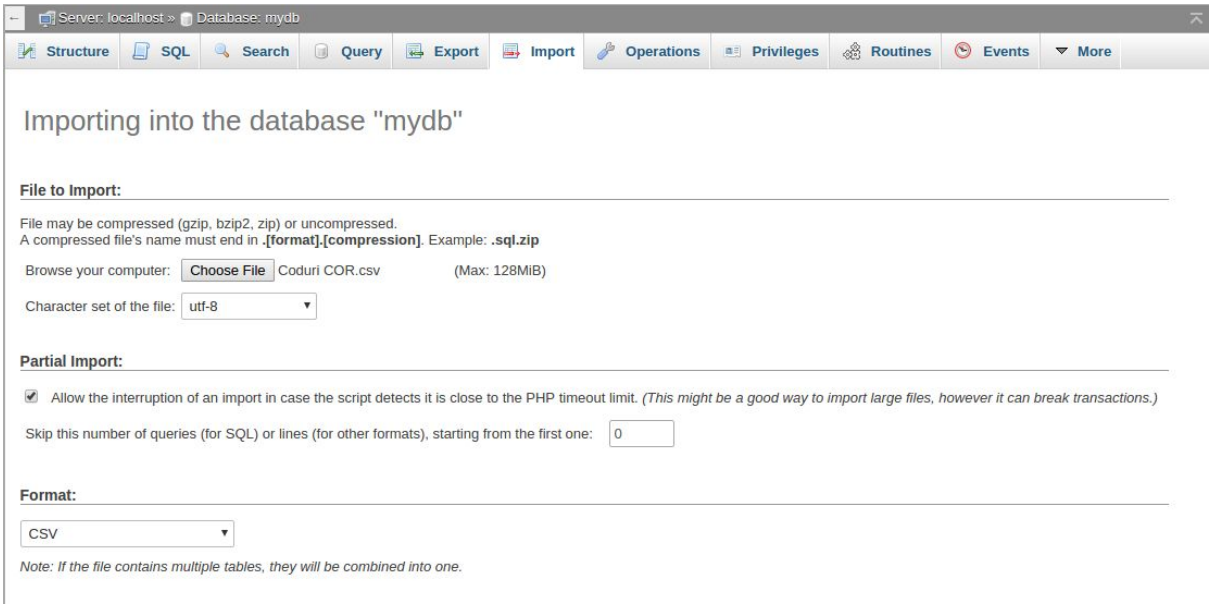
Deşi din analiză ar rezulta entităţile *grupă majoră, subgrupă majoră, grupă minoră, grupă de bază, ocupaţie,* baza de date rezultată va avea un singur tabel. Putem face o astfel de proiectare pentru că entităţile sunt identice ca structură (toate au un *cod*, o *denumire* şi o *descriere*) iar relaţia arborescentă se poate deduce din modul în care este construit codul.

Etapele prin care creez o bază de date pornind de la un fişier *.csv*:

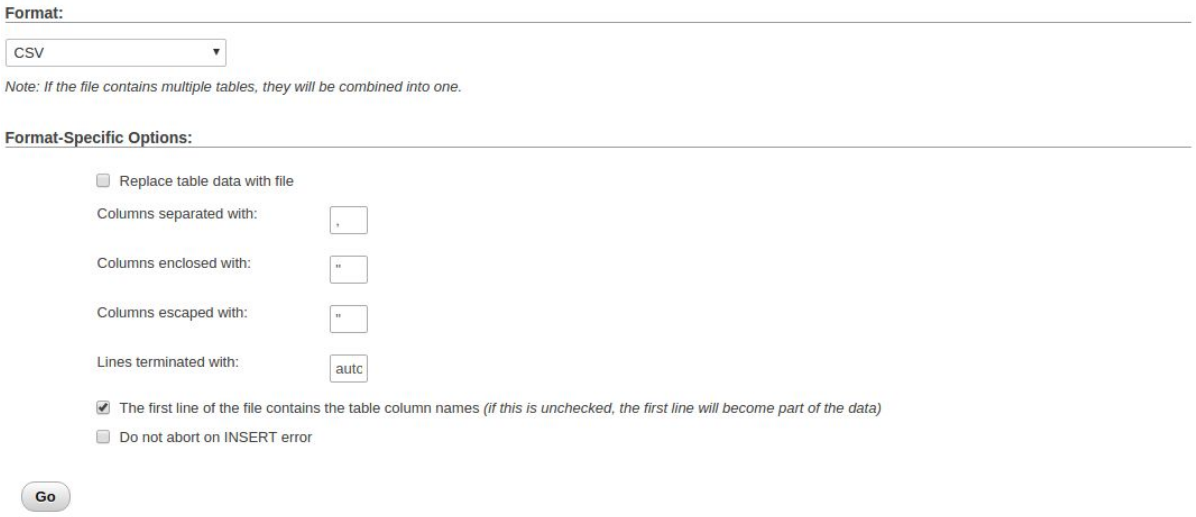
1. Accesăm *phpMyAdmin*, accesăm tab-ul *Databases* şi creăm o bază de date nouă



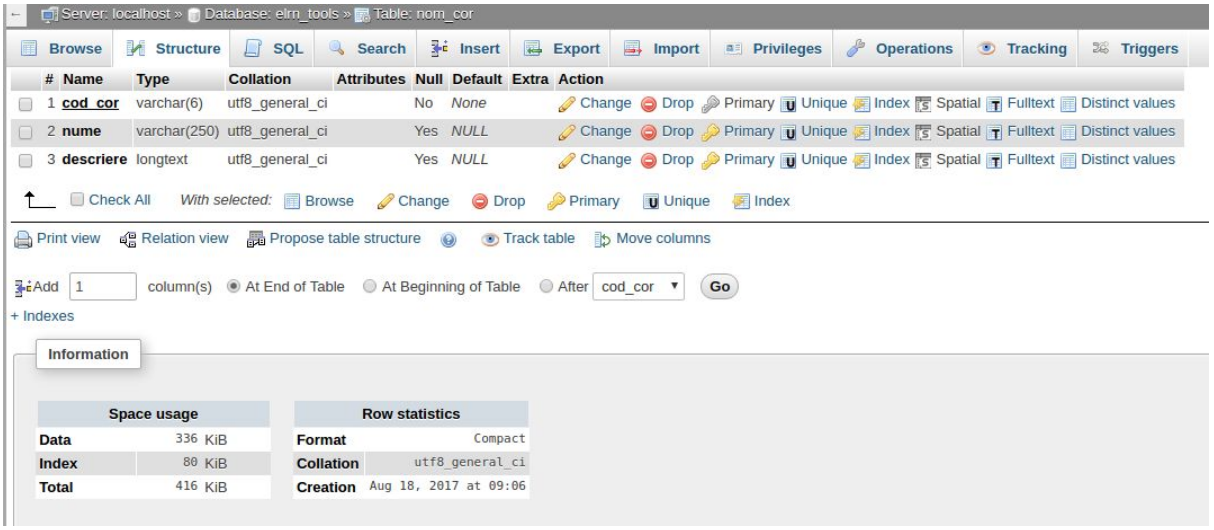
2. Selectăm baza de date şi accesăm tab-ul *Import*. Încărcăm fişierul *.csv* şi alegem din lista *Format* valoarea *CSV*.



3. Stabilim detaliile de format şi selectăm butonul *Go.*



Baza de date rezultată:



4.4 Etape de lucru

În această aplicaţie vom face încă un pas spre principiul separării structurii de funcţionalitate. Astfel, în *index.php* vom avea numai cod HTML pentru descrierea structurii paginii. Toate prelucrările cu PHP le vom face în fişierul *ajax.php.*

4.4.1 Conexiune cu baza de date

Conexiunea cu baza de date o vom face, ca şi până acum, în fişierul *config/db.php* folosind extensia PDO (PHP Data Object).

PHP. Fisierul *db.php*

```
<?php
// date de acces la baza de date
$host = "localhost";
$db_name = "elrn_tools";
$username = "root";
$password = "";

// incearca conexiunea la baza de date
try {
	$db = new PDO("mysql:host={$host};dbname={$db_name}", $username,
	$password);
} catch(PDOException $exception){// arata eroarea in caz de esec
	echo "Eroare la conectare: " . $exception->getMessage();
}
?>
```

Observaţii asupra codului:

Daca avem date de acces la baza de date diferite faţă de cele din exemplu, adaptăm fişierul *db.php.*

4.4.2 Structura fişierului index.php

Pornim de la structura iniţială a fişieruluil *index.php*. O găsim într-un capitol anterior sauo descărcăm accesând link-ul specificat în anexa *Fişiere*.

Pentru că avem deja o complexitate mare, vom dezvolta aplicaţia pas cu pas, verificându-ne şi bucurându-ne de reuşitele parţiale.

În div-ul *.container*, descriem mai întâi structura paginii. Ipoteza de lucru este aceea că pagina se poate extinde adăugând şi alte instrumente similare (de exemplu, CAEN). Astfel, la încărcarea paginii, vom face să fie vizibilă doar lista instrumentelor, detaliile apărând doar la selectarea unui element al listei. Biblioteca *materialize* ne permite să facem asta fără prea multe complicaţii.

Index.php.* Structura iniţială a div-ului *.container

```
<div class="row">
	<h4 class="card-panel teal white-text valign center">Nomenclator COR</h4>
```

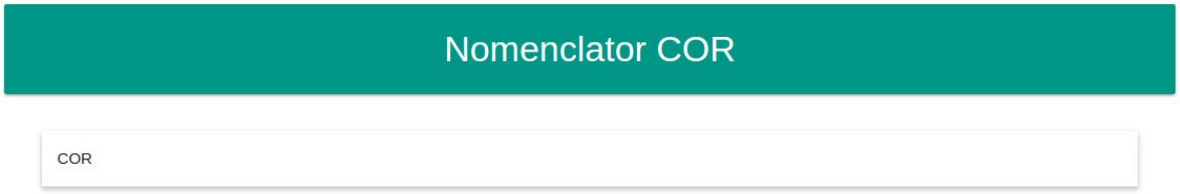
```
</div>
<div id="tools" class="relative">
	<ul class="collapsible popout" data-collapsible="accordion">
		<li id="li_cor">
			<div class="collapsible-header">
				COR
			</div>
			<div class="collapsible-body">
				<div class="row" id="head_cor">
					Explicatii
				</div>
				<div class="row" id="div_cor">
					Continut
				<!--Aici actualizez cu datele transmise prin AJAX-->
				</div>
				<button class="waves-effect waves-green btn
				right">Inchide</button>
				<div class="clearfix"></div>
			</div>
		</li>
	</ul>
</div>  <!-- end #tools -->
```

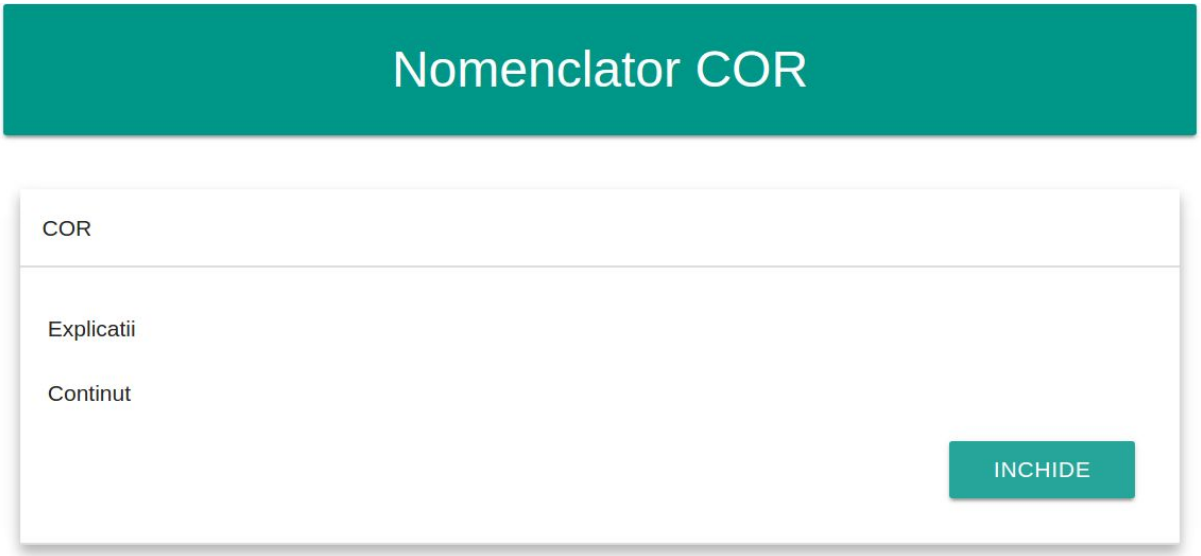
Observaţii asupra codului:

Detalii despre rolul claselor folosite, putem afla accesând

http://materializecss.com/

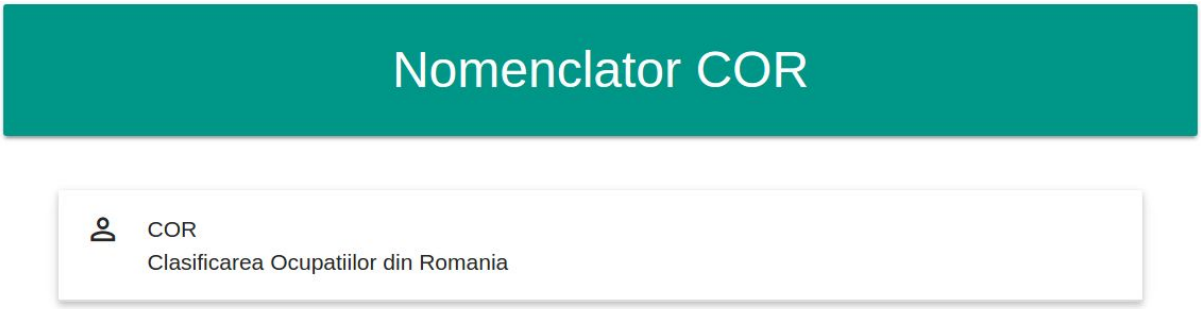
Rezultaul în browser la încărcarea paginii şi la click pe caseta *COR*:





Index.php. **Stilizarea etichetei din lista de instrumente**

```
<div class="collapsible-header">
	<i class="material-icons">perm_identity</i>
	<div>
		<div class="top">COR</div>
		<div class="desc">Clasificarea Ocupatiilor din Romania</div>
	</div>
</div>
```



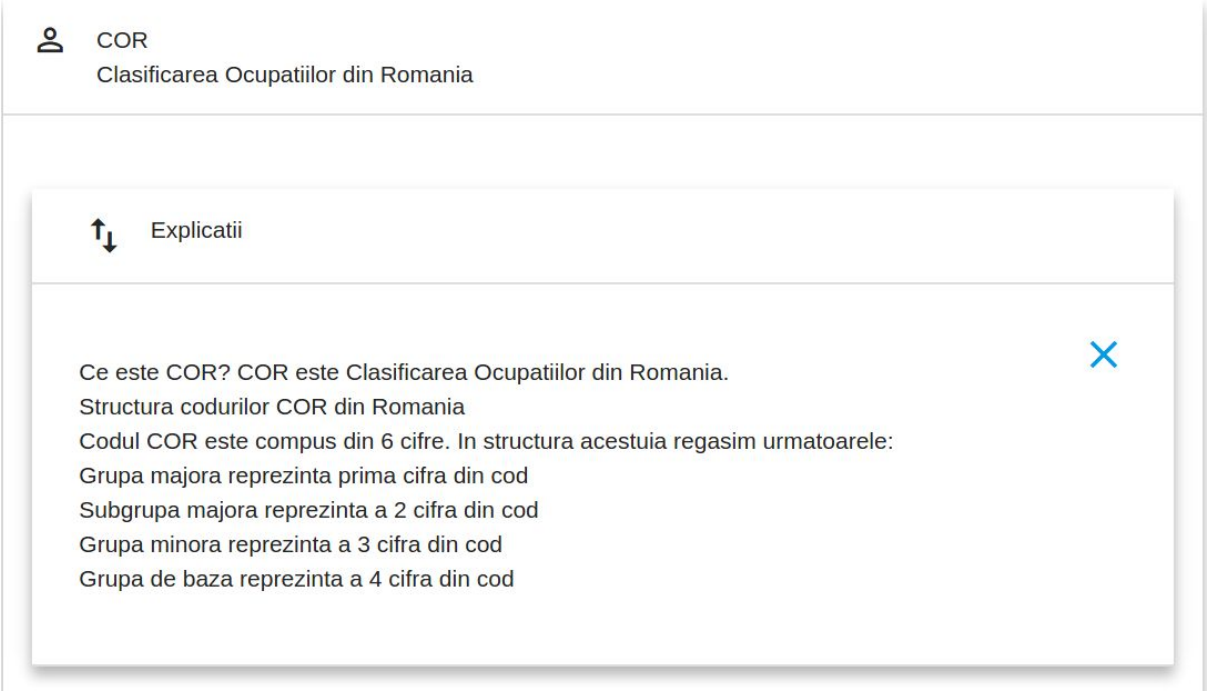
Index.php. **Header-ul *Explicații***

```
<div class="row" id="head_cor">
	<ul class="collapsible popout" data-collapsible="accordion">
	<li>
		<div class="collapsible-header">
			<i class="material-icons right" tyle="font-size:2rem;
			padding-top:0;">Swap_vert</i>
			Explicatii
```

```
            </div>
            <div class="collapsible-body" style="display: none;">
                  <a class="right" style="display:block;cursor:pointer;">
                  <i class="material-icons" style="font-size:2rem;
                  padding-top:0;">close</i>
                  </a>
                   <p>Ce este COR? COR este Clasificarea Ocupatiilor din
                  Romania. <br /> Structura codurilor COR din Romania….
                  </p>
                  <div class="clearfix"></div>
            </div>
      </li>
      </ul>
</div><!--end #head_cor-->
```

Observaţii asupra codului

Blocul *<a>* crează butonul de închidere (x) pentru caseta de explicaţii. Momentan nu este funcţional. Funcţionalitatea o vom da în continuare, folosind cod JavaScript.



Urmează să dăm funcţionalitate butoanelor de închidere.

Index.php.* Codul JavaScript pentru butoanele X şi *Închide

Definim funcţia *closeThis()*:

```
function closeThis(e) {
	$(e).parents('.collapsible-body').first().siblings('.collapsible-header')
	.removeClass('active');
	$(e).parents('li').first().removeClass('active');
	$(e).parents('.collapsible-body').first().slideUp('slow');
}
```

Apelăm funcţia closeThis() la descrierea celor două butoane:

```
<a class="right" onclick="closeThis(this);return false;" style="display:block;
cursor:pointer;">
	<i class="material-icons" style="font-size:2rem;padding-top:0;">close</i>
</a>
<button class="waves-effect waves-green btn right" onclick="closeThis(this);
return false;">Inchide</button>
```

Observaţii asupra codului

Funcţia este definită în zona de scripturi JavaScript.

```
<script type="text/javascript">
	function closeThis(e) {
		$(e).parents('.collapsible-body').first().siblings('.collapsible-header').removeClass('active');
		$(e).parents('li').first().removeClass('active');
		$(e).parents('.collapsible-body').first().slideUp('slow');
	}
</script>
```

Implementăm codul descris mai sus, salvăm, refresh în browser şi verificăm dacă, la click pe fiecare dintre cele două butoane de închidere, avem rezultatul dorit.

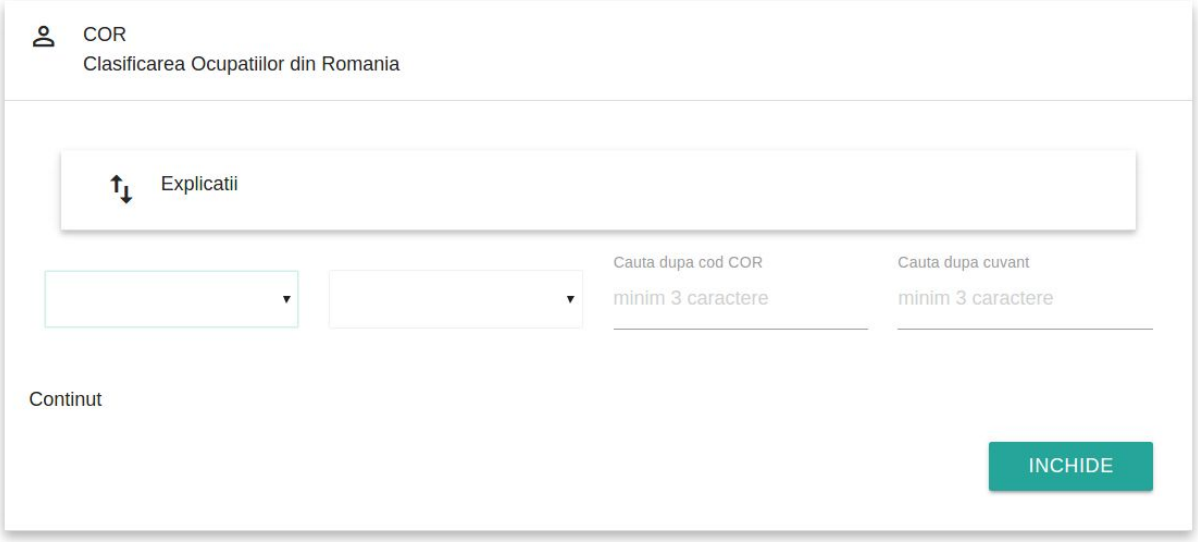
Pasul următor este să descriem structura câmpurilor de căutare.

Index.php. Codul JavaScript pentru butoanele X şi *Închide*

Codul de mai jos îl adăugăm la sfârşitul blocului *#head_cor*.

```
<div class="input-field col s12 m6 l3 buyer" id="div_cor_grupa">
	<select id="cor_grupa" class="browser-default">
	<!--Aici actualizez cu datele transmise prin AJAX-->
	</select>
</div>

<div class="input-field col s12 m6 l3 buyer" id="div_cor_subgrupa">
	<select id="cor_subgrupa" class="browser-default">
	<!--Aici actualizez cu datele transmise prin AJAX-->
	</select>
</div>

<div class="input-field col s12 m6 l3">
	<input id="cor_cod" type="text" class="" placeholder="minim 3
	caractere"/>
```

```
      <label for="cor_cod" >Cauta dupa cod COR</label>
</div>

<div class="input-field col s12 m6 l3">
      <input id="cor_word" type="text" class="" placeholder="minim 3
caractere"/>
      <label for="cor_word">Cauta dupa cuvant</label>
</div>
```



4.4.3 Listele de selecţie. Popularea cu date

Populăm listele de selecţie cu date preluate din baza de date.

Pentru lista de selecţie *grupa majoră*, extragem datele din baza de date în fişierul *ajax.php.* La încărcarea documentului, actualizăm codul HTML al listei de selecţie.

Vom face asta în trei paşi:

1. Definim funcţia JavaScript, *refreshGrupa()*, care va transmite prin AJAX ajax : 1, *action : 'grupa'* cu semnificaţia "*extragem din baza de date toate grupele majore şi generăm câte un element <option> pentru fiecare.*"
2. Apelăm funcţia *refreshGrupa()* în *$(document).ready(function(){}*.
3. În *ajax.php* inerogăm baza de date şi generăm câte un element <option> pentru fiecare grupă majoră.

***Index.php.* Definiţia funcţiei JavaScript** *refreshGrupa()*

Codul de mai jos îl adăugăm în blocul *<script></script>*.

```
function refreshGrupa() {
	$('#cor_grupa').html('');
	$.ajax({
		type: "POST",
		cache:false,
		url: 'ajax.php',
		data: {
			action:'grupa',
			ajax: 1,
		},
		dataType: 'json',
		success: function(response) {
			$('#cor_grupa').html(response.html);
			$('#cor_grupa').material_select();//reactulizez select
			//console.log(response.html); // pt debug
		}
	  });
}
```

Index.php. **Apelul funcţiei JavaScript** *refreshGrupa()*

Codul de mai jos îl adăugăm în blocul *<script></script>*.

```
$(document).ready(function(){
	$('select').material_select();//initializez elementul select
	refreshGrupa();
})
```

ajax.php. **Generarea opţiunilor pentru grupa majoră**

```
<?php
include 'config/db.php';// includ conexiunea la baza de date

//Ajax imi transmite ce am selectat
if($_POST['ajax']){
	$action 	= isset($_POST['action']) ? trim($_POST['action']):NULL;
	$grupa 		= isset($_POST['grupa']) ? trim($_POST['grupa']):NULL;
	$response = array();

	switch($action){
		case 'grupa': // date pentru select #cor_grupa
			$sql = "SELECT cod_cor, nume, descriere
				FROM nom_cor WHERE CHAR_LENGTH(cod_cor)=1";
			$q = $db->prepare($sql);
			$q->execute();
```

```
                    $response['html'] = '<option value="">
                                                Selecteaza grupa majora
                                            </option>';
                    while($row = $q->fetch(PDO::FETCH_ASSOC)){
                          $response['html'] .= '
                          <option value="' .$row['cod_cor']. '">'
                                .$row['cod_cor'].'-' .$row['nume'].
                          '</option>';
                    }
                    break;
      }// end switch
echo json_encode($response);
}// end if($_POST['ajax'])
?>
```

În continuare ne vom ocupa de lista de selecţie pentru subgrupa majoră. Iniţial, lista va conţine toate subgrupele majore din baza de date. Dacă însă selectăm o grupă majoră, lista se va actualiza automat şi va conţine numai subgrupele majore asociate grupei selectate. Etapele pe care le vom parcurge sunt următoarele:

1. Definim funcţia JavaScript, *refreshSubgrupa()*, care va transmite prin AJAX ajax : 1, *action : 'subgrupa'*, dar şi id-ul grupei selectate cu semnificaţia "*extragem din baza de date subgrupele majore asociate grupei selectate şi generăm câte un element <option> pentru fiecare.*"
2. Apelăm funcţia *refreshSubgrupa()* în *$(document).ready(function(){}*, dar şi în momentul în care se schimbă selecţia grupei majore .
3. În *ajax.php* inerogăm baza de date şi generăm câte un element <option> pentru fiecare subgrupă majoră rezultată.

Index.php. **Definiţia funcţiei JavaScript** *refreshSubgrupa()*

Codul de mai jos îl adăugăm în blocul *<script></script>*.

```
function refreshSubgrupa() {
	var grupa = $('#cor_grupa').val();//depind de grupa
	$('#cor_subgrupa').html('');

	$.ajax({
		type: "POST",
		url: 'ajax.php',
		data: {
			action:'subgrupa',
			ajax: 1,
			grupa: grupa,
		},
		dataType: 'json',
		success: function(response) {
			$('#cor_subgrupa').html(response.html);
			$('#cor_subgrupa').material_select();//reactulizez select
			//console.log(response.html); // pt debug
		}
	});// end ajax
}
```

Index.php. **Apelul funcţiei JavaScript** *refreshSubgrupa()*

Codul de mai jos îl adăugăm în blocul *<script></script>*.

```
$(document).ready(function(){
	$('select').material_select();//initializez elementul select
	refreshGrupa();
	$('#cor_grupa').on('change',function(){
			refreshSubgrupa();
	});
})
```

ajax.php. **Generarea opţiunilor pentru subgrupa majoră**

Codul de mai jos îl adăugăm în blocul *switch($action)*

```
case 'subgrupa': // date pentru select #cor_subgrupa
	$grupa 		= isset($_POST['grupa']) ?
	trim($_POST['grupa']):NULL;
	$sql = "SELECT cod_cor, nume, descriere FROM nom_cor WHERE
	CHAR_LENGTH(cod_cor)=2";
	if($grupa!==""){
```

```
            $sql .= " AND SUBSTRING(cod_cor, 1, 1) = :grupa ";
        }
        $q = $db->prepare($sql);
        if($grupa!=="") $q->bindValue(':grupa', $grupa, PDO::PARAM_INT);
        $q->execute();
        $response['html'] = '<option value="">Selecteaza subgrupa
majora</option>';

        while($row = $q->fetch(PDO::FETCH_ASSOC)){
            $response['html'] .= '
                                <option value="' .$row['cod_cor']. '">'
                                    .$row['cod_cor']. '-' .$row['nume'].
                                '</option>';
        }
        break;
```

4.4.4 Arborescența în funcție de selecție

Scopul urmărit era să afișăm structura arborescentă de coduri COR în funcție de filtrul stabilit în zona de căutare.

Etapele pe care le vom parcurge:

1. Definim funcția JavaScript, *refreshCOR()*, care va transmite prin AJAX ajax : 1, *action : 'cor'*, dar și id-ul grupei și subgrupei selectate cu semnificația "*extragem din baza de date și vizualizăm în browser toată arborescența asociată grupei și/sau subgrupei grupei selectate.*"
2. Apelăm funcția *refreshCOR()* în *$(document).ready(function(){}*, dar și în momentul în care se schimbă selecția grupei majore sau subgrupei majore.

3. În *ajax.php* interogăm baza de date şi generăm arborescenţa prin structuri de liste indentate.

***Index.php.* Definiţia funcţiei JavaScript** *refreshCOR()*

Codul de mai jos îl adăugăm în blocul *<script></script>*.

```
function refreshCOR() {
	var grupa = $('#cor_grupa').val();
	var subgrupa = $('#cor_subgrupa').val();
	$('#div_cor').html('');
	if (grupa != null || subgrupa != null) {
		$.ajax({
			type: "POST",
			url: 'ajax.php',
			data: {
				action:'cor',
				ajax: 1,
				grupa: grupa,
				subgrupa: subgrupa,
			},
			dataType: 'json',
			success: function(response) {
				$('#div_cor').html(response.html);
				//console.log(response.html); // pt debug
				$('#div_cor .collapsible').collapsible({
					accordion : false
				});
			}
		 });// end ajax
	}// end if
}
```

***Index.php.* Apelul funcţiei JavaScript** *refreshCOR()*

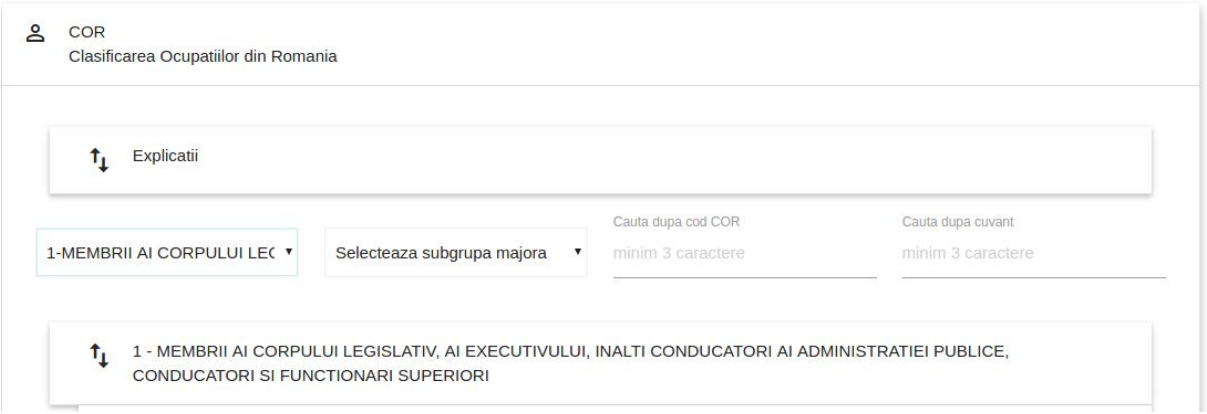
Codul de mai jos îl adăugăm în blocul *<script></script>*.

```
$(document).ready(function(){
	$('select').material_select();//initializez elementul select
	refreshGrupa();
	refreshCOR();
	$('#cor_grupa').on('change',function(){
		refreshSubgrupa();
		refreshCOR();
	});
	$('#cor_subgrupa').on('change',function(){
			refreshCOR();
		});
})
```

ajax.php. Generarea nivelului 1 de arborescenţei în funcţie de selecţie

Codul de mai jos îl adăugăm în blocul *switch($action)*

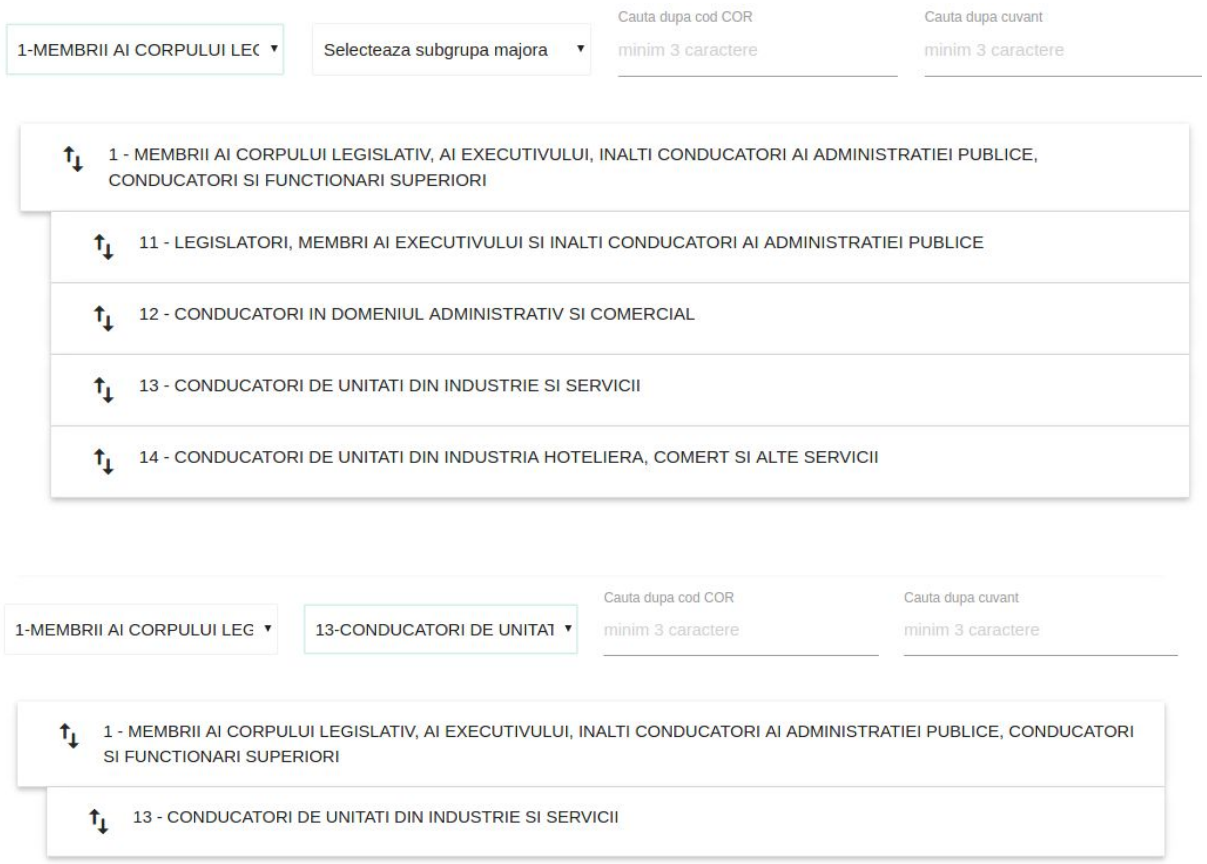
```
case 'cor': // date pentru #div_cor
	$response['html'] = '';
	$response['html'] .= '<ul class="collapsible popout"
	data-collapsible="accordion">';
	if(!$cod && !$word){ // search dupa grupa sau/si subgrupa; search dupa
	cod/cuvant fac numai pe ultimul nivel
		$sql = "SELECT cod_cor, nume, descriere
				FROM nom_cor
				WHERE CHAR_LENGTH(cod_cor)=1";
		if($grupa!==""){
			$sql .= " AND cod_cor = :grupa ";
		}
		if($subgrupa!==""){// am selectat subgrupa
			$sql .= " AND cod_cor = :subgrupa ";
		}
		$q1 = $db->prepare($sql);
		if($grupa!==""){
			$q1->bindValue(':grupa', $grupa, PDO::PARAM_INT);
		}
		if($subgrupa!=="") {
			$q1->bindValue(':subgrupa', substr($subgrupa,0,1),
			PDO::PARAM_INT);
		}
		$q1->execute();
		while($nivel1 = $q1->fetch(PDO::FETCH_ASSOC)){
			$response['html'] .= '
			<li>
				<div class="collapsible-header">
				<i class="material-icons right"
				style="font-size:2rem; padding-top:0;">swap_vert</i>'
				.$nivel1['cod_cor'].' - '.$nivel1['nume'].
				'</div>
				<div class="collapsible-body">
				<a class="right" onclick="closeThis(this);return
				false;" style="display:block; cursor:pointer;">
				<i class="material-icons"
				style="font-size:2rem;padding-top:0;">close</i>
				</a>'
				.$nivel1['descriere'].
				'<div class="clearfix"></div>
				</div>
			</li>';
	}// end while nivel1
}// end if(!$cod && !$word)
break;
```



ajax.php. Generarea nivelului 2 de arborescenţei în funcţie de selecţie

Codul de mai jos îl adăugăm la sfârşitul blocului *while ($nivel1 = $q1->fetch(PDO::FETCH_ASSOC)){}*

```
$sql = "SELECT cod_cor, nume, descriere
            FROM nom_cor
            WHERE CHAR_LENGTH(cod_cor)=2
            AND SUBSTRING(cod_cor, 1, 1) = ".$nivel1['cod_cor'];
if($subgrupa!==""){// am selectat subgrupa
      query .= " AND cod_cor = :subgrupa ";
}
$q2 = $db->prepare($sql);
if($subgrupa!=="") {
      $q2->bindValue(':subgrupa', $subgrupa, PDO::PARAM_INT);
}
$q2->execute();
while($nivel2 = $q2->fetch(PDO::FETCH_ASSOC)){
      $response['html'] .= '
      <li style="margin-left:50px;">
            <div class="collapsible-header">
                  <i class="material-icons right" style="font-size:2rem;
                  padding-top:0;">swap_vert</i>'
                  .$nivel2['cod_cor'].' - '.$nivel2['nume'].
            '</div>
            <div class="collapsible-body">
                  <a class="right" onclick="closeThis(this); return false;"
                  style="display:block;cursor:pointer;">
                  <i class="material-icons" style="font-size:2rem;
                  padding-top:0;">close</i>
                  </a>'
                  .$nivel2['descriere'].
                  '<div class="clearfix"></div>
            </div>
      </li>';
}// end while nivel2
```

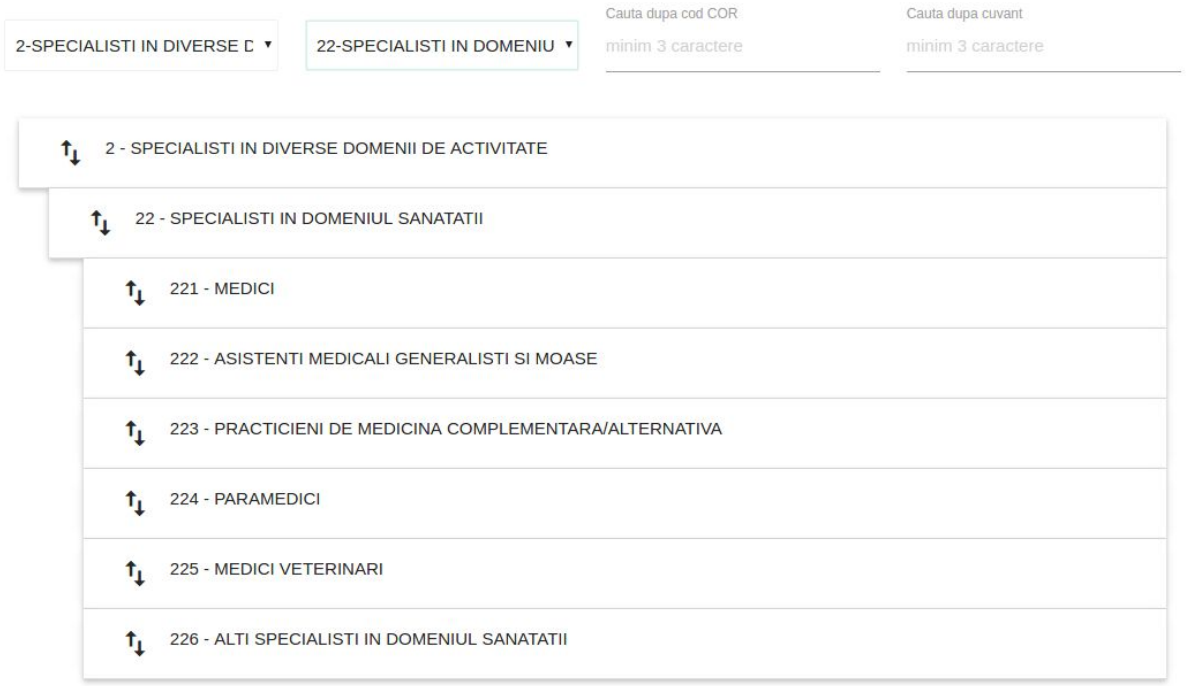


ajax.php. Generarea nivelului 3 de arborescenţei în funcţie de selecţie

Codul de mai jos îl adăugăm la sfârşitul blocului *while ($nivel2 = $q2->fetch(PDO::FETCH_ASSOC)){}*

```
$sql = "SELECT cod_cor, nume, descriere
            FROM nom_cor
            WHERE CHAR_LENGTH(cod_cor)=3
            AND SUBSTRING(cod_cor, 1, 2) = ".$nivel2['cod_cor'];
$q3 = $db->prepare($sql);
$q3->execute();

while($nivel3 = $q3->fetch(PDO::FETCH_ASSOC)){
      $response['html'] .= '
      <li style="margin-left:80px;">
            <div class="collapsible-header">
                  <i class="material-icons right" style="font-size:2rem;
                  padding-top:0;">swap_vert</i>'
                  .$nivel3['cod_cor'].' - '.$nivel3['nume'].
            '</div>
            <div class="collapsible-body">
                  <a class="right" onclick="closeThis(this); return false;"
                  style="display:block;cursor:pointer;">
                  <i class="material-icons" style="font-size:2rem;
                  padding-top:0;">close</i>
                  </a>'
```

```
                    .$nivel3['descriere'].
                    '<div class="clearfix"></div>
                </div>
        </li>';
}// end while nivel3
```

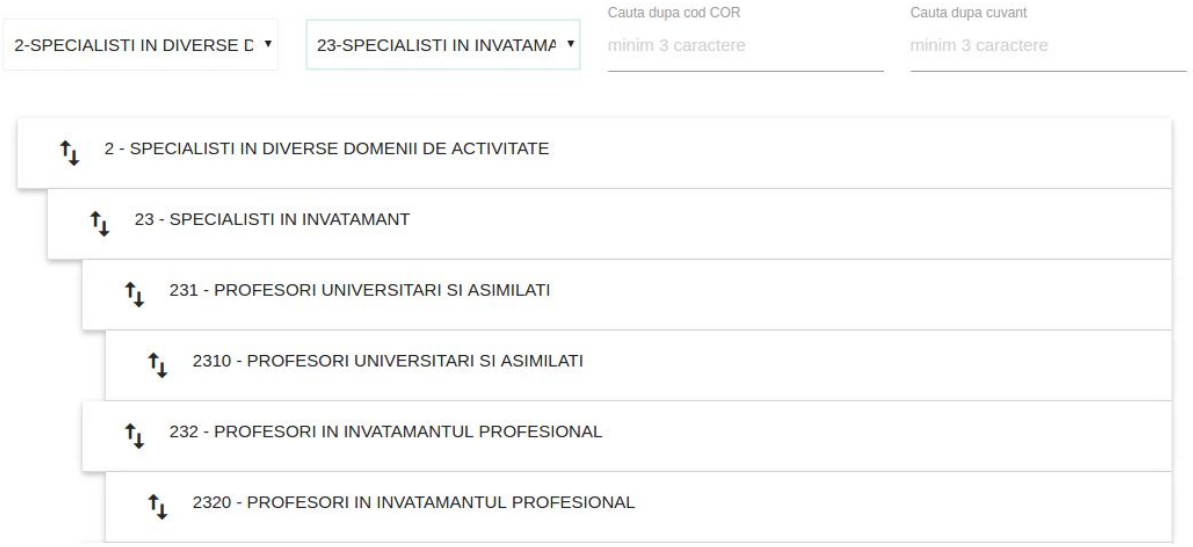


ajax.php. Generarea nivelului 4 de arborescenţei în funcţie de selecţie

Codul de mai jos îl adăugăm la sfârşitul blocului *while ($nivel3 = $q3->fetch(PDO::FETCH_ASSOC)){}*

```
$sql = "SELECT cod_cor, nume, descriere
        FROM nom_cor
        WHERE CHAR_LENGTH(cod_cor)=4
        AND SUBSTRING(cod_cor, 1, 3) = ".$nivel3['cod_cor'];
$q4 = $db->prepare($sql);
$q4->execute();

while($nivel4 = $q4->fetch(PDO::FETCH_ASSOC)){
        $response['html'] .= '
        <li style="margin-left:100px;">
                <div class="collapsible-header">
                        <i class="material-icons right" style="font-size:2rem;
                        padding-top:0;">swap_vert</i>'
                        .$nivel4['cod_cor'].' - '.$nivel4['nume'].
                '</div>
                <div class="collapsible-body">
                        <a class="right" onclick="closeThis(this); return false;"
                        style="display:block;cursor:pointer;">
                        <i class="material-icons" style="font-size:2rem;
                        padding-top:0;">close</i>
```

```
                    </a>'
                    .$nivel4['descriere'].
                    '<div class="clearfix"></div>
                </div>
        </li>';
}// end while nivel4
```



ajax.php. Generarea nivelului 5 de arborescenţei în funcţie de selecţie

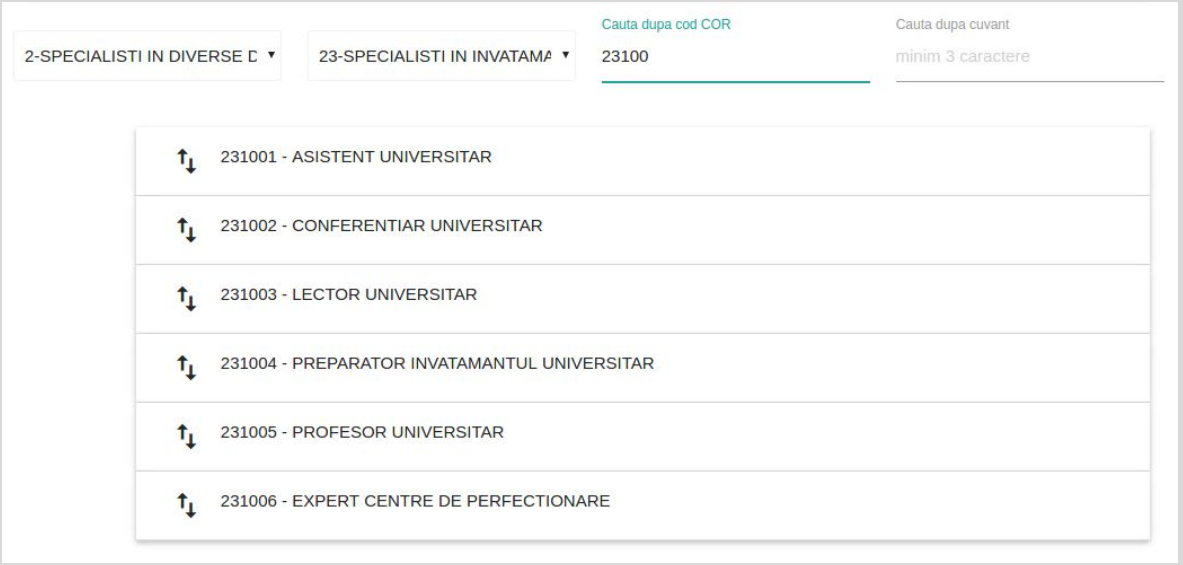
Codul de mai jos îl adăugăm la sfârşitul blocului *while ($nivel4 = $4->fetch(PDO::FETCH_ASSOC)){}*

```
$sql = "SELECT cod_cor, nume, descriere
        FROM nom_cor
        WHERE CHAR_LENGTH(cod_cor)=6
        AND SUBSTRING(cod_cor, 1, 4) = ".$nivel4['cod_cor'];
$q5 = $db->prepare($sql);
$q5->execute();

while($nivel5 = $q5->fetch(PDO::FETCH_ASSOC)){
        $response['html'] .= '
        <li style="margin-left:120px;">
                <div class="collapsible-header">
                        <i class="material-icons right" style="font-size:2rem;
                        padding-top:0;">swap_vert</i>'
                        .$nivel5['cod_cor'].' - '.$nivel5['nume'].
                '</div>
                <div class="collapsible-body">
                        <a class="right" onclick="closeThis(this); return false;"
                        style="display:block;cursor:pointer;">
                        <i class="material-icons" style="font-size:2rem;
                        padding-top:0;">close</i>
                        </a>'
                        .$nivel5['descriere'].
                        '<div class="clearfix"></div>
```

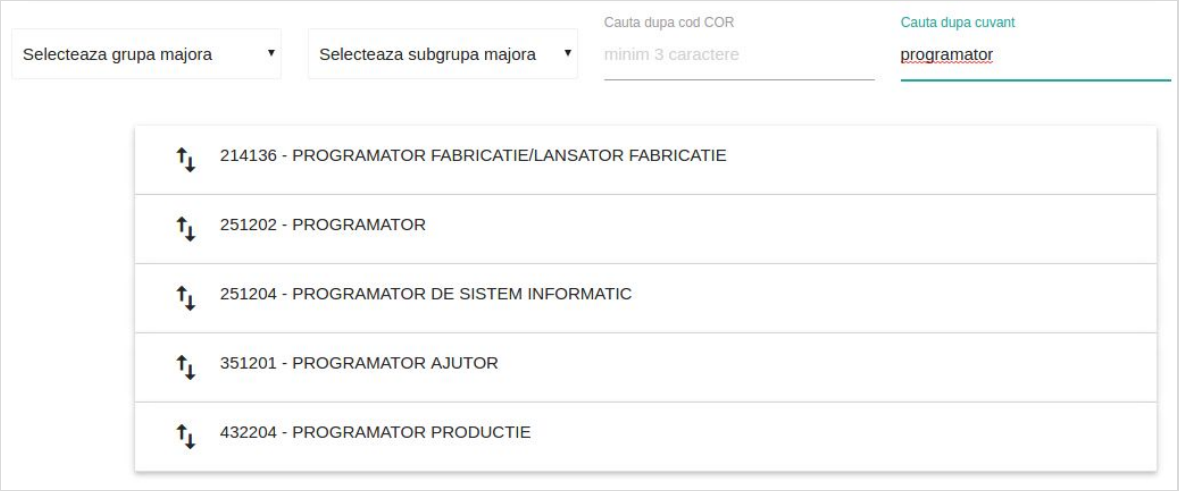
```
            </div>
        </li>';
}// end while nivel5
```

3.6 Aplicaţii propuse

Căutarea după cod COR. Să se adauge funcţionalitate pentru căutarea după cod COR. La tastarea a cel puţin trei caractere din cod, se vor afişa toate ocupaţiile care au în cod secvenţa de căutare.



Căutare după cuvânt. Să se adauge funcţionalitate pentru căutarea după cod cuvânt. La tastarea a cel puţin trei caractere, se vor afişa toate ocupaţiile care au în denumire secvenţa de căutare.



BIBLIOGRAFIE

1. Florentin Eugen Ipate, Monica Popescu, *Dezvoltarea aplicațiilor de baze de date în Oracle 8 şi Oracle Forms 6*, Editura All;
2. Octavian Bîscă, *Baze de date*;
3. Larry Ullman, *PHP for the World Wide Web: Visual QuickStart Guide,* ISBN 0-321-73345-2;
4. Larry Ullman, *PHP and MySQL for Dynamic Web Sites: Visual QuickStart Guide,* ISBN: 0-321-78407-3
5. Cristian Masalagiu, Ioan Asiminoaei, Mirela Tibu, *Didactica predarii informaticii. Editia a II-a revazuta si adaugita,* Editura Polirom 2016
6. http://materializecss.com/
7. http://php.net/
8. https://www.mysql.com/
9. http://data.gov.ro/dataset
10. https://www.edu.ro/