

# **PROIECT SISTEME DE GESTIUNE A BAZELOR DE DATE**

## **DEZVOLTAREA UNUI SITE WEB CU SUPORT PENTRU INTEROGĂRI ȘI NORMALIZARE**

### **TEMĂ: JOB – CERERE ȘI OFERTĂ**

Realizată de: Băbeanu George Valentin

Profesor Laborator: Lect. univ. dr. Șchiopu Daniela

Profesor Curs: Conf. univ. dr. Vlădoiu Monica

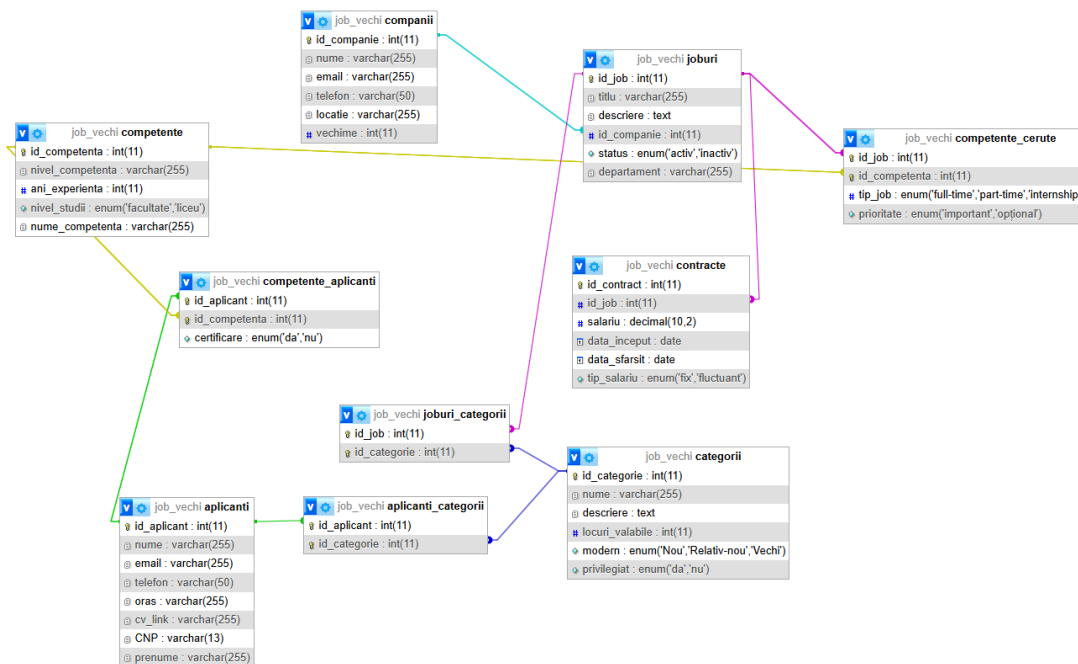
Specializare: Informatică

## Cuprins

Descriere generală.....	2
Schimbările pe care baza de date le-a suferit.....	3
Prezentarea bazei de date curente, după schimbările prezentate .....	7
Analiza legăturilor noi dintre tabele .....	7
Identificarea formei normalizate a bazei de date curente .....	8
Forma normală 1 (FN1) .....	8
Forma normală 2 (FN2) .....	8
Forma normală 3 (FN3) .....	9
Forma normală Boy-Codd (FNBC) .....	9
Analiza anomaliilor și rezolvarea acestora .....	10
Implementarea în site-ul web.....	11
Descriere general și aspect.....	11
Implementarea în cod și interogările date .....	13
Conexiunea la baza de date.....	13
Pagina principală – index.php.....	13
Pagina firmelor – firme.php.....	22
Pagina aplicanților – applicant.php.....	27
Concluzii și observații finale .....	31

## Descriere generală

În cadrul acestui proiect am avut de modificat baza de date pe care am avut-o pe primul semestru, în cazul meu, am avut tema: ”job-cerere-și-ofertă”, așa că m-am gândit încă de pe atunci că această bază de date poate servi unui site web de angajări, cum ar fi ”etoro”, ”e-jobs”, ”bestjobs” ș.a.m.d. Am proiectat baza de date, iar schema conceptuală a acesteia arată așa:



De notat că această bază de date nu este încă normalizată, se poate vedea asta prin faptul că există câmpuri multi-valoare, câmpuri care nu sunt bine-definite sau vagi, dar și existența unor tabele care nu au vreo folosință pentru un site web de angajări. Dar trebuie să înțelegem prima dată logica primului tabel pentru a putea dezvolta baza de date mai departe.

Logica este una simplă, într-un site de angajări aplici la un job diferit, iar acel job este dat de o companie, la jobul respectiv aplică aplicanți. Fiecare job aparține unei categorii (spre ex. IT, HR etc.) are un contract și cere competențe. Se poate vedea că există în baza de date niște tabele complementare care asigură legăturile multi-multi dintre tabele, cum ar fi tabela competențe

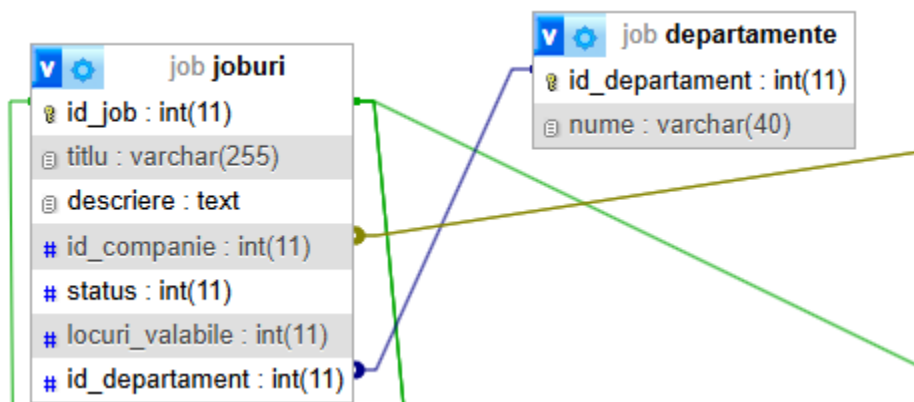
cerute, deoarece un job poate cere mai multe competențe, și aceleași competențe pot fi cerute de mai multe joburi, dar am să explic mai pe larg toate legăturile în baza de date normalizată.

## Schimbările pe care baza de date le-a suferit

Pentru normalizarea bazei de date au fost nevoie de multe schimbări, cum ar fi ștergerea tabelului `aplicanți_categorii`, deoarece nu avea sens în cadrul bazei de date, deoarece un applicant nu aplică la o categorie, aplică direct la jobul respectiv, care acesta aparține unei categorii, deci prima schimbare a fost să șterg tabela: `aplicanți_categorii`. O altă schimbare a fost să șterg toată tabela "categorii", și în schimb am ales să adaug o tabelă departament care conține decât numele departamentului, iar am adăugat o cheie străină pentru a le conecta, legătura dintre aceste tabele este de "one-to-many" deoarece un departament poate avea mai multe joburi, dar un job poate aparține decât unui singur departament. Cu această schimbare am șters, implicit, și tabela `joburi_categorii`.




















Ca un rezumat, această schimbare a șters tabelele: `categorii`, `joburi_categorii` și le-a înlocuit cu tabela departamente.

### LEGĂTURA TABELĂ JOBURI - DEPARTAMENTE



O urmare logică a acestei schimbări a fost să schimb faptul că în prima tabelă, un applicant aplică la o categorie, așa că am creat tabela aplicări care semnifică legătura multi-multi dintre tabela aplicanți și joburi, structura tabelului aplicări:

## STRUCTURĂ TABELĂ APLICARI

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 <b>id_apicare</b> 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 <b>id_job</b> 	int(11)			No	None			 Change  Drop  More
<input type="checkbox"/>	3 <b>id_aplicant</b> 	int(11)			No	None			 Change  Drop  More
<input type="checkbox"/>	4 <b>data_apicare</b>	date			Yes	curdate()			 Change  Drop  More
<input type="checkbox"/>	5 <b>id_status</b> 	int(11)			Yes	NULL			 Change  Drop  More

Aceasta conține id-ul aplicării respective, jobul la care este această aplicație, id-ul applicantului care a aplicat la acel job, data la care a aplicat, și statusul (care poate să aibă 3 valori, 1 – în curs, aplicația nu a fost acceptată sau respinsă, 2 – acceptat, 3 - respins). Legăturile acestei tabele este de "one-to-many" cu tabela joburi, dar și cu tabela aplicanți, deoarece mai mulți aplicanți pot aplica la joburi, iar joburile primesc mai multe aplicații, de asemenea vedem și cheia străină id\_status care conține doar 3 valori (id\_status aparține tabelii status\_aplicare), care deservește mai mult ca un "lookup table", deoarece acesta are doar 3 valori, și noi selectăm ce valori are aplicația respectivă

## CONTINUT TABELĂ "LOOKUP" STATUS APLICARE

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

```
SELECT * FROM `status_aplicare`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

< T >

		id_status	denumire
<input type="checkbox"/>	Edit Copy Delete	1	in curs
<input type="checkbox"/>	Edit Copy Delete	2	acceptat
<input type="checkbox"/>	Edit Copy Delete	3	respins

De asemenea, pentru simplitate puteam să avem decât un câmp, denumire care să fie și cheia primară, deși după am văzut că nu este foarte eficientă această abordare, deoarece eu ar fi trebuit să declar și să folosesc în interogări câmpul denumire care să fie un varchar(12) ceea ce ar fi dus la un timp de așteptare mai mare decât cum ar fi folosit int.

O altă schimbare majoră pe care am făcut-o este să creez o tabelă specială pentru adrese, deoarece în tabela inițială aveam câmpul ”locație”, care era foarte vag, puteam să scriu orașul, strada, țara, nu știm exact ce însemna (de asemenea, aveam câmpul oraș și la aplicanți), am decis

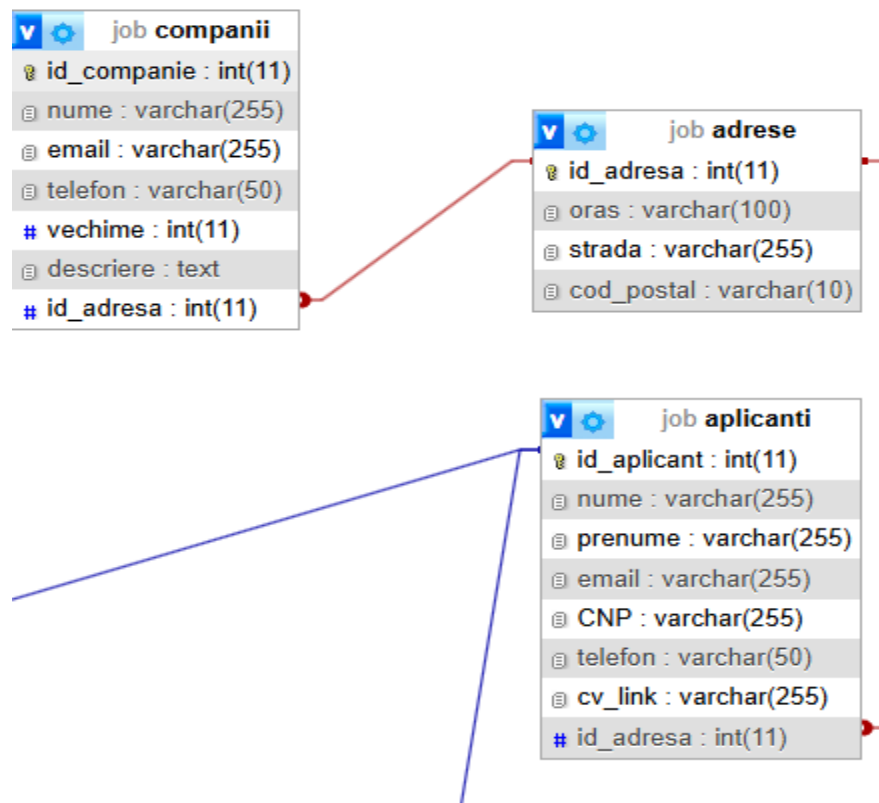
să creez o tabelă universală pentru adrese, pe care să pot să le atribui cui trebuie, fie el companie sau aplicant, structura acestei tabelă arată așa:

## STRUCTURĂ TABELĂ ADRESE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	<b>id_adresa</b>	int(11)			No	None		AUTO_INCREMENT	<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
<input type="checkbox"/> 2	<b>oras</b>	varchar(100)	utf8mb4_general_ci		Yes	NULL			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
<input type="checkbox"/> 3	<b>strada</b>	varchar(255)	utf8mb4_general_ci		Yes	NULL			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>
<input type="checkbox"/> 4	<b>cod_postal</b>	varchar(10)	utf8mb4_general_ci		Yes	NULL			<a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a>

Și pentru a putea conecta această tabelă cu un aplicant sau cu o companie, am adăugat la fiecare dintre table o cheie străină pe care am numit-o id\_adresa și conține adresa persoanei/companiei respective.

## REPREZENTARE VIZUALĂ A LEGĂTURII DINTRE ADRESE ȘI APLICANȚI/COMPANII

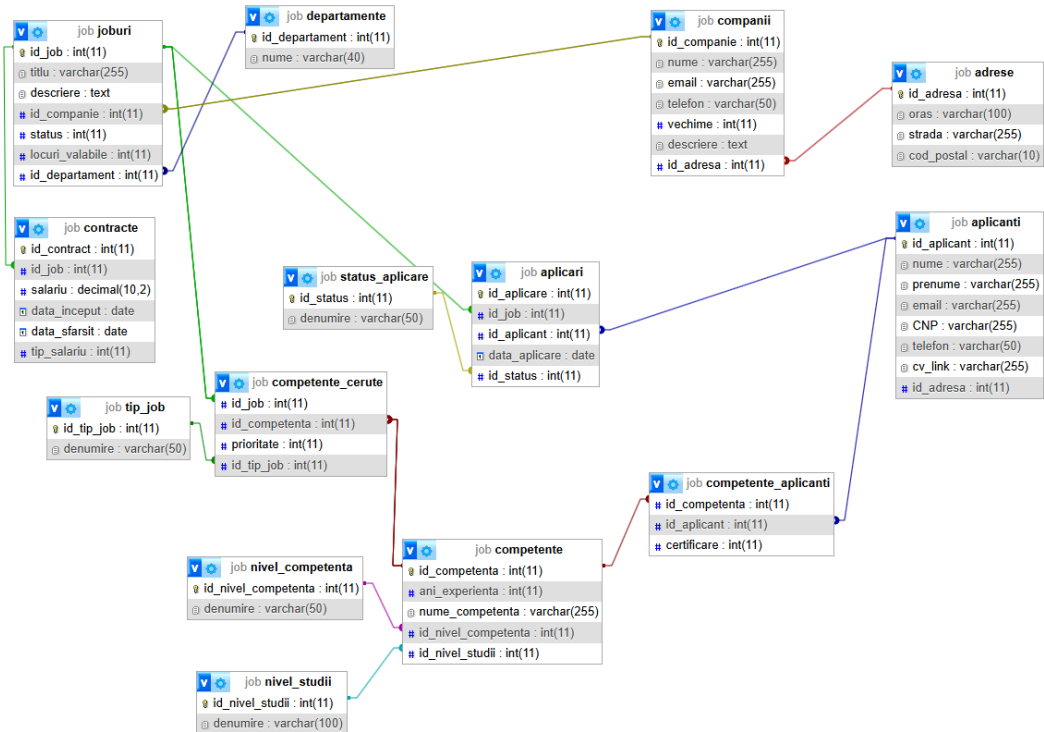


Ultima schimbare pe care am îndeplinit-o a fost aceea să modific toate câmpurile "enum" prezente, ori cu o codificare (spre ex: în tabela contracte 1 – fix, 2 – fluctuant, adică dacă salariu este fix, sau dacă poate fluctua, sau în competențe aplicanti la câmpul certificare, 1 înseamnă că are certicare, 2 înseamnă că nu) ori cu ajutorul unui "lookup" table, așa cum am prezentat la una dintre schimbările anterioare (la table aplicari).

\*Am notat toate schimbările, și majoritatea lucrurilor pe care le-am făcut într-un fișier markdown, pe care am să îl uploadez dacă este nevoie pentru o aprofundare mai bună a schimbărilor.

## Prezentarea bazei de date curente, după schimbările prezentate

După toate schimbările prezentate la capitolul anterior, baza noastră de date este normalizată, lucru pe care îl vom explica în următorul capitol



Observăm că baza noastră este mai bine ”închegată” după schimbările prezentate mai sus, este mai ușor de înțeles și de implementat, ca urmare a normalizării. Un alt lucru pe care îl mai putem observa este că sunt mult mai multe tabele, deoarece am eliminat toate enum-urile de la stagiul inițial, și le-am schimbat, ori cu lookup tables, ori cu o codificare pe care am prezentat-o la schimbări

## Analiza legăturilor noi dintre tabele

În privința legăturilor, acestea sunt în mare parte aceleași, cu excepția faptului că am șters unele tabele, în schimbul lor bagând altele, spre exemplu tabelele aplicanți – adrese sunt într-o relație ”multi-to-one” (fiecare aplicant are o singură adresă), analog și pentru companii-adrese. Relația pe care eu aș numi-o de ”lookup” dintre aplicari și status\_aplicare, dar și relația multi-multi dintre aplicanti și joburi prin prisma tabelii aplicari.



# Identificarea formei normalizate a bazei de date curente

După savârșirea tuturor schimbărilor putem vedea că baza de date se află în **Forma Normală Boyce-Codd (FNBC)** deoarece prezintă toate cerințele pentru formele normale anterioare (1, 2 și 3), dar și cele din FNBC.

## Forma normală 1 (FN1)

Baza de date aparține formei normale 1 pentru că toate atributele conțin valori atomice (nu liste sau seturi), o valoare atomică înseamnă că fiecare celulă dintr-un tabel conține o singură valoare indivizibilă, un exemplu ar putea fi numele unei persoane, să zicem în aplicații avem nume = "Popescu", o valoare care nu este atomică, ar fi în cazul în care este o persoană care are 2 numere de telefon, iar noi gestionăm asta punându-le pe amândouă în aceeași tabelă, ex: telefon = "0721... , 0753...". Observăm că valorile atomice reprezintă o singură informație, și nu poate fi împărțită logic în alte valori mai mici.

După înțelegerea acestor definiții putem observa că baza noastră de date trece acest prim test, deoarece nu avem astfel de valori non-atomice, nu există câmpuri care sunt compuse din mai multe valori, iar fiecare tabel are o cheie primară (excepția fiind tabelele de legătură)

## Forma normală 2 (FN2)

Pentru ca o bază de date să fie în FN2, logic, trebuie să fie și în FN1 și fiecare atribut non-cheie să depindă de întreaga cheie primară, nu doar de o parte din ea, acest lucru este realizat în toate tabelele, spre exemplu în tabelul aplicații:

### STRUCTURĂ TABELUL APLICANȚI:

<div>Table structure</div> <div>Relation view</div>									
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id_aplicant	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
<input type="checkbox"/>	2 nume	varchar(255)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3 prenume	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	4 email	varchar(255)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	5 CNP	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	6 telefon	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	7 cv_link	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	8 id_adresa	int(11)			Yes	NULL			Change  Drop  More

În cazul acesta, toate atributele depind de cheia primară, care este "id\_aplicant", un exemplu în care un tabel nu ar putea trece de FN2 ar fi când am avea o cheie compusă, iar atributele ar depinde numai de o cheie, nu de toată cheia primară compusă

## Forma normală 3 (FN3)

Cerințele formei normale 3 (FN3) sunt similare cu cele ale formei normale 2 (FN2), dar adaugă o condiție suplimentară, pe lângă faptul că trebuie să fie deja în FN2, și, în plus, în FN3 nu trebuie să existe dependențe tranzitive între coloanele non-cheie.

De exemplu, în tabela companii, dacă presupunem că email este unic pentru fiecare companie (ceea ce este realist), putem adăuga o constrângere UNIQUE(email). Astfel, email devine un identificator alternativ pentru companie. În acest caz, toate celelalte coloane non-cheie (precum telefon, descriere, id\_adresa) depind direct de cheia primară(id\_companie) sau de un identificator unic(email), dar nu depind unele de altele.

Așadar, dacă într-o baza de date toate coloanele non-cheie depind doar de cheia primară și nu există dependențe între ele, iar baza este deja în FN1 și FN2, atunci putem spune că este și în FN3.

## STRUCTURĂ TABELĂ COMPANII:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id_companie	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
<input type="checkbox"/>	2 nume	varchar(255)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3 email	varchar(255)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	4 telefon	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	5 vechime	int(11)			Yes	NULL			Change  Drop  More
<input type="checkbox"/>	6 descriere	text	utf8mb4_general_ci		Yes	NULL			Change  Drop  More
<input type="checkbox"/>	7 id_adresa	int(11)			Yes	NULL			Change  Drop  More

## Forma normală Boy-Codd (FNBC)

O bază de date este în forma normală Boy-Codd deoarece este în FN3 (aș cum am argumentat mai sus), și pentru orice dependență ne-trivială  $X \Rightarrow Y$ , X este o supercheie, am să explic această definiție pe baza tabelii aplicanți, pentru că este cea mai clară acolo.

Tabelul aplicanti este în FNBC deoarece toate dependențele funcționale ne-triviale au ca determinant o supercheie (id\_aplicant). Deși CNP ar putea determina alte attribute, nu este folosit ca determinant în schema actuală și nu există dependențe funcționale implicite care să încalce această regulă. Dacă CNP este considerat unic, se poate adăuga o constrângere UNIQUE pentru a-l transforma într-o supercheie și a garanta respectarea FNBC. Putem spune că FNBC este o formă mai strictă a formei normale 3. De asemenea, am prezentat și tabela companii mai sus (la FN3) care se află în FNBC.

## Analiza anomaliilor și rezolvarea acestora

După prezentarea tuturor schimbărilor pe care baza de date le-a suferit și stabilirea formei normale în care aparține baza noastră de date, ce ne rămâne pentru a avea o bază de date funcțională și practică este să analizăm cele 4 anomalii: redundanța nedorită, anomalia de inserare/ștergere/actualizare.

Redundanțele au fost eliminate prin:

- Separarea entităților în tabele distincte (ex: aplicanti, companii, joburi)
- Utilizarea de chei primare și externe pentru a lega entitățile fără a repeta informații descriptive.

Anomaliile sunt prevenite astfel:

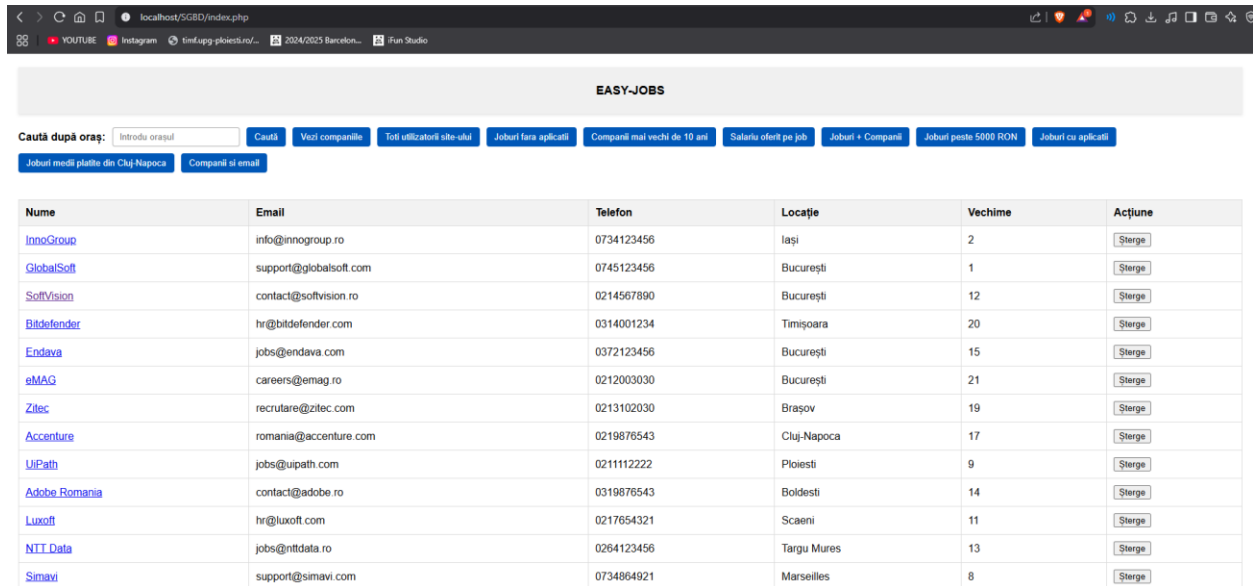
- **Inserarea** unui job sau aplicant nu depinde de existența unui alt set de date inutile.
- **Ștergerea** unei aplicații nu conduce la pierderea datelor despre aplicant sau job, deoarece acestea sunt stocate în tabele separate.
- **Actualizările** asupra denumirii unui status se fac într-un singur loc (status\_aplicare), fără riscul de inconsistență.

Prin modelarea relațiilor *unu-la-mulți* și *mulți-la-mulți* cu ajutorul tabelelor intermediare (ex: aplicari, competente\_aplicanti), baza de date își păstrează integritatea și previne apariția anomaliilor logice sau fizice în operarea datelor.

# Implementarea în site-ul web

## Descriere general și aspect

Pentru implementarea acestei baze de date într-un site, m-am gândit să o fac mai practică, în sensul că, pagina principală a site-ului conține majoritatea lucrurilor de care ai nevoie, pentru implementarea site-ului am folosit php și interogări mysql pentru manipularea datelor din baza de date.



The screenshot shows a web browser displaying the 'EASY-JOBS' website. The page has a search bar at the top with the text 'Caută după oraș:' and a dropdown menu showing 'Introdu orașul'. Below the search bar are several filter buttons: 'Caută', 'Vezi companiile', 'Toți utilizatorii site-ului', 'Joburi fara aplicatii', 'Companii mai vechi de 10 ani', 'Salarii oferite pe job', 'Joburi + Companii', 'Joburi peste 5000 RON', and 'Joburi cu aplicatii'. Below the filters is a table with the following data:

Nume	Email	Telefon	Locație	Vechime	Acțiune
<a href="#">InnoGroup</a>	info@innogroup.ro	0734123456	Iași	2	<a href="#">Sterge</a>
<a href="#">GlobalSoft</a>	support@globalsoft.com	0745123456	București	1	<a href="#">Sterge</a>
<a href="#">SoftVision</a>	contact@softvision.ro	0214567890	București	12	<a href="#">Sterge</a>
<a href="#">Bitdefender</a>	hr@bitdefender.com	0314001234	Timișoara	20	<a href="#">Sterge</a>
<a href="#">Endava</a>	jobs@endava.com	0372123456	București	15	<a href="#">Sterge</a>
<a href="#">eMAG</a>	careers@emag.ro	0212003030	București	21	<a href="#">Sterge</a>
<a href="#">Zilec</a>	recrutare@zilec.com	0213102030	Brașov	19	<a href="#">Sterge</a>
<a href="#">Accenture</a>	romania@accenture.com	0219876543	Cluj-Napoca	17	<a href="#">Sterge</a>
<a href="#">UiPath</a>	jobs@upath.com	0211112222	Ploiesti	9	<a href="#">Sterge</a>
<a href="#">Adobe Romania</a>	contact@adobe.ro	0319876543	Boldesti	14	<a href="#">Sterge</a>
<a href="#">Luxoft</a>	hr@luxoft.com	0217654321	Scaeni	11	<a href="#">Sterge</a>
<a href="#">NTT Data</a>	jobs@nttdata.ro	0264123456	Targu Mures	13	<a href="#">Sterge</a>
<a href="#">Simavi</a>	support@simavi.com	0734864921	Marseilles	8	<a href="#">Sterge</a>

Observăm că fix cum intri pe site avem toate companiile care angajează listate, cu toate datele importante listate, cum ar fi numele companiei, datele de contact, orașul din care este compania și vechimea acelei companii. În plus, am gândit site-ul ca și cum aș fi un administrator, și am posibilitatea să șterg o companie (\*de notat, cheile străine sunt configurate în așa fel încât să se șteargă și celelalte intrări care au legătura cu acea companie).

Putem vedea că putem filtra aceste valori după oraș, în dreapta acestuia avem un buton care realizează o reuniune între tabela de companii și aplicanți (\*pentru a fi o companie trebuie să ți-o înregistrezi pe site), un buton care ne arată toate joburile fără aplicații (\*s-a realizat o diferență pentru realizarea acestei interogări), mai avem și alte butoane la care o să explic mai pe larg folosințele când o să vă prezint codul.

Un lucru important este faptul că companiile au un hyperlink care te trimite pe pagina specială a lor, spre exemplu, dacă dăm click pe InnoGroup, putem vedea date specifice, precum descrierea companiei, un tabel cu joburile valabile ale acestei companii, și numărul de locuri valabile pe post. Se mai vede și numărul de aplicații pe care această firmă le-a primit, câte sunt în curs, și

câte sunt acceptate (mă gândeam să pun și câte sunt respinse, dar am zis ca e mai bine pentru utilizator să vadă doar câte sunt acceptate și în curs, nu era complicat să pun și respinse)

## Bine ai venit la compania InnoGroup

Furnizor de top de echipamente medicale avansate și consumabile, cu un portofoliu diversificat ce include dispozitive inovatoare pentru diagnosticare, tratament și monitorizare, susținute de servicii personalizate de training și mentenanță, cu angajament puternic față de calitatea serviciilor și satisfacția clienților din sistemul medical public și privat.

Mai jos vezi posturile disponibile!

Titlu	Departament	Descriere	Locuri Valabile
Data Analyst	Analiză date	Analizează și interpretează datele pentru a ajuta la luarea deciziilor	34
Data Scientist	Analiză date	Analizează date complexe pentru a sprijini deciziile	8
Financial Analyst	Financiar	Evaluează performanțele financiare și face rapoarte lunare	5
Marketing Analyst	Marketing	Analizează campaniile de marketing și tendințele pieței	5

Aplicații primite: 6 — Acceptați: 3, În curs: 2

Se angajează în: Analiză date, Financiar, Marketing.

Aplicații companiei InnoGroup:

Filtru status: Toți

Nume	Prenume	Titlu Job	Status	Profil
Ion	Popescu	Specialist Marketing Digital	acceptat	<a href="#">Profil</a>
Dumitrescu	Elena	Specialist Marketing Digital	acceptat	<a href="#">Profil</a>
George	Mihai	Data Analyst	respins	<a href="#">Profil</a>
Iudor	Gabriel	Data Analyst	acceptat	<a href="#">Profil</a>
Tai	Mikhail	Financial Analyst	în curs	<a href="#">Profil</a>
Kramnik	Vladimir	Data Scientist	în curs	<a href="#">Profil</a>

De asemenea, am adăugat și numele aplicanților și prenumele, titlul pe care aplică și statusul, și acesta conține un hyperlink, dar de data asta acesta duce spre profilul de pe site al utilizatorului. De precizat că această tabelă am intenționat să o pun să o vadă doar firma respectivă, în momentul acesta, fiecare utilizator se consideră a fi admin, deci are acces la toate tabelele de pe site.

Dacă dăm click pe profilul utilizatorului putem vedea datele acesteia specifice, pe care le introduci atunci când îți creezi cont pe site (înca nu am introdus această funcție), fiecare pagină a unei persoane conține:

EASY-JOBS

[Acasă](#)

### Dumitrescu Elena

Email: elena.dumitrescu@example.com

CNP: 2981123245678

Telefon: 0755444333

Oras: Iasi

CV: [Deschide CV](#)

### Aplicații ale aplicantului

Filtru după status: Toate

Job	Data aplicare	Status
Specialist Marketing Digital	2025-05-30	acceptat
HR Specialist	2025-05-06	acceptat

### Competențe

Nume competență	Nivel	Ani experiență	Studii	Certificare
PHP	Intermediar	3	liceu	nu
Design grafic	Intermediar	2	master	da

Se poate vedea că această pagină conține foarte multe detalii din tabela cu aplicanți, dar și detalii despre companiile la care a aplicat și competențele respectivei persoane. Evident, aici se poate pune o fereastră de ”privacy” pentru că nu sunt date care să fie publice oricui.

În principiu, acestea sunt paginile principale ale site-ului.

## Implementarea în cod și interogările date

### Conexiunea la baza de date

Conexiunea la baza de date este ușor de realizat, am făcut-o într-un fișier separat pe care l-am numit ”db.php”, pentru a realiza conexiunea am declarat 4 variabile care să rețină datele precum parola, username-ul, numele bazei de date și serverul. Eu le am pe cele de bază, acestea se pot modifica folosind ”SET PASSWORD FOR” sau cu ”ALTER USER”

```
db.php
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "job";
6
7 $conn = new mysqli(hostname: $servername, username: $username, password: $password, database: $dbname);
8
9 if($conn -> connect_error)
10     die("Conexiunea a esuat: ". $conn ->connect_error);
11 ?>
```

### Pagina principală – index.php

Prima pagină este formată dintr-un șir de butoane, și rezultatul acestora pe intrările din baza noastră de date, observăm că le-am pus pe toate într-un form, avem un label și un input type pentru filtrarea după oraș

```
<form method="GET" action="index.php">
  <label for="oras">Caută după oraș:</label>
  <input type="text" id="oras" name="oras" value="<?php echo htmlspecialchars(string: $oras_filter); ?>" placeholder="Introdu orașul">
  <button type="submit">Caută</button>
  <a href="index.php"><button type="button">Vezi companiile</button></a>
  <button type="submit" name="op" value="Toti utilizatorii site-ului">Toti utilizatorii site-ului</button>
  <button type="submit" name="op" value="Joburi fara aplicatii">Joburi fara aplicatii</button>
  <button type="submit" name="op" value="Companii mai vechi de 10 ani">Companii mai vechi de 10 ani</button>
  <button type="submit" name="op" value="Salariu oferit pe job">Salariu oferit pe job</button>
  <button type="submit" name="op" value="Joburi + Companii">Joburi + Companii</button>
  <button type="submit" name="op" value="Joburi peste 5000 RON">Joburi peste 5000 RON</button>
  <button type="submit" name="op" value="Joburi cu aplicatii">Joburi cu aplicatii</button>
  <button type="submit" name="op" value="Joburi medii platite din Cluj- napoca">Joburi medii platite din Cluj- Napoca</button>
  <button type="submit" name="op" value="Proiectie">Companii si email</button>
</form>
```

Codul pentru a reuși filtrarea după oraș este acesta, observ acum că am un ”session\_start()” desi acesta nu are vreo folosință în codul meu actual, era dintr-o versiune mai veche în care voiam să fac un utilizator admin și unul care să nu fie admin, oricum nu modifică nimic, este doar o linie

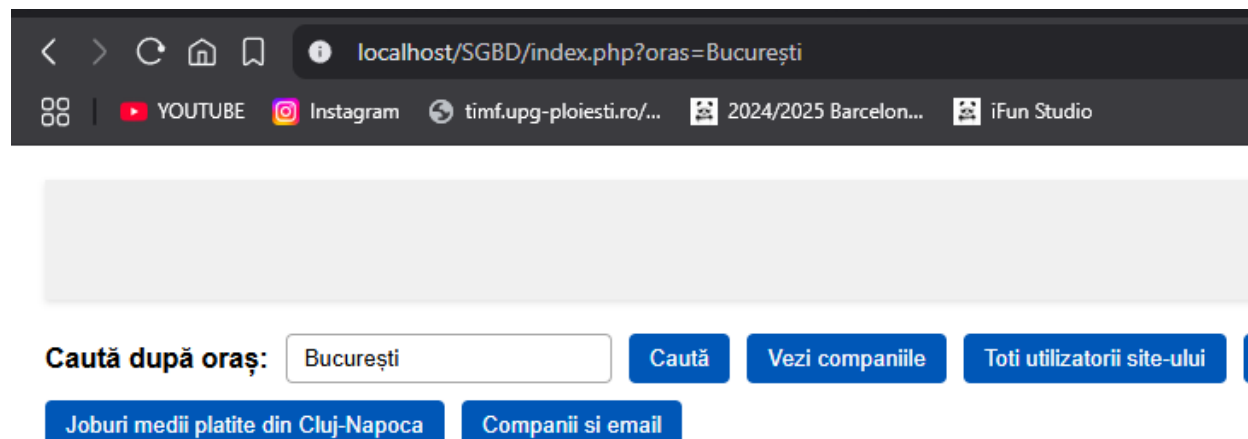
de care am uitat. Avem include('db.php') pentru a ne conecta la baza de date, am prezentat anterior ce conține fișierul db.php.

```
<?php
session_start();
include('db.php');

$oras_filter = "";
$op = isset($_GET['op']) ? $_GET['op'] : "";

if ($op == "" && isset($_GET['oras']) && !empty(trim(string: $_GET['oras']))) {
    $oras_filter = trim(string: $_GET['oras']);
    $stmt = $conn->prepare(query: "SELECT c.*, a.oras FROM companii c JOIN adrese a ON c.id_adresa = a.id_adresa WHERE a.oras LIKE ?");
    $like_param = "%" . $oras_filter . "%";
    $stmt->bind_param(types: "s", var: &$like_param);
    $stmt->execute();
    $result = $stmt->get_result();
} elseif ($op == "") {
    $sql = "SELECT c.*, a.oras FROM companii c JOIN adrese a ON c.id_adresa = a.id_adresa";
    $result = $conn->query(query: $sql);
}
?>
```

Pentru căutarea pe orașe, initial am declarat 2 variabile, oras\_filter care este goală, și \$op care caută în URL să vadă dacă există vreun parametru pentru op, dacă este îl adaugă. (\* de notat, '?' este un if, teoretic îi zice, if(isset(\$\_GET['op'])) \$\_GET['op'] : ""; De menționat, este că \$op nu preia niciodată un oraș, acesta preia de fapt un buton, ne va ajuta la filtrarea pe oraș a altor interogări.



Observăm cum se modifică URL-ul când pune ceva în acel câmp și apăsăm pe butonul "Caută", inițial era doar localhost/SGBD/index.php, de aceea avem o condiție în primul if să existe "oras". De notat, este că avem în if ca op sa fie gol de 2 ori, așa că l-am șters din primul pentru a mai eficientiza puțin codul.

După ce intrăm în instrucțiunea if, oras filter devine ce oraș am introdus noi, apoi avem un \$stmt care selectează toate companiile unde orașul din care provin este "LIKE" (adica asemănător) cu oraș filter, deoarece pe acela l-am dat ca like\_param cu o linie mai jos, apoi executăm. Am folosit "LIKE" deoarece este bun pentru căutări parțiale, spre exemplu dacă dau ca oraș "cluj", nouă o să ne returneze "Cluj-Napoca", precum și alte iterații, adică dacă aveam "Cluj-Nistria" și pe

acela ni-l afișa, și pentru astfel de site-uri, este mai bine să folosești o căutare parțială decât una exactă

Caută după oraș:  Caută Vezi companiile Toti utilizatorii site-ului Joburi fara aplicatii Companii mai vechi de 10 ani Salariu oferit pe job Joburi + Companii Joburi peste 5000 RON Joburi cu aplicatii

Joburi medii platite din Cluj-Napoca Companii si email

Nume	Email	Telefon	Locație	Vechime	Acțiune
Accenture	romania@accenture.com	0219876543	Cluj-Napoca	17	<a href="#">Sterge</a>

Caută după oraș:  Caută Vezi companiile Toti utilizatorii site-ului Joburi fara aplicatii Companii mai vechi de 10 ani Salariu oferit pe job Joburi + Companii Joburi peste 5000 RON Joburi cu aplicatii

Joburi medii platite din Cluj-Napoca Companii si email

Nume	Email	Telefon	Locație	Vechime	Acțiune
GlobalSoft	support@globalsoft.com	0745123456	București	1	<a href="#">Sterge</a>
SoftVision	contact@softvision.ro	0214567890	București	12	<a href="#">Sterge</a>
Endava	jobs@endava.com	0372123456	București	15	<a href="#">Sterge</a>
eMAG	careers@emag.ro	0212003030	București	21	<a href="#">Sterge</a>

Am stabilit inițial că toate butoanele au fost create în acel forms, iar cum am realizat logica cât mai ușor, am inițializat variabila op în primul bloc php, folosința lui urmând să fie aici, avem un switch, iar pe baza valorii dintr-un caz putem să folosim interogarea necesară

```
if ($op != "") {
    echo "<h2>Rezultat pentru operația: " . htmlspecialchars(string: $op) . "</h2><div id='container'><table border='1'><tr>";
    switch ($op) {
        case 'Toti utilizatorii site-ului':
            $sql = "(SELECT nume, email FROM companii)
                UNION
                (SELECT nume, email FROM aplicanti)";
            break;
    }
}
```

Tehnic, rând cu rând, ce facem, este să verificăm dacă op are vreo valoare (acesta își ia valoarea din URL-ul site-ului), vedem în poza de mai jos că scrie explicit valoarea variabilei op.

localhost/SGBD/index.php?oras=&op=Toti+utilizatorii+site-ului

EASY~

Caută după oraș:  Caută Vezi companiile Toti utilizatorii site-ului Joburi fara aplicatii Companii

Joburi medii platite din Cluj-Napoca Companii si email

## Rezultat pentru operația: Toti utilizatorii site-ului

De asemenea, printăm și pe ecran folosind "echo" și numele operației. Acum, putem să analizăm și prima interogare pe care am dat-o, fiind o reuniune, am decis să afișăm într-un tabel toți utilizatorii site-ului, precum și toate firmele. Am decis că este folositor pentru că ajută la evidența utilizatorilor site-ului, spre exemplu, dacă facem anual diferența dintre statutul ăsta al



bazei de date și cel de pe un an, putem vedea ce utilizatori au ieșit ș.a.m.d. De asemenea, am uitat să menționez dar folosim break la final pentru a ieși din switch.

Rezultatul acestei uniuni este:

Rezultat pentru operația: Toti utilizatorii site-ului

nume	email
InnoGroup	info@innogroup.ro
GlobalSoft	support@globalsoft.com
SoftVision	contact@softvision.ro
Bitdefender	hr@bitdefender.com
Endava	jobs@endava.com
eMAG	careers@emag.ro
Zitec	recrutare@zitec.com
Accenture	romania@accenture.com
UiPath	jobs@uipath.com
Adobe Romania	contact@adobe.ro
Luxoft	hr@luxoft.com
NTT Data	jobs@nttdata.ro
Simavi	support@simavi.com
Ion	ion.popescu@mail.com
Ionescu	maria.ionescu@mail.com

Acum că am explicat cam cum funcționează acest cod simplu, o să ne axăm mai mult pe logica interogărilor

## Joburi fără aplicații

Observăm că următoarea interogare este o diferență (pentru că avem NOT IN) între tabela joburi și aplicări (înainte să facem diferența, realizăm și o operație de joncțiune între tabela joburi și companii) și teoretic diferența aceasta ne spune că toate id-urile de joburi care nu sunt în tabela de aplicari, atunci, logic, acel job nu are aplicații.

```
case 'Joburi fara aplicatii':
    $sql = "SELECT j.titlu, c.nume AS companie
            FROM joburi j
            JOIN companii c ON j.id_companie = c.id_companie
            WHERE j.id_job NOT IN (
                SELECT id_job FROM aplicari
            )";
    break;
```

## Rezultatul:

Rezultat pentru operația: Joburi fara aplicatii

titlu	companie
Marketing Analyst	InnoGroup
Technical Writer	Accenture
IT Support Technician	Endava
SEO Specialist	Bitdefender
Product Manager	Accenture
Operations Manager	GlobalSoft
Cybersecurity Specialist	eMAG
Graphic Animator	Bitdefender
HR Coordinator	Endava
Logistics Manager	Zitec
Cloud Engineer	eMAG

## Joburi peste 5000 ron

Această interogare este o joncțiune simplă între două tabele (joburi și contracte) care are drept folosință afișarea joburilor mai bine plătite, cele care pleacă de la 5000 de RON în sus, este o joncțiune simplă cu o clauză (WHERE) după.

```
case 'Joburi peste 5000 RON':  
    $sql = "SELECT joburi.titlu, joburi.locuri_valabile, contracte.salariu FROM joburi  
            JOIN contracte ON joburi.id_job = contracte.id_job  
            WHERE salariu >= 5000";  
    break;
```

## Rezultatul:

titlu	locuri_valabile	salariu
Data Analyst	34	5300.00
Project Manager	2	8000.00
Marketing Specialist	65	5200.00
DevOps Engineer	3	9000.00
Sales Manager	40	8500.00
Product Owner	23	8200.00
Business Analyst	1	6100.00
Frontend Developer	15	7200.00
Social Media Manager	12	5600.00
Financial Analyst	5	6700.00
Technical Recruiter	6	6000.00
Sales Representative	20	7100.00
Scrum Master	4	7500.00
BI Developer	9	6800.00
Fullstack Developer	3	7000.00

## Companii mai vechi de 10 ani

Interogarea aceasta este o selecție simplă, cu opțiunea că arată companiile mai vechi de 10 ani, m-am gândit că această interogare poate face utilizatorul să aplice la firmele care sunt deja cimentate pe piață

```
case 'Companii mai vechi de 10 ani':  
    $sql = "SELECT nume, email, vechime FROM companii WHERE vechime > 10";  
    break;
```

Rezultatul:

Rezultat pentru operația: Companii mai vechi de 10 ani

nume	email	vechime
SoftVision	contact@softvision.ro	12
Bitdefender	hr@bitdefender.com	20
Endava	jobs@endava.com	15
eMAG	careers@emag.ro	21
Zitec	recrutare@zitec.com	19
Accenture	romania@accenture.com	17
Adobe Romania	contact@adobe.ro	14
Luxoft	hr@luxoft.com	11
NTT Data	jobs@nttdata.ro	13

## Salariu oferit pe job

Această funcție ajută la transparența dintre companii și joburile oferite, deoarece această interogare afișează numele companiei, titlul jobului și salariul oferit pe acel job în momentul actual, se realizează două joncțiuni deoarece folosim 3 coloane din 3 tabele diferite.

```
case 'Salariu oferit pe job':  
    $sql = "SELECT j.titlu, c.nume AS companie, co.salariu FROM contracte co  
            JOIN joburi j ON co.id_job = j.id_job  
            JOIN companii c ON j.id_companie = c.id_companie";  
    break;
```

Rezultatul:

Rezultat pentru operația: Salariu oferit pe job

titlu	companie	salariu
Specialist Marketing Digital	InnoGroup	3500.00
Manager Vânzări	GlobalSoft	4000.00
Data Analyst	InnoGroup	5300.00
Project Manager	GlobalSoft	8000.00
UX/UI Designer	SoftVision	4800.00
Marketing Specialist	Bitdefender	5200.00
HR Specialist	Endava	4600.00
DevOps Engineer	eMAG	9000.00

## Joburi + Companii

Această joncțiune simplă este destul de folositoare deoarece îți arată toate joburile cu locurile valabile și așa se pot face foarte multe rapoarte pe cele mai căutate joburi, cele mai necăutate, cele mai multe ș.a.m.d.

```
case 'Joburi + Companii':
    $sql = "SELECT j.titlu, c.nume AS companie, j.locuri_valabile
            FROM joburi j
            JOIN companii c ON j.id_companie = c.id_companie";
    break;
```

Rezultatul:

Rezultat pentru operația: Joburi + Companii

titlu	companie	locuri_valabile
Specialist Marketing Digital	InnoGroup	12
Manager Vânzări	GlobalSoft	5
Data Analyst	InnoGroup	34
Project Manager	GlobalSoft	2
UX/UI Designer	SoftVision	6
Marketing Specialist	Bitdefender	65
HR Specialist	Endava	32
DevOps Engineer	eMAG	3
Sales Manager	Zitec	40
Product Owner	Accenture	23
Business Analyst	UIPath	1
Frontend Developer	SoftVision	15

## Joburi medii plătite din Cluj-Napoca

Această interogare este utilă pentru a evidenția toate joburile bine plătite (peste 4000 RON) din orașul Cluj-Napoca, împreună cu informații despre aplicanții care au aplicat la aceste joburi. Interogarea este compusă din mai multe joncțiuni, ceea ce permite extragerea de date din mai multe tabele într-un mod eficient și clar.

Se observă că în interogare se folosește o subinterogare care selectează doar joburile cu salarii mari (job\_salariu\_mare) și apoi se face legătura cu restul tabelor prin joncțiuni:

- cu aplicari, pentru a afla cine a aplicat la aceste joburi,
- cu aplicanti, pentru a extrage detalii despre acei candidați,
- cu companii și adrese, pentru a aduce informații despre companie și locație.

Filtrarea finală se face astfel încât să fie returnate doar joburile din orașul Cluj-Napoca.

Această interogare poate fi folosită în rapoarte legate de cele mai competitive și bine plătite joburi dintr-un anumit oraș, dar și pentru a evidenția interesul aplicanților pentru astfel de oferte.

```

case 'Joburi medii platite din Cluj-napoca':
    $sql = "SELECT
        ap.nume AS NumeAplicant,
        ap.prenume AS PrenumeAplicant,
        ap.email AS EmailAplicant,
        job_salariu_mare.titlu AS JobAplicat,
        c.nume AS CompanieJob,
        adr.oras AS OrasCompanie
    FROM
        (
            SELECT
                j.id_job,
                j.titlu,
                j.id_companie
            FROM
                joburi j
            JOIN
                contracte co ON j.id_job = co.id_job
            WHERE
                co.salariu > 4000
        ) AS job_salariu_mare
    JOIN
        aplicari apl ON job_salariu_mare.id_job = apl.id_job
    JOIN
        aplicanti ap ON apl.id_aplicant = ap.id_aplicant
    JOIN
        companii c ON job_salariu_mare.id_companie = c.id_companie
    JOIN
        adrese adr ON c.id_adresa = adr.id_adresa
    WHERE
        adr.oras = 'Cluj-Napoca';";
break;

```

Rezultatul:

Rezultat pentru operația: Joburi medii platite din Cluj-napoca

NumeAplicant	PrenumeAplicant	EmailAplicant	JobAplicat	CompanieJob	OrasCompanie
Tudor	Gabriel	gabriel.tudor@example.com	Product Owner	Accenture	Cluj-Napoca
Tal	Mikhail	m.tal@mail.ru	Product Owner	Accenture	Cluj-Napoca
Botvinnik	Mikhail	m.botvinnik@mail.ru	Scrum Master	Accenture	Cluj-Napoca

## Joburi cu aplicații la companii cu vechime de peste 5 ani

Această interogare este folosită pentru a evidenția toate joburile la care s-au făcut aplicări și care aparțin unor companii cu vechime mai mare de 5 ani. Este o interogare compusă, care folosește atât o subinterogare, cât și o joncțiune pentru a aduce informațiile necesare din mai multe tabele.

În prima parte, subinterogarea selectează toate joburile care au cel puțin o aplicație (verificând existența lor în tabelul aplicari), împreună cu titlul jobului și ID-ul companiei care a postat acel job.

După aceea, printr-o joncțiune cu tabelul companii, se extrag date despre compania care a postat fiecare job – în special numele companiei și vechimea ei în ani.

Filtrarea finală se face astfel încât să se returneze doar companiile cu o vechime mai mare de 5 ani, ceea ce poate fi foarte util în analize privind stabilitatea și experiența companiilor care atrag aplicanți.

Această interogare este utilă pentru a înțelege mai bine ce fel de companii primesc aplicații – cele cu experiență mai mare, care pot inspira mai multă încredere candidaților.

```
case 'Joburi cu aplicatii':
    $sql = "SELECT
        job_cu_aplicatii.titlu AS TitluJob,
        c.nume AS NumeCompanie,
        c.vechime AS VechimeCompanie
    FROM
        (
            SELECT
                j.id_job,
                j.titlu,
                j.id_companie
            FROM
                joburi j
            WHERE
                j.id_job IN (SELECT id_job FROM aplicari)
        ) AS job_cu_aplicatii
    JOIN
        companii c ON job_cu_aplicatii.id_companie = c.id_companie
    WHERE
        c.vechime > 5";
break;
```

Rezultatul:

Rezultat pentru operația: Joburi cu aplicatii

TitluJob	NumeCompanie	VechimeCompanie
UX/UI Designer	SoftVision	12
Marketing Specialist	Bitdefender	20
HR Specialist	Endava	15
DevOps Engineer	eMAG	21
Sales Manager	Zitec	19
Product Owner	Accenture	17
Business Analyst	UiPath	9
Frontend Developer	SoftVision	12

## Companii+email

Acum că am trecut de interogările mai complexe, avem poate cea mai simplă interogare, proiecția, aceasta se vrea o variantă minimalistă primului tabel care afișează toate companiile, aceasta afișează doar compania și mailul acesteia.

```

case 'Proiectie':
    $sql = "SELECT nume, email from companii";
    break;

default:
    $sql = "";

```

Afișarea acestor interogări este foarte simplă, se realizează fix după switch-ul de mai sus.

```

if (!empty($sql)) {
    $res = $conn->query(query: $sql);
    if ($res && $res->num_rows > 0) {
        $fields = $res->fetch_fields();
        foreach ($fields as $field) {
            echo "<th>" . htmlspecialchars(string: $field->name) . "</th>";
        }
        echo "</tr>";
        while ($row = $res->fetch_assoc()) {
            echo "<tr>";
            foreach ($row as $cell) {
                echo "<td>" . htmlspecialchars(string: $cell) . "</td>";
            }
            echo "</tr>";
        }
    } else {
        echo "<tr><td colspan='10'>Niciun rezultat.</td></tr>";
    }
    echo "</table></div>";
}
?>

```

Pe scurt, verificăm ca sql-ul să existe, salvăm rezultatul în variabila res, apoi dacă res există și returnează un număr de linii mai mare de 0 (adică dacă există înregistrări) se afișăm, este ceva destul de basic și destul de comun folosit.

De asemenea, la sfârșitul acestuia închidem conexiunea la baza de date folosind \$conn -> close().

## Pagina firmelor – firme.php

Pagina firmelor este realizată foarte simplu, nu conține atât de multe interogări folosind butoane, de regulă acestea sunt deja afișate, folosința acestei pagini este de a oferi aplicantului informații clare despre o companie, precum o descriere, o tabelă cu joburile valabile ale firmei respective.

Dar oferă și un fel de control panel pentru firme pentru că acestea pot vedea aplicațiile primite și să le filtreze.

localhost/SGBD/firme.php?nume=InnoGroup

EASY-JOBS

**Bine ai venit la compania InnoGroup**

Furnizor de top de echipamente medicale avansate și consumabile, cu un portofoliu diversificat ce include dispozitive inovatoare pentru diagnosticare, tratament și monitorizare, susținute de servicii personalizate de training și mentenanță, cu angajament puternic față de calitatea serviciilor și satisfacția clienților din sistemul medical public și privat.

Mai jos vezi posturile disponibile!

Titlu	Departament	Descriere	Locuri Valabile
Data Analyst	Analiză date	Analizează și interpretează datele pentru a ajuta la luarea deciziilor.	34
Data Scientist	Analiză date	Analizează date complexe pentru a sprijini deciziile.	8
Financial Analyst	Financiar	Evaluează performanțele financiare și face rapoarte lunare.	5
Marketing Analyst	Marketing	Analizează campaniile de marketing și tendințele pieței.	5

Aplicații primite: 6 — Acceptați: 3, În curs: 2

Se angajează în: Analiză date, Financiar, Marketing.

Aplicații companiei InnoGroup:

Filtru status: Toți

Nume	Prenume	Titlu Job	Status	Profil
Ion	Popescu	Specialist Marketing Digital	acceptat	<a href="#">Profil</a>
Dumitrescu	Elena	Specialist Marketing Digital	acceptat	<a href="#">Profil</a>
George	Mihai	Data Analyst	respins	<a href="#">Profil</a>
Tudor	Gabriel	Data Analyst	acceptat	<a href="#">Profil</a>
Iul	Mikhail	Financial Analyst	în curs	<a href="#">Profil</a>
Kramnik	Vladimir	Data Scientist	în curs	<a href="#">Profil</a>

Inițial, pentru a putea merge pe această pagină, avem un hyperlink care ne trimite pe pagina localhost/SGBD/firme.php?nume= ”numele companiei”, așa că noi putem să setăm pentru fiecare pagină să arate doar tabelele lor, folosind o variabilă care reține numele pe baza acestui URL.

```
if (isset($_GET['nume'])) {  
    $nume_companie = $_GET['nume'];  
}
```

Prin această mini-instrucțiune, aproape să nici nu o vezi, prin aceasta funcționează întreaga pagină, altfel nu o să știm niciodată pe ce pagină a cărei firme suntem, de asemenea, dacă utilizatorul accesează această firmă fara să dea click pe hyperlinkul unei firme (adică scrie direct în bara de căutare localhost/SGBD/firme.php), site-ul nu recunoaște pe pagina cărei firme se află, așa că afișează un mesaj secret

localhost/SGBD/firme.php

EASY-JOBS

**\*Acesta este un secret.. Shh. Nu mai spune la nimeni și întoarce-te pe index...**



Acum că am stabilit puțin ce și cum, descrierea companiei, id-ul și toate datele pe care le-am folosit în începutul paginii am făcut printr-un statement, am "bind-uit", adică atribuit acele valori altor variabile pe care le-am numit \$descriere și id\_companie și le-am afișat pe ecran cu niște formataări speciale html (h2 și htmlspecialchars)

```
<div>
<?php
if (isset($_GET['nume'])) {
    $nume_companie = $_GET['nume'];
    $stmt = $conn->prepare(query: "SELECT descriere, id_companie FROM companii WHERE nume = ?");
    $stmt->bind_param(types: "s", var: &$nume_companie);
    $stmt->execute();
    $stmt->bind_result(var: &$descriere, vars: &$id_companie);
    $stmt->fetch();
    $stmt->close();

    echo "<h1>Bine ai venit la compania <strong>" . htmlspecialchars(string: $nume_companie) . "</strong></h1>";
    if ($descriere)
        echo "<h2>" . htmlspecialchars(string: $descriere) . "<br>Mai jos vezi posturile disponibile!</h2>";

    $stmt = $conn->prepare(query: "
        SELECT DISTINCT d.numa
        FROM joburi j
        JOIN departamente d ON j.id_departament = d.id_departament
        WHERE j.id_companie = ? AND j.status = 1");
    $stmt->bind_param(types: "i", var: &$id_companie);
    $stmt->execute();
    $res = $stmt->get_result();
    $departamente = [];
    while ($row = $res->fetch_assoc()) {
        $departamente[] = $row['numa'];
    }
    $stmt->close();
} else {
    echo "<h1>*Acesta este un secret.. Shh. Nu mai spune la nimeni și întoarce-te pe index...</h1>";
    exit;
}
?>
</div>
```

Totul este realizat într-un div.

Apoi pentru realizarea tabelului avem o joncțiune în care afișăm toate joburile oferite de acea companie pe site-ul nostru, este o joncțiune simplă (\* de notat că am folosit ca referință id-ul pe care l-am extras mai devreme). Interogarea SQL inițială afișează tabelul acesta, iar vedem vectorul \$departamente pe care l-am declarat, acela afișează ca un fel de rezumat doar departamentele care angajează, aceasta în cazul în care sunt foarte multe intrări ne ajută să vedem foarte clar unde se angajează

## Bine ai venit la compania InnoGroup

Furnizor de top de echipamente medicale avansate și consumabile, cu un post-tratament și monitorizare, susținute de servicii personalizate de training și mentoriaj pentru clienții din sistemul medical public și privat.  
Mai jos vezi posturile disponibile!

Titlu	Departament	Descriere	Locuri Valabile
Data Analyst	Analiză date	Analizează și interpretează datele pentru a ajuta la luarea deciziilor.	34
Data Scientist	Analiză date	Analizează date complexe pentru a sprijini deciziile.	8
Financial Analyst	Financiar	Evaluează performanțele financiare și face rapoarte lunare.	5
Marketing Analyst	Marketing	Analizează campaniile de marketing și tendințele pieței.	5

De asemenea, am folosit și o interogare în care am folosit o funcție în care să avem o sumă care ne adaugă 1 pe variabila specifică aplicației, adică avem "acceptati" care crește cu 1 dacă găsim o aplicație acceptată, "incurs" dacă găsim o aplicație în curs, și "respinsi" dacă găsim o aplicație respinsă

```
<?php
$stmt = $conn->prepare(query: "
SELECT
    SUM(CASE WHEN a.id_status = 2 THEN 1 ELSE 0 END) AS acceptati,
    SUM(CASE WHEN a.id_status = 1 THEN 1 ELSE 0 END) AS incurs,
    SUM(CASE WHEN a.id_status = 3 THEN 1 ELSE 0 END) AS respinsi
FROM aplicari a
JOIN joburi j ON a.id_job = j.id_job
WHERE j.id_companie = ?");
$stmt->bind_param(types: "i", var: &$id_companie);
$stmt->execute();
$stmt->bind_result(var: &$acc, vars: &$curs, $resp);
$stmt->fetch();
$total = $acc + $curs + $resp;
$stmt->close();

echo "<h2>Aplicații primite: $total &mdash; Acceptați: $acc, În curs: $curs</h2>";
if (!empty($departamente)) {
    echo "<h2>Se angajează în: " . implode(separator: ' ', array: array_map(callback: 'htmlspecialchars', array: $departamente)) . "</h2>";
}
?>
```

Această porțiune de cod și de afișare, arată așa pe aplicația web:

Titlu	Departament	Descriere	Locuri Valabile
Data Analyst	Analiză date	Analizează și interpretează datele pentru a ajuta la luarea deciziilor.	34
Data Scientist	Analiză date	Analizează date complexe pentru a sprijini deciziile.	8
Financial Analyst	Financiar	Evaluează performanțele financiare și face rapoarte lunare.	5
Marketing Analyst	Marketing	Analizează campaniile de marketing și tendințele pieței.	5

**Aplicații primite: 6 — Acceptați: 3, În curs: 2**

**Se angajează în: Analiză date, Financiar, Marketing.**

**Aplicații companiei InnoGroup:**

Filtru status: Totți

Nume	Prenume	Titlu Job	Status	Profil
Ion	Popescu	Specialist Marketing Digital	acceptat	<a href="#">Profil</a>
Dumitrescu	Elena	Specialist Marketing Digital	acceptat	<a href="#">Profil</a>
George	Mihai	Data Analyst	respins	<a href="#">Profil</a>
Tudor	Gabriel	Data Analyst	acceptat	<a href="#">Profil</a>
Tai	Mikhail	Financial Analyst	in curs	<a href="#">Profil</a>
Kramnik	Vladimir	Data Scientist	in curs	<a href="#">Profil</a>

De asemenea, observăm că putem să filtrăm intrările din tabela aplicanților, aceasta este realizată prin porțiunea aceasta de cod

```
<form method="get" action="">
  <input type="hidden" name="nume" value="<?php echo htmlspecialchars(string: $nume_companie); ?>">
  <label for="filtru">Filtru status:</label>
  <select name="filtru" id="filtru" onchange="this.form.submit()">
    <option value="">Toti</option>
    <?php
    $res = $conn->query(query: "SELECT id_status, denumire FROM status_aplicare");
    while ($row = $res->fetch_assoc()) {
      $selected = (isset($_GET['filtru']) && $_GET['filtru'] == $row['id_status']) ? 'selected' : '';
      echo "<option value='" . $row['id_status'] . "' $selected" . htmlspecialchars(string: $row['denumire']) . "</option>";
    }
  </select>
</form>
```

Acest formular este folosit pentru a filtra aplicațiile în funcție de statusul lor, adică dacă sunt acceptate, în curs sau respinse. Este util pentru companii atunci când vor să vadă doar un anumit tip de aplicații, fără să le parcurgă pe toate.

Formularul trimite datele prin metoda GET și conține un câmp ascuns care reține numele companiei curente. Apoi, printr-un select, sunt afișate toate statusurile posibile, luate direct din tabela status\_aplicare. Fiecare opțiune are asociat un ID de status și o denumire, cum ar fi „Acceptat” sau „Respins”.

Atunci când utilizatorul alege un status din listă, formularul se trimite automat cu ajutorul funcției `onchange="this.form.submit()"`. Dacă nu este selectat niciun filtru, se afișează toate aplicațiile. În caz contrar, se afișează doar cele care au statusul ales.

Codul mai verifică și dacă utilizatorul selectase deja un filtru anterior, iar dacă da, marchează opțiunea ca fiind selectată, pentru ca aceasta să rămână vizibilă la reîncărcarea paginii.

În concluzie, acest mecanism ajută la o vizualizare mai clară și mai organizată a aplicațiilor pentru un anumit job, în funcție de statusul lor actual.

## Pagina aplicanților – applicant.php

Pentru a ajunge pe această pagină trebuie să apăsăm pe numele celui care a aplicat în ultimul tabel de pe pagina cu companii.

Această pagină afișează detalii despre un anumit applicant, identificat prin id, primit prin URL (metoda GET). Dacă id este trimis, aplicația face o interogare în baza de date pentru a obține datele personale ale applicantului, precum numele, prenumele, emailul, CNP-ul, telefonul, orașul și un link către CV-ul acestuia, dacă există.

Codul arată așa:

```
<?php
if (isset($_GET['id'])) {
    $id_aplicant = $_GET['id'];

    $stmt = $conn->prepare(query: "
        SELECT a.num, a.prenume, a.email, a.CNP, a.telefon, ad.oras, a.cv_link
        FROM aplicanti a
        JOIN adresa ad ON a.id_adresa = ad.id_adresa
        WHERE a.id_aplicant = ?");
    $stmt->bind_param(types: "i", var: &$id_aplicant);
    $stmt->execute();
    $stmt->bind_result(var: &$num, vars: &$prenume, $email, $cnp, $telefon, $oras, $cv_link);
    $stmt->fetch();
    $stmt->close();
}

<div align="center">
    <h1><?php echo "$num $prenume"; ?></h1>
    <p><strong>Email:</strong> <?php echo htmlspecialchars(string: $email); ?></p>
    <?php if ($cnp) echo "<p><strong>CNP:</strong> " . htmlspecialchars(string: $cnp) . "</p>"; ?>
    <?php if ($telefon) echo "<p><strong>Telefon:</strong> " . htmlspecialchars(string: $telefon) . "</p>"; ?>
    <?php if ($oras) echo "<p><strong>Oraș:</strong> " . htmlspecialchars(string: $oras) . "</p>"; ?>
    <?php if ($cv_link) echo "<p><strong>CV:</strong> <a href='" . htmlspecialchars(string: $cv_link) . "' target='_blank'>Deschide CV</a></p>";
</div>
```

Se poate vedea că salvăm rezultatele în niște variabile

		<b>Ion Popescu</b>	
		Email: ion.popescu@mail.com	
		CNP: 5040809297249	
		Telefon: 0756789012	
		Oraș: București	
		CV: <a href="#">Deschide CV</a>	

### Aplicații ale applicantului

Filtru după status: Totale

Job	Data aplicare	Status
Specialist Marketing Digital	2025-05-01	acceptat
Content Writer	2025-05-24	acceptat

### Competențe

Nume competență	Nivel	Ani experiență	Studii	Certificare
PHP	Avansat	5	facultate	nu
Vânzări	Avansat	9	master	da

Tabelul aplicari este locul unde se înregistrează fiecare aplicație făcută de un candidat pentru un anumit job. Practic, de fiecare dată când un applicant aplică la un job, se creează o linie nouă în

acest tabel. Fiecare aplicație este legată atât de candidatul care a aplicat, cât și de jobul vizat, și are un status care arată în ce stadiu se află aplicația — de exemplu, dacă este încă în curs, a fost acceptată sau a fost respinsă. Se reține și data la care s-a făcut aplicația, ca să se poată urmări evoluția în timp.

Pe de altă parte, tabelul status\_aplicare este folosit pentru a defini statusurile posibile ale unei aplicații. Nu se scrie direct „acceptat” sau „respins” în tabelul aplicari, ci se salvează un cod numeric care face legătura cu denumirea din acest tabel. În felul ăsta, toate denumirile de status sunt ținute într-un singur loc, lucru care face mai ușor de gestionat aplicațiile și previne greșelile de scriere sau inconsecvențele.

Împreună, cele două tabele permit urmărirea clară a aplicațiilor făcute de candidați și oferă flexibilitate în filtrarea sau afișarea lor în funcție de statusul curent.

```
<?php
    $filtru = isset($_GET['filtru']) ? $_GET['filtru'] : '';

    if ($filtru != "") {
        $stmt = $conn->prepare(query: "
            SELECT j.titlu, a.data_aplicare, sa.denumire
            FROM aplicari a
            JOIN joburi j ON a.id_job = j.id_job
            JOIN status_aplicare sa ON a.id_status = sa.id_status
            WHERE a.id_aplicant = ? AND a.id_status = ?");
        $stmt->bind_param(types: "ii", var: &$id_aplicant, vars: &$filtru);
    } else {
        $stmt = $conn->prepare(query: "
            SELECT j.titlu, a.data_aplicare, sa.denumire
            FROM aplicari a
            JOIN joburi j ON a.id_job = j.id_job
            JOIN status_aplicare sa ON a.id_status = sa.id_status
            WHERE a.id_aplicant = ?");
        $stmt->bind_param(types: "i", var: &$id_aplicant);
    }

    $stmt->execute();
    $result = $stmt->get_result();
    while ($row = $result->fetch_assoc()) {
        echo "<tr>
            <td>" . htmlspecialchars(string: $row['titlu']) . "</td>
            <td>" . htmlspecialchars(string: $row['data_aplicare']) . "</td>
            <td>" . htmlspecialchars(string: $row['denumire']) . "</td>
        </tr>";
    }
    $stmt->close();
?>
```

Al doilea tabel este cel care afișează competențele aplicantului. Aici sunt listate toate abilitățile pe care le-a introdus în sistem și care pot fi legate de experiența sa profesională. Pentru fiecare competență este afișat numele, nivelul declarat (de exemplu începător, mediu sau avansat), numărul de ani de experiență, nivelul de studii necesar și dacă are sau nu o certificare pentru acea competență. Toate aceste date vin din mai multe tabele legate între ele și sunt prezentate într-un tabel clar, tocmai pentru ca firma care analizează candidatul să poată înțelege rapid cât de pregătit este și pe ce domenii. Dacă nu există competențe asociate aceluia aplicant, atunci tabelul rămâne gol.

```
<div class="container" style="padding: 20px;">
  <h2>Competențe</h2>
  <table border="1" cellpadding="10">
    <tr>
      <th>Nume competență</th>
      <th>Nivel</th>
      <th>Ani experiență</th>
      <th>Studii</th>
      <th>Certificare</th>
    </tr>
  </table>
</div>

<?php
$stmt = $conn->prepare(query: "
  SELECT c.nume_competenta, nc.denumire AS nivel, c.ani_experienta,
         ns.denumire AS studii, ca.certificare
  FROM competente_aplicanti ca
  JOIN competente c ON ca.id_competenta = c.id_competenta
  JOIN nivel_competenta nc ON c.id_nivel_competenta = nc.id_nivel_competenta
  JOIN nivel_studii ns ON c.id_nivel_studii = ns.id_nivel_studii
  WHERE ca.id_aplicant = ?");
$stmt->bind_param(types: "i", var: &$id_aplicant);
$stmt->execute();
$result = $stmt->get_result();

while ($row = $result->fetch_assoc()) {
  $certificare = ($row['certificare'] == 1) ? 'da' : 'nu';
  echo "<tr>
    <td>" . htmlspecialchars(string: $row['nume_competenta']) . "</td>
    <td>" . htmlspecialchars(string: $row['nivel']) . "</td>
    <td>" . htmlspecialchars(string: $row['ani_experienta']) . "</td>
    <td>" . htmlspecialchars(string: $row['studii']) . "</td>
    <td>" . $certificare . "</td>
  </tr>";
}
$stmt->close();
```

Codul acesta afișează într-un tabel toate competențele pe care le are un aplicant. Mai întâi apare titlul „Competențe”, apoi un tabel cu mai multe coloane: numele competenței, nivelul ei (de

exemplu „Avansat”), câți ani de experiență are aplicantul pe acel domeniu, ce studii sunt necesare pentru competența respectivă și dacă are sau nu o certificare.

Informațiile sunt luate din baza de date, din mai multe tabele legate între ele. Se folosește id\_aplicant ca să se știe ale cui competențe să fie afișate. După ce sunt extrase, datele sunt puse rând cu rând în tabel, iar dacă la o competență scrie că are certificare (cu valoarea 1), în tabel se afișează „da”, altfel apare „nu”.

Totul e afișat într-un format clar, ca să poți vedea dintr-o privire ce știe să facă acel candidat.

## Concluzii și observații finale

Pot spune că realizarea acestui proiect mi-a adus o mulțime de cunoștințe noi, în special legate de filtrarea și manipularea eficientă a datelor dintr-o bază de date. Am învățat să construiesc o aplicație web funcțională, care să ofere o interfață intuitivă pentru utilizatori. Nu doar implementarea efectivă a proiectului a fost utilă, ci și redactarea documentației, care m-a ajutat să înțeleg mai bine cum funcționează totul „în spatele scenei”.

Pe parcurs, m-am confruntat cu diverse probleme, precum erori de conexiune sau dificultăți în afișarea datelor, însă am reușit să le depășesc și să îndeplinesc toate cerințele proiectului. Acesta este un proiect pe care îl voi păstra ca exemplu de bună practică.

De asemenea, am aprofundat concepte importante legate de normalizarea bazei de date, ceea ce mă va ajuta să construiesc baze de date eficiente și fără redundanțe în viitor. Am învățat și despre importanța securității în accesarea datelor, cum ar fi folosirea interogărilor parametrizate pentru a evita injecțiile SQL.

Pe viitor, voi putea aplica aceste cunoștințe în alte proiecte similare sau în medii de baze de date diferite, consolidându-mi astfel abilitățile de dezvoltator web și administrator de baze de date.

\*De notat: am schimbat numele din SGBD în SGBD\_BABEANU\_GEORGE\_VALENTIN pentru a evita eventualele confuzii.