LUCIAN MARIUS RUSU

# INFORMATICĂ APLICATĂ


Editura POLITEHNICA

Colecţia "INFORMATICA"

---

# INFORMATICĂ APLICATĂ

Materialul are ca scop prezentarea noțiunilor de bază despre calculatoare, rețele de calculatoare și gestionarea bazelor de date. Noțiunile prezentate în acest volum sunt de mare interes pentru studenții din domeniul ingineresc, dar și pentru toți cei care doresc să se familiarizeze cu aceste noțiuni. Materialul, bogat în conținut, reprezintă o sursă de documentare foarte utilă, pentru cei care doresc să dezvolte aplicații cu interfețe grafice utilizator în mediul Matlab.

**Referent ştiinţific:** Conf. dr. ing. Adriana ALBU

Cartea oferă un ghid pentru realizarea unei aplicaţii informatice în Matlab pornind de la exemplificarea şi descrierea elementelor constitutive, culminând cu asamblarea acestora într-un tot unitar funcţional. Volumul realizează o descriere amplă a domeniului informaticii şi a posibilităţilor aplicaţiilor sale practice în ceea ce priveşte utilizarea bazelor de date prin intermediul sistemului de gestionare MySQL. Prin explicarea conceptelor de bază, prezentarea tehnologiilor şi aplicaţiilor specifice, precum şi discutarea tendinţelor şi provocărilor actuale, autorul ne ghidează în înţelegerea conceperii unei astfel de aplicaţii.

**Referent ştiinţific:** Conf. dr. ing. Karoly MENYHARDT

LUCIAN MARIUS RUSU

# INFORMATICĂ APLICATĂ

Colecţia "INFORMATICA"

EDITURA POLITEHNICA
TIMIŞOARA – 2023

**EDITURA POLITEHNICA**
Bd. Republicii nr. 9
300159 Timişoara, România

**Tel.** 0256.403.822
**E-mail:** editura@upt.ro

**Redactor:** Claudia MIHALI

**ISBN** 978-606-35-0535-5

# CUPRINS

# PREFAȚĂ

Informatica se referă la utilizarea calculatoarelor și a tehnologiilor informatice pentru a rezolva anumite probleme. Informatica a pătruns în toate domeniile, și a creat noi posibilități de dezvoltare. La momentul actual calculatorul este disponibil tuturor, iar datorită rețelei de internet accesul la informație este foarte facil.

Acest volum își propune să realizeze o introducere în fascinanta lume a informatici prezentând noțiuni de bază despre calculatoare, rețele de calculatoare, și gestionarea bazelor de date.

Cartea este structurată pe patru capitole. În primul capitol este prezentată o scurtă istorie a calculatoarelor. Este prezentată evoluția calculatoarelor evidențiind momentele cheie, precum și structura unui calculator, descriind componentele hardware și software.

Capitolul al doilea prezintă rețele de calculatoare. În acest capitol se face referire la modul de comunicare între calculatoare și la dispozitivele utilizate pentru crearea unui rețele informatice.

În cel de al treilea capitol este exemplificat modul de gestionare a bazelor de date. Este prezentat modul și regulile de creare a unei baze de date relaționale în MySQL și comenzile de bază în limbajul SQL.

Cel de al patrulea capitol prezintă etapele și codul sursă pentru realizarea unor interfețe grafice utilizator dezvoltate în programul Matlab pentru gestionarea unei baze de date MySQL.

Această carte se adresează studenților din învățământul tehnic, care doresc să dobândească noțiunile de bază despre calculatoare, rețele de calculatoare și baze de date.

Doresc să mulțumesc doamnei conf.dr.ing. Adriana ALBU și domnului conf.dr.ing. Karoly MENYHARDT pentru sprijinul acordat, precum și tuturor celor care m-au susținut și au făcut posibilă realizarea acestei cărți.

# 1 INTRODUCERE

În momentul de față tehnica de calcul se regăsește în toate domeniile. Istoria sistemelor de calcul începe odată cu apariția primelor dispozitive mecanice utilizate pentru calcul, cum ar fi abacul sau dispozitivele de numărat mecanice, care datează din antichitate și au fost utilizate în principal pentru efectuarea de operații simple de numărare.

În secolul al XVII-lea, matematicianul scoțian John Napier a dezvoltat logaritmii, care au făcut posibilă efectuarea operațiilor matematice complexe cu ajutorul unei simple adunări sau scăderi a logaritmilor. Aceasta a dus la dezvoltarea primelor dispozitive mecanice de calcul, cum ar fi riglele de calcul și calculatoarele mecanice.

În secolul al XIX-lea, Charles Babbage, un matematician și inventator englez, a dezvoltat o serie de proiecte pentru calculatoare mecanice programabile, cum ar fi Mașina Analitică (Fig. 1.1) și Mașina Diferențială (Fig. 1.2), care au fost primele dispozitive concepute pentru a fi programate pentru a efectua operațiuni matematice complexe[1].

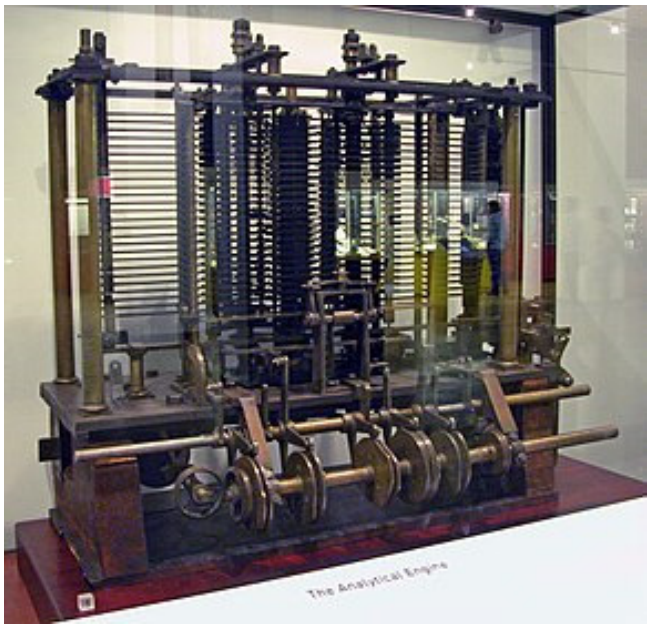



Fig. 1.1. Modelul unei părți a motorului analitic expus la muzeul de știință din Londra

Mașina diferențială este formată din mai multe coloane, numerotate de la 1 la n. Mașina poate stoca o valoare zecimală în fiecare coloană. Singura operație pe care o poate face este să adune valoarea unei coloane n+1 cu cea a coloanei n

pentru a produce noua valoare a coloanei n. Coloana N poate doar să stocheze o constantă, coloana 1 afișează valoarea calculului de la iterația curentă[2].

Mașina se programează prin setarea valorilor inițiale ale coloanelor. Coloana 1 se setează la valoarea polinomului la începutul calculelor. Coloana 2 se setează la valoarea obținută din prima și a doua derivată a polinomului la aceeași valoare a lui X. Fiecare dintre coloanele de la 3 la N se setează la o valoare calculată din primele derivate ale polinomului. [2]

Fig. 1.2. A treia mașină diferențială a lui Per Georg Scheutz

În anii 1930 și 1940, odată cu dezvoltarea tehnologiei electronice, au fost dezvoltate primele calculatoare electrice, cum ar fi Atanasoff-Berry Computer (Fig. 1.3) și Harvard Mark I (Fig. 1.4), care au utilizat componente electronice precum tuburile electronice pentru a efectua calcule[3,4].

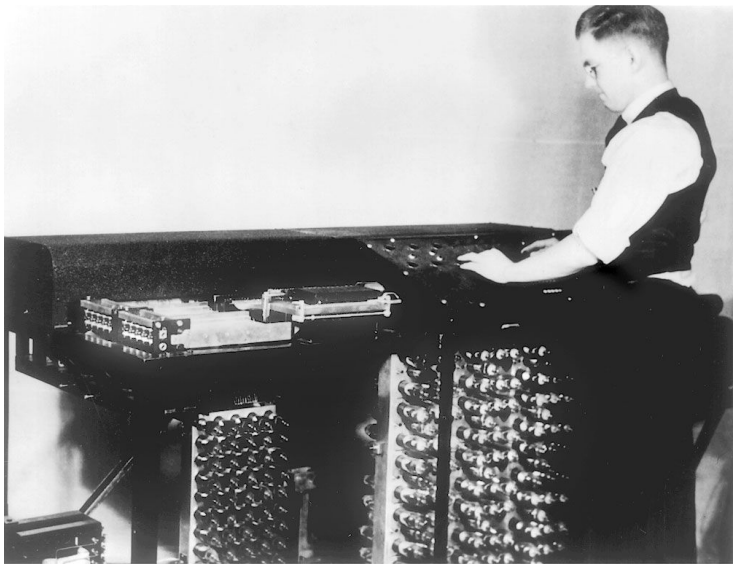


Fig. 1.3. Atanasoff-Berry Computer

În anii 1950, dezvoltarea computerelor a continuat cu dezvoltarea primelor computere comerciale, cum ar fi IBM 701 și UNIVAC I, care au fost utilizate pentru a efectua calcule complexe și pentru a gestiona operațiunile de afaceri.

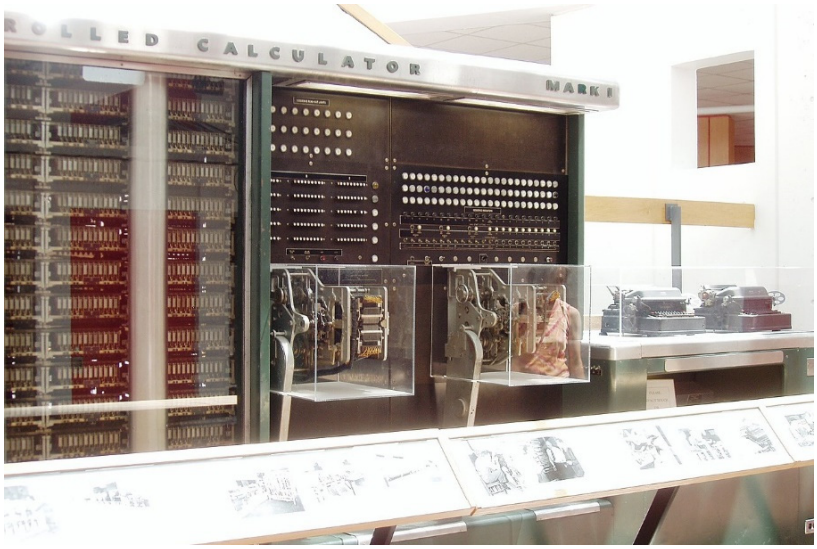



Fig. 1.4. Harvard Mark I

Apariția tuburilor electronice (lămpi – cum sunt uzual numite) (Fig. 1.5) a determinat și apariția primei generații de calculatoare electronice. Aceste dispozitive erau foarte mari, scumpe, generau multă căldură și efectuau doar calcule simple.


Fig. 1.5. Tuburi electronice [5]

Dezvoltarea electronicii a condus odată cu apariția tranzistoarelor (Fig. 1.6) și la apariția celei de a doua generații de calculatoare. Astfel aceste dispozitive aveau dimensiuni semnificativ mai mici decât generația anterioară. De asemenea puteau efectua calcule mai complexe și de asemenea prețul acestor dispozitive a scăzut.

Fig. 1.6. Tranzistoare [6]

Cea de a treia generație de calculatoare este caracterizată de utilizarea circuitelor integrate (Fig. 7). Aceste circuite integrate sunt componente electronice de dimensiuni mici care au crescut performanțele calculatoarelor.

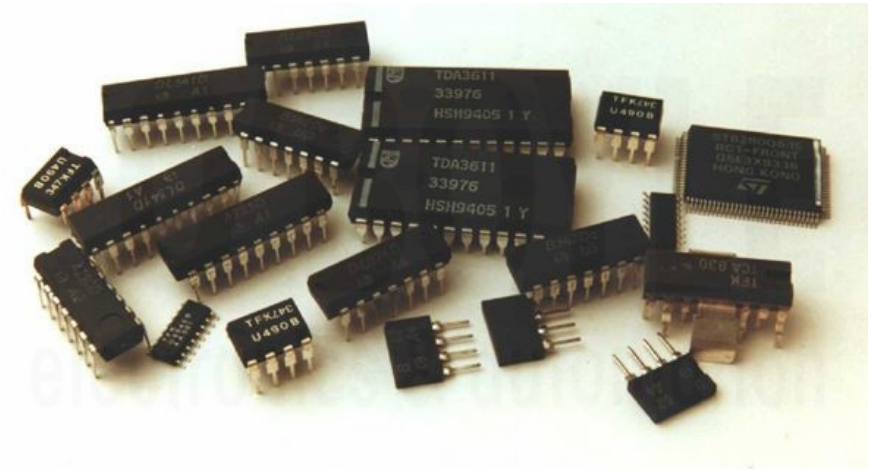



Fig. 1.7. Circuite integrate [7]

Odată cu apariția microprocesoarelor (Fig. 1.8) a apărut generația a patra de calculatoare. Microprocesorul a mărit capacitatea de procesare și a redus prețul de producere a calculatoarelor, acestea devenind accesibile publicului larg.

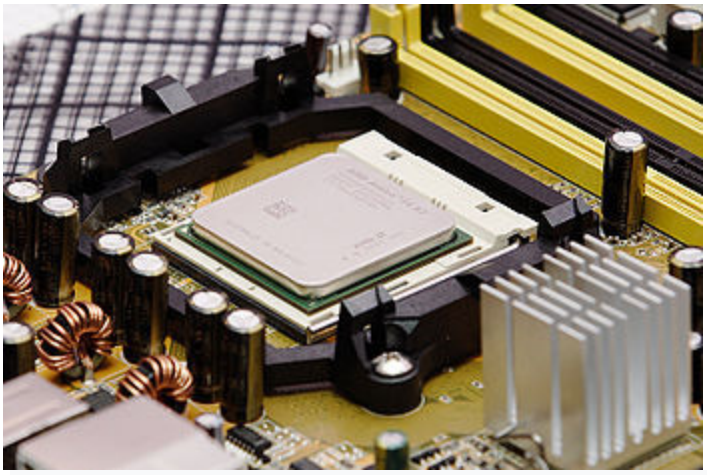


Fig. 1.8. Procesor AMD Athlon 64 X2 conectat pe soclu la placa de bază [8]

Generația acesta de calculatoare bazată pe microprocesoare este cea pe care o utilizăm astăzi. Știința evoluează în continuare iar o nouă generație de calculatoare se preconizează a apărea odată cu dezvoltarea inteligentei artificiale și a fizicii cuantice.

Calculatorul este format din două componente majore:

- Componenta hardware prin care înțelegem toate componentele fizice;
- Componenta software prin care înțelegem totalitatea programelor.

Aceste două componente se găsesc într-o strânsă legătură deoarece niciuna nu poate funcționa independent.

Un calculator este compus din unitatea centrala (CPU – Central Processing Unit) și periferice (echipamentele, dispozitivele conectate la unitatea centrală).

Unitatea centrală este compusă din următoarele componente:

- Carcasă – aceasta susține toate componentele unității centrale;
- Placa de bază – aceasta reprezintă componenta electronică care conectează toate componentele calculatorului și permite comunicarea între acestea.
- Procesorul – este un microprocesor care reprezintă creierul calculatorului și care procesează toate informațiile;
- Memorie RAM – este memoria calculatorului care utilizată pentru stocarea temporară a informației și pentru efectuarea calculelor;
- Hard Disk – utilizat pentru stocarea datelor;
- Sursă de alimentare – alimentează la o anumită tensiune fiecare componentă electronică a unității centrale.

Carcase sunt de mai multe feluri și dimensiuni (Fig. 1.9). Ele adăpostesc toate componentele unității centrale.

a. Carcasa verticală DELL [11]

b. Carcasa orizontală DELL [12]

Fig.1.9. Carcasa unității centrale

Placa de bază (Fig. 1.10) este componenta care interconectează toate componentele și permite conectarea diferitelor periferice. Placa de bază este alimentată de la sursă. Pe placa de bază se află procesorul, are conectori SATA pentru conectarea hard disc-urilor, conexiuni pentru memoria RAM, conectori PCI și PCI EXPRESS pentru diferite placi video, de achiziție, precum și o serie de porturi pentru conectarea diferitelor periferice.

Cele mai uzuale porturi (Fig. 1.11) care se găsesc pe placa de baza sunt:

- PS2 – pentru conectarea tastaturi și a mouse-ului;
- VGA – ieșire video;
- DVI – ieșire video;
- HDMI – ieșire video;
- DP – ieșire video;
- USB – port intrare / ieșire;
- RJ-45 – pentru conexiunea la o rețea de calculatoare;
- Jacks – pentru conexiunea audio;
- RS232 – port serial;
- DB25 – port paralel.

Porturile PS2 dedicate conectări tastaturi și mouse-ului au fost treptat eliminate de pe placa de bază fiind înlocuite de porturile USB. De asemenea pentru reducerea dimensiunilor calculatoarelor s-a renunțat la majoritate porturilor de ieșire video păstrându-l doar pe cel mai performat și mai des utilizat HDMI. Portul RJ-45 pentru conexiunea fizică la o rețea de calculatoare s-a păstrat pentru majoritatea plăcilor de bază, chiar dacă majoritatea dețin și o antenă pentru conectarea wireless. Porturile seriale (RS232), paralele (DB25) și audio au fost treptat înlocuite de porturile USB. Astfel porturile USB (Universal Serial BUS) au devenit o interfață standard de conectare a perifericelor la un calculator.

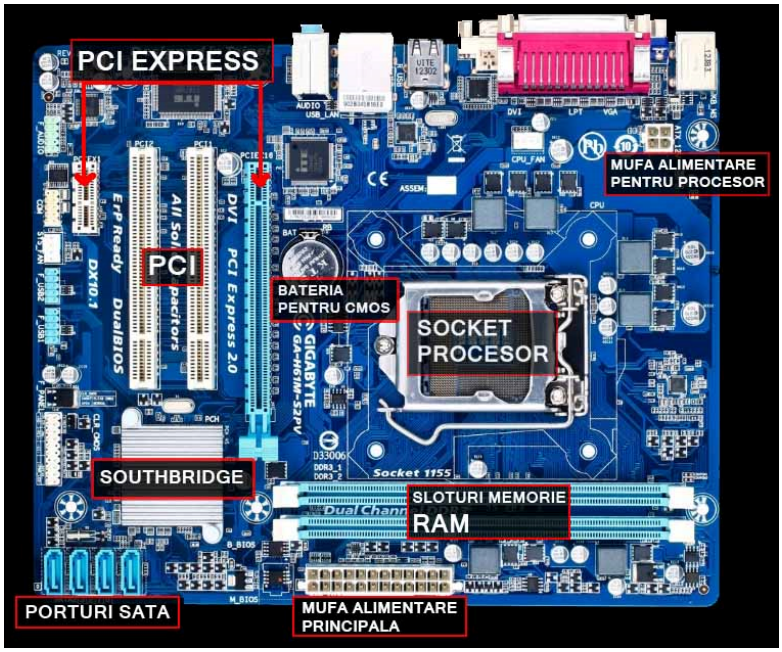



Fig. 1.10. Placă de baza [13]

Fig. 1.11. Poturi placă de bază [14]

Procesorul este responsabil pentru efectuarea calculelor și a altor operații. Procesorul este o componentă electronică care efectuează diferite operații pe baza datelor stocate în memorie. Pentru o performanță ridicată, majoritatea procesoarelor actuale conțin nuclee multiple.

Memoria RAM (Random Access Memory) (Fig. 1.12) este o memorie temporară care stochează informația ce este procesată. În general performanța procesorului și capacitatea de memorare definesc performanțele unui calculator.

Mai există și memoria ROM (Read Only Memory) ce reprezintă un spațiu de stocare nevolatil pentru informații necesare despre componentele unității centrale. În general memoria ROM este utilizată pentru stocarea BIOS-ului (Basic Input / Output System) care este primul software ce rulează la pornire. Bios-ul face legătura între componentele hardware ale sistemului.


Fig. 1.12. Memorie RAM [15]

Hard disk-ul reprezintă componenta care stochează toate informațiile și ne permite accesul la acestea ori de câte ori este nevoie. Există două tipuri de hard disk-uri:

- HDD – dispozitiv electromagnetic de stocare;
- SSD – dispozitiv de stocare ce utilizează o memorie flash nevolatilă.

Hard disk-urile de tip HDD (Fig. 1.13) stochează informația pe discuri magnetice (platane) cu ajutorul unui cap de scriere / citire, acestea fiind în mișcare. Pe când cele de tip SSD (Solid State Drive) (Fig. 1.14) sunt de fapt memorii flash care nu au componente în mișcare, acest lucru reflectându-se mai ales printr-o viteză ce acces a informației mult mai mare.

Fig. 1.13. Hard disk - HDD [16]

Sursa de alimentare este acea componentă care alimentează cu energie toate componentele unității centrale. Ea este alimentată de la rețeaua electrică și convertește energia funcție de necesitate fiecărei componente. Sursa de alimentare trebuie să fie suficient de puternică pentru a putea oferi fiecărei componente energia necesară unei bune funcționări.


Fig. 1.14. Hard disk – SSD [17]

Pentru ca un calculator să poată funcționa, pe lângă unitatea centrală, mai are nevoie de următoarele periferice: monitor, tastatură și mouse. Aceste periferice sunt necesare pentru a putea utilizatorul sa interacționeze cu calculatorul adică să îi ofere informații utilizând tastatura și mouse-ul, respectiv să primească informația procesată de acesta prin intermediul monitorului.

Alte periferice care pot fi conectate la calculator sunt:

- Periferice de intrare (care transmit informație calculatorului)
  - Tastatură;
  - Mouse;
  - Scaner;
  - Microfon;
  - Cameră video;
  - CD/DVD;
- Periferice de ieșire (care permit afișarea informației de la calculator)
  - Monitor;
  - Videoproiector;
  - Imprimantă;
  - Boxe (sistem audio);
  - Diferite ecrane led, TV
- Periferice de intrare și ieșire
  - Ecran tactil;
  - Card de memorie;
  - USB drive;
  - Multifuncționale;
  - Modem.

Pentru gestionarea tuturor acestor componente hardware este necesară componenta software care permite intercomunicare între acestea. Componenta

software a unui calculator se referă la toate programele, aplicațiile, driverele și sistemul de operare care rulează pe calculator și care permit utilizatorilor să își efectueze activitățile zilnice.

Sistemul de operare este interfața dintre utilizator și calculator. Cu ajutorul acestuia, utilizatorul poate interacționa cu calculatorul. Driverele sunt programe care permit sistemului de operare să comunice cu componentele hardware. Aplicațiile sunt diverse programe care pun la dispoziția utilizatorului diferite funcții care facilitează munca acestuia.

În 1950, Laboratorul de Cercetare General Motors a realizat primul sistem de operare pentru calculatorul lor, IBM 701 [9]. Acesta rula un sistem de procesare în serii, deoarece programele și datele erau introduse pe rând. Au mai existat diferite sisteme de operare create de diverse companii dedicate pentru echipamentele produse de acestea, dar primul sistem de operare compatibil cu mai multe echipamente și care a fost utilizat la scara largă a fost MS-DOS (MicroSoft Disk Operating System). Acesta este un sistem de operare în linie de comandă. Windows 1.0 este primul mediu de operare cu interfață grafică creat de Microsoft și lansat pe 20 noiembrie 1985[10].

În momentul actual există o mare diversitate de sisteme de operare. Ce le mai cunoscute sisteme de operare existente pe piață astăzi sunt:

- Sistemul de operare Windows, dezvoltat de Microsoft, este unul dintre cele mai utilizate sisteme de operare pentru computere personale. Windows este cunoscut pentru interfața sa grafică prietenoasă, versatilitate și flexibilitate.
- Sistemul de operare Linux, este un sistem de operare open-source, ceea ce înseamnă că codul său sursă este disponibil publicului pentru a fi modificat și îmbunătățit. Linux este cunoscut pentru stabilitatea sa, securitatea ridicată și flexibilitatea sa, putând fi instalat și utilizat pe o varietate largă de dispozitive, de la calculatoare personale la servere.
- Sistemul de operare Mac OS, dezvoltat de Apple, este sistemul de operare utilizat de calculatoarele Mac. Mac OS este cunoscut pentru design-ul său elegant, securitatea ridicată și ușurința de utilizare.
- Sistemul de operare Android, dezvoltat de Google, este un sistem de operare pentru dispozitive mobile, cum ar fi telefoanele și tabletele. Android este cunoscut pentru flexibilitatea sa, suportul pentru o varietate largă de aplicații și integrarea sa cu serviciile Google.
- Sistemul de operare iOS, dezvoltat de Apple, este sistemul de operare utilizat de dispozitivele mobile iPhone și iPad. iOS este cunoscut pentru securitatea sa ridicată, ușurința de utilizare și integrarea strânsă cu ecosistemul Apple.

Pe lângă sistemul de operare mai sunt necesare și drivere-le. Acestea reprezintă programe furnizate de producătorul componentei hardware care permite sistemului de operare să interacționeze cu componenta hardware. Aceste drivere sunt dedicate fiecărei componente hardware pentru un anume sistem de operare. Drivere-le sunt gratuite și se pot descărca de obicei de pe pagina oficială a producătorului. La momentul actual majoritatea componentelor hardware sunt de tip plug & play, adică în momentul conectări acestora la unitatea centrală, sistemul de operare sesizează acest lucru și pe baza informațiilor primite de la componenta hardware instalează un driver generic sau dacă sistemul are acces la internet, sistemul de operare se conectează la pagina producătorului, download-ând și instalând driver-ul dedicat.

Calculatoare tradiționale (desktop), chiar dacă dimensiunea acestora s-a redus substanțial, sunt destul de greu mutat. Odată cu dezvoltarea tehnologiei s-au realizat și sisteme de calcul portabile cum ar fi laptop-urile și tabletele.

Conform dicționarului explicativ al limbii române, informatica reprezintă știința care se ocupă cu studiul prelucrării informației cu ajutorul sistemelor automate de calcul. Astfel pentru o mai ușoară utilizare a calculatoarelor s-au dezvoltat diferite aplicații care pun la dispoziția utilizatorului funcții specifice unui anumit domeniu.

Aceste aplicații pot fi;

- aplicații de calcul matematic (MathCad, MatLab);
- aplicații de programare (python, C++, java);
- aplicații de proiectare (SolidWorks, Catia);
- aplicații de simulare (Ansys, Abaqus);
- aplicații de achiziție de date (LabView);
- aplicații divertisment (jocuri video);
- etc.

Aplicațiile au fost dezvoltate pentru a facilita utilizarea componentelor hardware de către utilizatori. Utilizând aceste aplicații utilizatorii trebuie să introducă parametrii și condițiile inițiale apoi aplicația pe baza algoritmilor integrați șa conveargă la o solutie.

# 2 REȚELE DE CALCULATOARE

Datorită creșterii rapide a numărului de calculatoare personale, a apărut necesitatea cuminicării și partajării informațiilor între acestea.

Departamentul Apărării din Statele Unite ale Americii (DARPA) a început, prin ani 1960, să dezvolte un sistem de comunicare între computere, numit ARPANET. Acesta a fost proiectat pentru a permite oamenilor de știință din întreaga lume să colaboreze într-un mod facil și eficient. Acest sistem de rețea de calculatoare a funcționat până în 1990 și a pus bazele pentru dezvoltarea internetului și a rețelelor de calculatoare moderne.

O rețea informatică este un grup de unități de calcul interconectate prin cabluri electrice, fibre optice, raze laser, unde radio, care pot accesa simultan un resurse software și/sau hardware.

Avantajele utilizării rețelelor informatice sunt:

- Partajarea resurselor pe arii geografice deosebit de întinse (servere de fișiere, baze de date, sisteme periferice);
- Utilizarea unor aplicații puternice care rulează pe servere mari (aplicații de baze de date, sisteme de geo poziționare, sisteme de telecomunicații, aplicații științifice);
- Accesul la informație devine planetar;
- Utilizarea unor servicii specifice rețelelor de calculatoare (email, ftp, www, irc, chat, VoIP, e-shoping).

Din punct de vedre al topologiei există două tipuri de rețele:

- de difuzare;
- punct la punct(PtP sau P2P).

Rețelele de calculatoare de difuzare sunt acele rețele care permit transmiterea unui flux de date către un număr mare de dispozitive, fără a necesita o conexiune directă între fiecare dispozitiv. În cadrul unui astfel de tip de rețea, datele sunt transmise de la o singură sursă la toate dispozitivele conectate la rețea. O astfel de rețea poate fi utilă pentru furnizarea de informații în timp real, cum ar fi știri sau informații meteorologice, la un publicul larg.

Există mai multe tehnologii folosite în rețelele de calculatoare de difuzare, printre care se numără:

- **Sateliți de comunicare:** Acești sateliți sunt plasați pe orbită și transmit semnale radio către receptorii de pe pământ. Aceste semnale pot fi recepționate de un număr mare de dispozitive conectate la rețea
- **Rețele de cablu:** Acestea sunt rețele care transmit semnalul pe cablu. Acestea pot fi folosite pentru a transmite informații către un număr mare de dispozitive conectate la rețea (ex: TV sau internet);

- **Rețele de telefonie mobilă:** Acestea sunt rețele mobile care permit transmisia de date către dispozitive mobile (telefoanele mobile sau tabletele). Aceste rețele pot fi folosite pentru a transmite informații în timp real către un număr mare de dispozitive mobile (ex: RoAlert).

Rețelele de calculatoare de difuzare sunt importante în furnizarea de informații la un număr mare de utilizatori, fără a fi nevoie de o conexiune directă între fiecare dispozitiv.

Rețelele de difuzare utilizează un singur canal de comunicație care este partajat de toate mașinile din rețea. Comunicația se realizează prin intermediul unor mesaje scurte, numite pachete, care au în structura lor, printre altele, un câmp pentru desemnarea expeditorului și unul pentru desemnarea destinatarului. Se pot trimite pachete către toate mașinile din rețea, acest mod de operare numindu-se difuzare

Rețelele de calculatoare punct la punct sunt rețele care permit comunicarea directă între două sau mai multe dispozitive prin intermediul unei conexiuni dedicate. În acest tip de rețea, datele sunt transmise direct de la un dispozitiv la altul, fără a fi necesar să treacă prin un punct central. Dispun de numeroase conexiuni între perechi de mașini ce formează rețeaua. Pentru a ajunge la destinație, un pachet de date trebuie sa treacă prin mai multe mașini intermediare, fiind nevoie de algoritmi pentru dirijarea pachetelor pe un drum optim. Este un model folosit pentru rețelele mari.

Rețelele de calculatoare punct la punct sunt foarte populare în mediile de afaceri și sunt utilizate pentru a permite angajaților să comunice între ei sau să acceseze resursele de pe rețea, cum ar fi fișierele sau imprimantele.

Există mai multe tipuri de topologii a rețelelor de calculatoare dintre care precizăm următoarele tipuri:

- stea;
- inel;
- arbore;
- completă;
- magistrală;
- neregulată.

Într-o topologie de tip stea (Fig. 2.1), calculatoarele sunt conectate la un nod central numit hub. Aceasta topologie se folosește în rețelele concentrate, unde fiecare calculator poate fi conectat la un nod central. Fiecare calculator al unei rețele de tip stea comunică cu hub-ul central, care retransmite mesajele unui alt calculator, fie tuturor celorlalte calculatoare.[18]

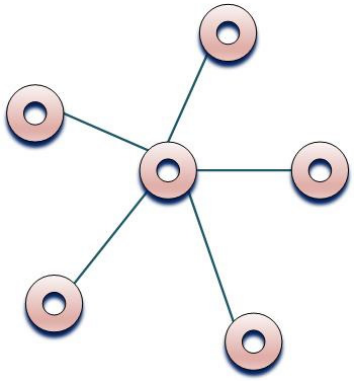


Fig.2.1. Topologie tip stea

Într-o rețea de calculatoare cu topologie de tip inel (Fig. 2.2), fiecare calculator este conectat la următorul, iar ultimul este conectat la primul. Topologia de tip inel este utilizată în rețelele de mare performanta, pentru a reduce întârzierile de timp dacă informația este accesată de un număr mare de utilizatori.

Fiecare calculator este conectat la următorul calculator aflat în inel și fiecare retransmite următorului calculator ceea ce a primit de la precedentul. Mesajele traversează inelul într-o singura direcție. Deoarece fiecare calculator retransmite ceea ce primește, nu se pune problema pierderii pachetelor. De asemenea, nu există terminații ale rețelei, deoarece nu exista un capăt al acesteia.[19]

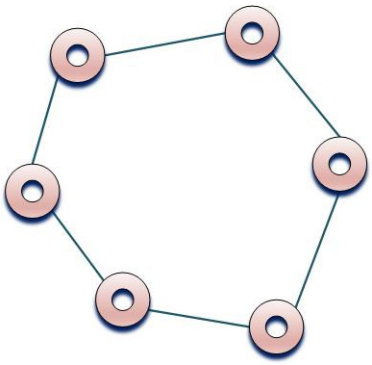


Fig. 2.2. Topologie tip inel

Rețelele de calculatoare de tip arbore (Fig. 2.3) există o singură cale între fiecare pereche de noduri, dar între nodurile nepereche pot exista mai multe căi. Toate informațiile necesare pentru funcționarea algoritmului de dirijare sunt conținute în adresa nodului de destinație [20].

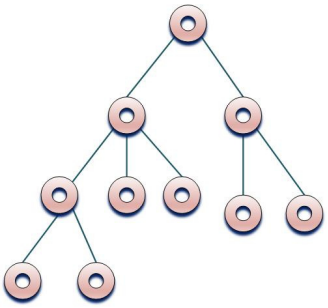


Fig. 2.3. Topologie tip arbore

Topologia completă (Fig. 2.4) presupune o conectivitate totală între noduri, fiind folosită atunci când se cer rețele de fiabilitate ridicată [20]. Într-o astfel de rețea de calculatoare fiecare calculator este conectat direct cu toate celelalte calculatoare ale rețelei.

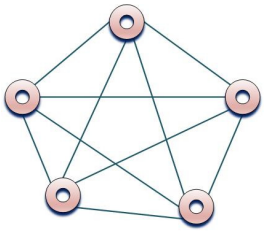


Fig. 2.4. Topologie tip completă

Această topologie de tip magistrală (Fig. 2.5) este utilizată în general atunci când viitoarea rețea este simplă, mică sau temporară. Într-o rețea de tip magistrală, cablarea presupune unul sau mai multe cabluri, fără nici un dispozitiv activ de amplificare a semnalului de-a lungul cablului. Din acest motiv, topologia de tip magistrală este pasivă. Atunci când un calculator transmite un semnal toate celelalte calculatoarele îl primesc, dar numai celui căruia îi este destinat mesajul îl va accepta [21].

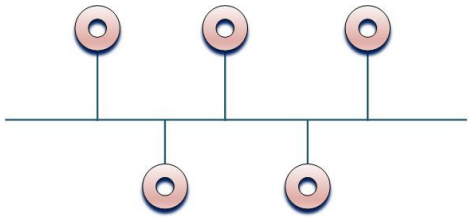


Fig. 2.5. Topologie tip magistrală

În cadrul rețelelor de calculatoare cu topologie neregulată (Fig. 2.6), calculatoarele sunt legate între ele într-un mod aparent aleatoriu dar care respectă diverse reguli funcție de proiectarea rețelei. Topologia neregulată este frecventă în cazul rețelelor răspândite pe o arie largă [20].

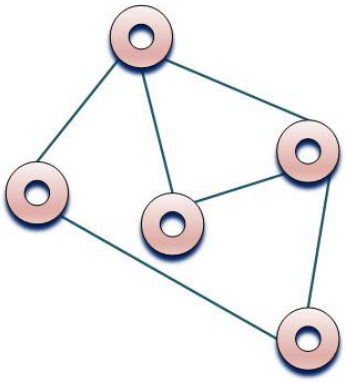


Fig. 2.6. Topologie tip neregulată

Numărul de calculatoare care se regăsesc într-o rețea și aria pe care se extinde rețeaua a condus la o clasificarea acestora astfel:

- rețele locale (LAN);
- rețele metropolitane (MAN);
- rețele răspândite pe arii geografice mari (WAN).

Rețelele locale (Local Area Networks), sunt rețele private localizate într-o singură clădire sau într-un campus de cel mult câțiva kilometri. Ele sunt frecvent utilizate pentru a conecta calculatoarele personale și/sau stațiile de lucru din companii, instituții, în scopul de a partaja informații și resurse [22].

Rețelele locale utilizează o tehnologie de transmisie care constă dintr-un singur cablu la care sunt atașate toate mașinile și funcționează la viteze cuprinse între 10 și 1000 Mbps, au întârzieri mici și produc erori puține [22].

O rețea metropolitană (Metropolitan Area Network) deservește un oraș. Cel mai bun exemplu de rețea metropolitană este rețeaua de televiziune prin cablu. La începuturi aceste rețele erau exclusiv dedicate recepției de televiziune. Din momentul în care Internet-ul a început să atragă audiența de masă, operatorii de rețele de cablu TV au realizat că, dacă vor face anumite schimbări în sistem, ar putea să ofere servicii bidirecționale de internet în părțile nefolosite ale spectrului [22].

O rețea larg răspândită geografic (Wide Area Network) acoperă o arie geografică întinsă (țară, continent) În majoritatea rețelelor de acest tip există mai multe subrețele, care la rândul lor sunt formate din două componente distincte: liniile de transmisie și elementele de comutare. Liniile de transmisie transportă pachetele (biții) între mașini iar elementele de comutare sunt calculatoare specializate, folosite pentru a conecta două sau mai multe linii de transmisie. Când sosesc date pe o anumită linie, elementul de comutare trebuie să aleagă o nouă linie pentru a retransmite datele mai departe [22].

## 2.1 Arhitectura unei rețele informatice

Calculatoarele existente într-o rețea trebuie să comunice între ele pentru putea partaja informația. Astfel o rețea informatică este organizată pe nivele ierarhice alcătuite pe principiul că fiecare nivel primește servicii de la nivelul inferior și oferă servicii nivelului superior pe fluxuri distincte de recepție (Rx) și transmitere (Tx).

Regulile de comunicare între nivele constituie protocolul de comunicare (IPX/SPX, TCP/IP, PCTCP, etc). Protocolul de comunicare este implementat într-o interfață care asigura transferul de date între nivele.

Exista trei tipuri de arhitecturi de rețele informatice:

- OSI (Open System Interconnection);
- DOD sau TCP/IP (Department of Defence);
- Microsoft.

Arhitectura OSI este o arhitectură de rețea standardizată, dezvoltată de Organizația Internațională pentru Standardizare (ISO). Această arhitectură definește un model de referință cu șapte nivele, fiecare cu propriile funcționalități, care facilitează comunicația între dispozitivele de pe rețea. Fiecare nivel are propriul său set de protocoale și standarde [23].

Cele șapte nivele ale unei rețele informatice de tip OSI sunt (Fig. 2.7)[23]:

- **Nivelul fizic:** Acesta este primul nivel al modelului OSI și se ocupă de transferul datelor prin mijloace fizice, cum ar fi cablurile, semnalele electrice sau luminoase.
- **Nivelul de legare de date:** Acesta se ocupă de transferul datelor între dispozitivele vecine și de detectarea și corectarea erorilor.
- **Nivelul rețea:** Acesta se ocupă de gestionarea rutării și transferului datelor între subrețele diferite.
- **Nivelul transport:** Acesta se ocupă de asigurarea unei comunicări fiabile și eficiente între dispozitivele de pe rețea, prin controlul fluxului și gestionarea erorilor.
- **Nivelul sesiune:** Acesta se ocupă de stabilirea, menținerea și întreruperea conexiunilor între dispozitivele de pe rețea.
- **Nivelul prezentare:** Acesta se ocupă de gestionarea și transformarea datelor în formatele necesare pentru a fi transmise pe rețea.
- **Nivelul aplicație:** Acesta se ocupă de furnizarea de servicii și aplicații pentru utilizatorii de pe rețea.

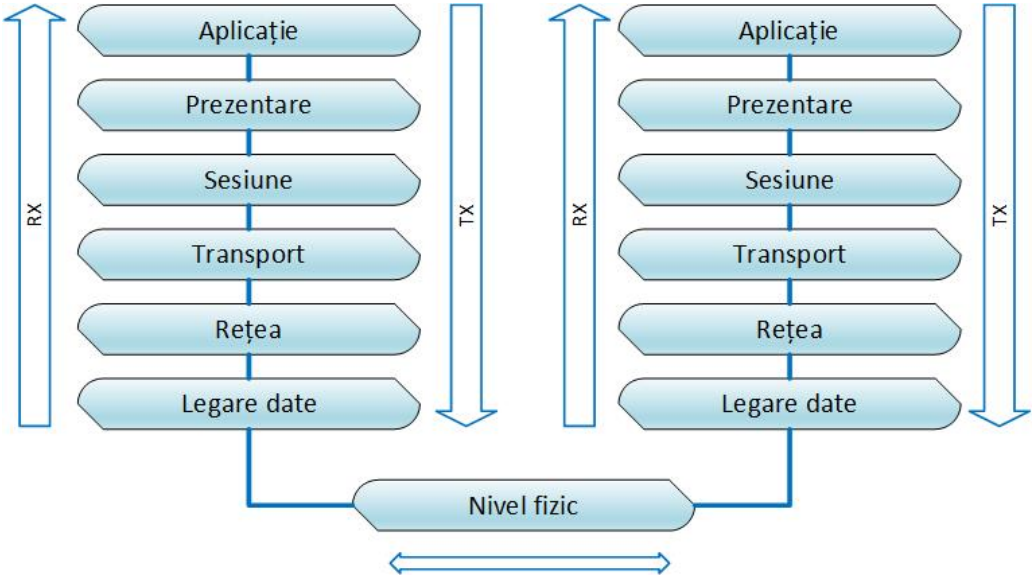



Fig. 2.7. Arhitectura OSI

Nivelul fizic are rolul de a transmite datele de la un calculator la altul prin intermediul unui mediu de comunicație. Datele sunt văzute la acest nivel ca un sir de biți. Problemele tipice sunt de natura electrica: nivelele de tensiune corespunzătoare unui bit 1 sau 0, durata impulsurilor de tensiune, cum se inițiază și cum se oprește transmiterea semnalelor electrice, asigurarea păstrării formei semnalului propagat. Mediul de comunicație nu face parte din nivelul fizic.

Nivelul legăturii de date corectează erorile de transmitere apărute la nivelul fizic, realizând o comunicare corectă între două noduri adiacente ale rețelei. Mecanismul utilizat în acest scop este împărțirea biților în cadre ( frame), cărora le sunt adăugate informații de control. Cadrele sunt transmise individual, putând fi verificate și confirmate de către receptor. Alte funcții ale nivelului se refera la fluxul de date (astfel încât transmițătorul sa nu furnizeze date mai rapid decât le poate accepta receptorul) și la gestiunea legăturii (stabilirea conexiunii, controlul schimbului de date și desființarea conexiunii).

Nivelul rețea asigură dirijarea datelor între nodurile sursă și destinație, trecând eventual prin noduri intermediare (routing). Este foarte important ca fluxul de date să fie astfel dirijat încât să se evite aglomerarea anumitor zone ale rețelei. Interconectarea rețelelor cu arhitecturi diferite este o funcție a nivelului rețea.

Nivelul transport realizează o conexiune între două calculatoare gazdă detectând și corectând erorile pe care nivelul rețea nu le tratează. Este nivelul aflat in mijlocul ierarhiei, asigurând nivelelor superioare o interfața independentă de tipul rețelei utilizate. Funcțiile principale sunt: stabilirea unei conexiuni sigure

între două maşini gazdă, iniţierea transferului, controlul fluxului de date si închiderea conexiunii.

Nivelul sesiune stabileşte şi întreţine conexiuni (sesiuni) între procesele aplicaţie, rolul sau fiind acela de a permite proceselor sa stabilească "de comun acord" caracteristicile dialogului şi să sincronizeze acest dialog.

Nivelul prezentare realizează operaţii de transformare a datelor în formate înţelese de entităţile ce intervin într-o conexiune. Transferul de date între maşini de tipuri diferite (Unix-DOS, de exemplu) necesită şi codificarea datelor în funcţie de caracteristicile acestora. Nivelul prezentare ar trebui să ofere şi servicii de criptare/decriptare a datelor, în vederea asigurării securităţii comunicaţiei in reţea.

Nivelul aplicaţie are rolul de "fereastra" de comunicaţie între utilizatori, aceştia fiind reprezentaţi de entităţile aplicaţie (programele). Nivelul aplicaţie nu comunica cu aplicaţiile ci controlează mediul în care se executa aplicaţiile, punând-le la dispoziţie servicii de comunicaţie [22].

Printre funcţiile nivelului aplicaţie se afla:

- identificarea partenerilor de comunicaţie, determinarea disponibilităţii acestora şi autentificarea lor;
- sincronizarea aplicaţiilor cooperante şi selectarea modului de dialog;
- stabilirea responsabilităţilor pentru tratarea erorilor;
- identificarea constrângerilor asupra reprezentării datelor;
- transferul informaţiei.

Arhitectura Microsoft a unei reţele de calculatoare se bazează pe soluţii software şi hardware dezvoltate de compania Microsoft. Această arhitectură utilizează protocoale şi servicii specifice ale sistemelor de operare Windows, precum Active Directory, DNS, DHCP şi altele.

Microsoft oferă soluţii integrate pentru gestionarea reţelelor de calculatoare, care utilizează Active Directory ca sistem de autentificare şi autorizare centralizat. Acest sistem permite administrarea centralizată a utilizatorilor, dispozitivelor şi politicilor de securitate în cadrul unei reţele de calculatoare.

Arhitectura DOD (Department of Defense) este o arhitectură de reţea dezvoltată de Departamentul de Apărare al Statelor Unite ale Americii. Această arhitectură este cunoscută şi sub numele de arhitectura TCP/IP, datorită protocoalelor de comunicare utilizate. Arhitectura DOD se bazează pe un model simplu de patru nivele, fiecare cu propriile funcţionalităţi [22].

Nivelele arhitecturi DOD sunt (Fig. 2.8)[22]:

- **Nivelul fizic:** Acesta se ocupă de furnizarea de conectivitate fizică la reţea, cum ar fi cablurile, interfeţele şi dispozitivele de conectare.

- **Nivelul rețea:** Acesta se ocupă de rutarea pachetelor de date prin intermediul rețelei, utilizând adresele IP.
- **Nivelul transport:** Acesta se ocupă de gestionarea conexiunilor și a transferului datelor între dispozitivele de pe rețea, utilizând protocoale precum TCP sau UDP.
- **Nivelul aplicație:** Acesta se ocupă de furnizarea de servicii și aplicații pentru utilizatorii de pe rețea, cum ar fi email, web, FTP etc.

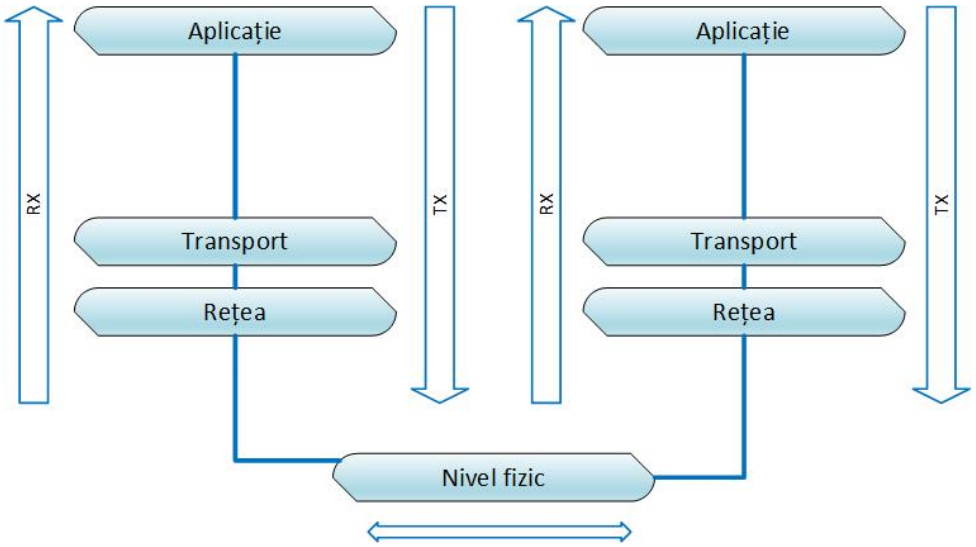



Fig. 2.8. Arhitectura DOD

La nivelul de rețea, protocolul dominant este protocolul IP (Internet Protocol) care se ocupă de rutarea mesajelor pe trasee optime și de gestionarea mesajelor ICMP sau IGMP. De asemenea stabilește regulile de adresare în rețea (adresele IP).

În cadrul nivelul de transport, protocolul dominant este protocolul TCP și UDP care asigură transmiterea secvențială a pachetelor de date și verificarea erorilor. Asigură gestiunea porturilor de comunicație și a serviciilor specifice.

La nivelul aplicație sunt mai multe protocoale implicate, legate de accesul la diverse servicii în rețea. Asigura gestiunea sesiunilor de lucru și securitatea rețelei bazată pe regulile și drepturile de acces. Convertește forma datelor din pachetele de date spre formate ușor accesibile. Asigura serviciile de care are nevoie o aplicație ca să poată rula (porturi, conexiuni, servere)

Arhitectura DOD a fost dezvoltată inițial pentru a asigura comunicațiile între dispozitivele militare ale Statelor Unite, însă, datorită standardizării sale și a adaptabilității la diferite tehnologii, aceasta a devenit o arhitectură de rețea foarte utilizată la nivel global, de către organizații, companii și utilizatori individuali.

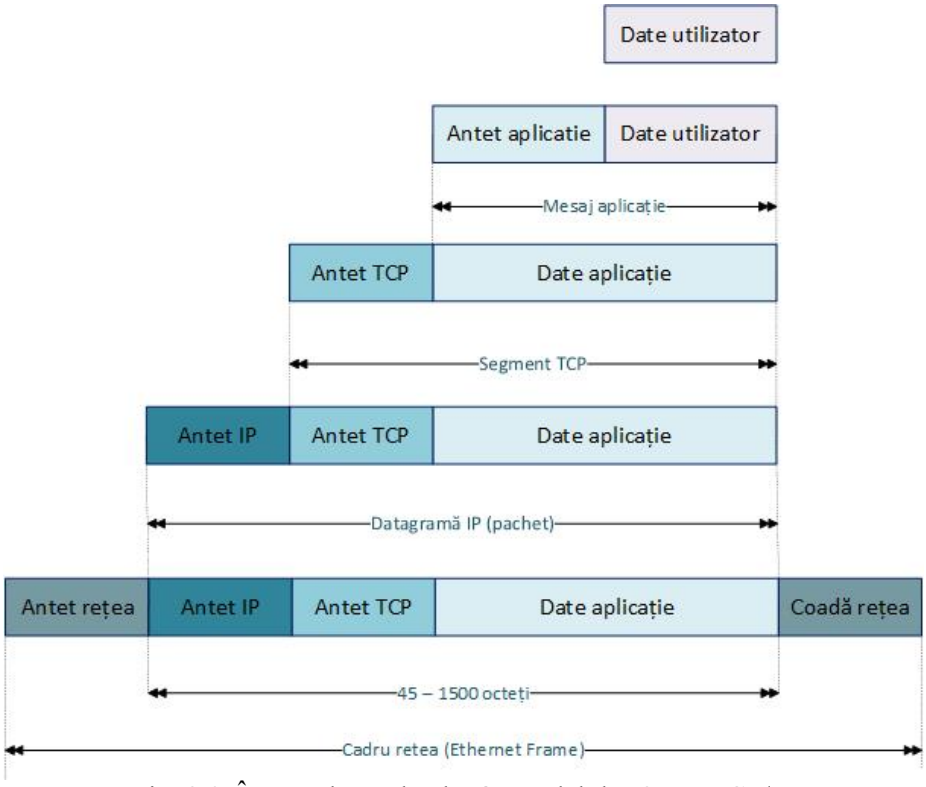



Fig. 2.9. Încapsularea datelor în modelul DOD – TCP/IP

Încapsularea datelor (Fig. 2.9) este procedeul prin care datele sunt transferate de la un nivel la altul al rețelei sub forma de pachete. Pe sensul de transmitere a datelor (Tx) pachetele primesc un antet la fiecare nivel, iar pe sensul de recepție a datelor (Rx) pachetele pierd antetul corespunzător fiecărui nivel (decapsulare).

## 2.2 Protocoale utilizate în rețelele de calculatoare

Protocoalele utilizate pentru fiecare nivel în modelul DOD sunt:

- Aplicatie
  - DNS – Domain Name System
  - FTP – File Transfer Protocol
  - SNMP – Simple Network Management Protocol
  - SMTP – Simple Mail Transfer Protocol
  - Telnet
  - SSH – Secure Shell
  - DHCP – Dynamic Host Configuration Protocol
  - WWW – World Wide Web
- Transport

  - TCP – Transmission Control Protocol
  - UDP – User Datagram Protocol
- Rețea
  - IP – Internet Protocol
  - ICMP – Internet Control Message Protocol
  - IGMP – Internet Group Management Protocol
- Legaturi de date
  - ARP – Adress Resolution Protocol
  - PPP – Point to Point Protocol

Protocolul DNS este un sistem distribuit de nume utilizat pentru identificarea calculatoarelor din Internet sau din alte rețele pe bază de Internet Protocol (IP). Fiecare implementare TCP/IP conține o rutină software specializată în interogarea serverului de nume în vederea obținerii translatării nume/adresă IP sau invers [24].

Protocolul FTP este un set de reguli utilizat pentru accesul la fișiere aflate pe servere din rețele de calculatoare particulare sau din Internet.

Protocolul SNMP este un protocol de gestionare utilizat pe scară largă pentru comunicarea cu dispozitive de rețea precum router, switch, hub, servere, etc. pe o rețea IP. Acesta prevede schimbul de rețea și informații între elementele de rețea utilizând porturile UDP 161 și 162 [25].

Simple Mail Transfer Protocol este un protocol simplu folosit la transmiterea mesajelor în format electronic în rețea de calculatoare. Protocolul SMTP folosește portul 25 TCP și determină adresa unui server SMTP pe baza înregistrării MX (Mail eXchange) din configurația serverului DNS [26].

Telnet este un protocol de rețea care se folosește la comunicația textuală, bidirecțională și interactivă, bazată pe realizarea unei conexiuni virtuale cu stația de lucru destinatară[22].

Protocolul SSH este asemănător protocolului telnet cu mențiunea că mesajele text transmise sunt criptate pentru o siguranță mărită.

Protocolul DHCP automatizează alocarea parametrilor de rețea la dispozitive de către unul sau mai multe servere DHCP. Chiar și în rețele mici, protocolul DHCP este folositor, deoarece simplifica adăugarea unor noi echipamente în rețea [27].

WWW este numai unul din numeroasele servicii și aplicații informatice disponibile în Internet. Hypertext Transfer Protocol (HTTP) este metoda cea mai des utilizată pentru accesarea informațiilor în Internet care sunt păstrate pe servere World Wide Web.

Protocolul TCP este folosit de obicei de aplicații care au nevoie de confirmare de primire a datelor. Efectuează o conectare virtuală între două puncte

terminale, fiecare punct fiind definit de către o adresă IP și de către un port TCP [28].

UDP este un protocol de comunicație pentru calculatoare ce aparține nivelului transport al arhitecturii standard OSI. Împreună cu IP, acesta face posibilă livrarea mesajelor într-o rețea. Spre deosebire de protocolul TCP, UDP constituie modul de comunicație fără conexiune. Adică pachetele de informații sunt trimise în general fără confirmare de primire, în speranța că ele vor ajunge, fără a exista o legătură efectivă între expeditor și destinatar [29].

Internet Protocol este un protocol prin care datele sunt trimise de la un calculator la altul prin intermediu Internetului. Fiecare calculator are cel puțin o adresă IP unică, care îl identifică între toate calculatoarele din rețea.

ICMP este un protocol din suita TCP/IP care folosește la semnalizarea și diagnosticarea problemelor din rețea.

Protocolul IGMP este utilizat în rețelele IP pentru gestionarea și controlul traficului multicast. Traficul multicast este utilizat pentru a transmite informații de la un singur emițător la mai mulți receptorii simultan, ceea ce face posibilă distribuirea de conținut multimedia, transmisii în direct[22].

Protocolul ARP este utilizat pentru a identifica adresele fizice ale dispozitivelor de rețea (adresele MAC) pornind de la adresele IP. ARP permite dispozitivelor de rețea să identifice adresele fizice ale altor dispozitive de pe aceeași rețea locală, astfel încât acestea să poată comunica în mod optim [22].

PPP este un protocol de rețea utilizat pentru a stabili conexiuni între două dispozitive de rețea, cum ar fi un computer și un router, sau două routere. PPP este utilizat adesea pentru a oferi acces la internet prin intermediul distribuitorului de internet.

Fiecare calculator din rețea are o adresă IP primită de la serverul DHCP sau setată manual. Această adresă îl identifică unic în rețea.

IPv4 este o versiune a protocolului IP (Internet Protocol) utilizată pentru a identifica și comunica între dispozitivele conectate la internet și alte rețele. IPv4 este prima și cea mai răspândită versiune a protocolului IP. IPv4 utilizează adrese IP pentru a identifica dispozitivele de rețea și a le permite să comunice între ele. Adresele IPv4 sunt formate din 32 de biți, împărțiți în patru octeți (grupuri de 8 biți), fiecare reprezentând o valoare între 0 și 255. Astfel, există aproximativ 4,3 miliarde de adrese IPv4 unice posibile. IPv4 a fost standardul principal pentru comunicațiile de rețea timp de mai multe decenii, dar este înlocuit în mod treptat de către IPv6, datorită numărul limitat de adrese IPv4 disponibile [22].

IPv6 este o versiune mai nouă a protocolului IP utilizată pentru a identifica și comunica între dispozitivele conectate la internet și alte rețele. IPv6 este o versiune mai nouă și mai avansată decât IPv4, care a fost dezvoltată pentru a rezolva limitările și problemele de securitate ale IPv4. Rețelele IPv6, au

adresele IP sunt formate din 128 de biți organizate in 8 grupe de cate 4 caractere hexazecimale, ceea ce permite o multitudine de adrese IP unice (aprox $(16^4)^8$=3,4x$10^{38}$ adrese IPv6 unice).

Există două categorii de adrese IP:

- **adrese publice** – care permite dispozitivului să fie vizibil, respectiv sa comunice pe internet;
- **adrese private** – care permite dispozitivului să fie vizibil doar în cadrul rețelei locale.

Un IP public este o adresă IP unică, atribuită de furnizorul de servicii internet unei rețele sau dispozitiv de rețea, care permite acestuia să comunice cu alte dispozitive de pe internet. Adresa IP publică este, de fapt, adresa unei rețele sau echipament de pe internet. Fiecare echipament care se conectează la internet are nevoie de o adresă IP publică unică pentru a putea comunica cu alte dispozitive de pe internet.

O adresă IP privat este atribuită unei rețele sau echipament de către un router sau server DHCP local. Adresele IP private sunt utilizate pentru a identifica echipamentele conectate la o rețea locală dar nu pot fi utilizate pentru a comunica direct cu dispozitive de pe internet. Adresele IP private sunt de obicei utilizate pentru a atribui adrese IP dispozitivelor din rețeaua locală și pentru a permite acestora să comunice între ele în cadrul rețelei locale[22].

Adresele IPv4, care sunt cele mai răspândite la momentul actual, sunt grupate în patru clase:

- **Clasa A** conține adrese între 1.0.0.0 și 126.0.0.0 adică 126 de rețele cu 2563 calculatoare fiecare; folosește 8 biți pentru rețea și 24 de biții pentru calculatoare; se folosește la rețele mari, cu multe dispozitive;
- **Clasa B** conține adrese între 128.1.0.0 și 191.254.0.0 adică 16320 rețele cu 2562 calculatoare fiecare; folosește 16 biții pentru rețea și 16 biții pentru calculatoare;
- **Clasa C** conține adrese între 192.0.1.0 și 223.255.254.0 adică aproximativ 2 milioane de rețele cu 256 de calculatoare fiecare; folosește 24 de biții pentru rețea și 8 biții pentru calculatoare; se folosește pentru rețele mici, de tip LAN;
- **Clasa D** conține adrese între 224.0.0.0 și 254.0.0.0 care sunt rezervate pentru aplicații speciale și nu fac parte din domeniul public.

Adresele IP private sunt grupe de adrese care nu se rutează în Internet și sunt folosite în rețele locale aflate in spatele unor echipamente care au firewall. Aceste adrese sunt dirijate (rutate) doar în interiorul subrețelei și nu în afara ei. Aceste adrese sunt:

- clasa 10.0.0.0 cu 2563 calculatoare;

- clasa 172.16.0.0 pana la 172.31.0.0 cu 16 rețele de cate 2562 calculatoare fiecare;
- clasa 192.168.0.0 cu 2562 calculatoare.

Un avantaj al utilizării adreselor IP private este faptul că acestea sunt mai sigure decât adresele IP publice, deoarece dispozitivele conectate la o rețea privată nu sunt conectate direct la internet, ele fiind protejate de către firewall-ul rețelei. Utilizând tehnica NAT (Network Address Translation ) avem posibilitatea ca dispozitivelor de pe o rețea privată să comunice cu dispozitive de pe internet.

## 2.3 Echipamente utilizate în cadrul rețelelor informatice

Există o varietate de echipamente utilizate în cadrul rețelelor informatice. Fiecare echipament are un rolul specific în funcție de topologia rețelei și de nevoile utilizatorilor. În continuare se vor prezenta câteva dintre cele mai uzuale echipamente folosite în cadrul rețelelor informatice:

În primul rând rețelele informatice au fost concepute pentru a conecta computerele între ele. Astfel pentru a conecta în rețea orice calculator sau dispozitiv portabil (laptop, tableta, telefon mobil, etc) acesta trebuie să dețină o placă de rețea (Fig. 2.10). Modul de conectare la rețea al plăcii putând fi prin cablu sau wireless.

Fig. 2.10. Placă de rețea [30]

Echipamentele care permit conectarea dispozitivelor în rețea sunt:

- router;
- switch;
- media convertor;
- firewall;
- modem;
- acces point;
- server;
- repeter;

Router-ul (Fig. 2.11) este un dispozitiv care interconectează mai multe dispozitive sau rețele de calculatoare. Acesta îndeplinește funcții importante precum rutarea traficului, filtrarea pachetelor și asigurarea securității rețelei.


Fig. 2.11. Router D-Link[31]

La majoritatea dispozitivelor de tip router se pot conecta echipamente atât prin cablu cât și fără fir. Aceste echipamente permit crearea unei rețele locale deoarece dispun de un server DHCP pentru gestionarea IP-urilor private ce vor fi alocate dispozitivelor ce se conectează la acesta.

Switch-ul (Fig. 2.12) este un dispozitiv utilizat pentru a interconecta mai multe dispozitive de rețea într-o rețea locală. Acesta transferă datele între dispozitive utilizând adrese MAC și poate fi administrat sau neadministrat.

Switch-urile neadministrate interconectează dispozitivele dar nu alocă IP-uri acestora, iar cele administrate oferă de obicei și servicii de DHCP și Firewall.


Fig. 2.12. Switch TP-Link[32]

Media convertor-ul (Fig. 2.13), este un dispozitiv utilizat pentru a converti semnalele de la un tip de mediu de transmisie la altul. Acesta poate fi utilizat pentru a conecta două dispozitive care folosesc medii de transmisie diferite, cum ar fi cablul de cupru și fibra optică. Un astfel de dispozitiv este necesar deoarece majoritate distribuitorilor de internet oferă, la momentul actual,

conectarea clienților la internet prin fibră optică pentru o viteză superioară. Deoarece majoritatea dispozitivelor utilizate permit conectarea la rețea doar prin cablu sau wireless este necesară utilizarea unui media convertor (lucru valabil și pentru semnalul TV nu doar de internet).


Fig. 2.13. Media convertor[33]

La momentul actual pe piață există și dispozitive mixte care combină de exemplu media convertorul cu router-ul pentru a avea un singur echipament. Aceste dispozitive preiau semnalul optic și îl convertesc în semnal electric care apoi este partajat prin cablu sau WiFi cu dispozitivele din subrețea datorită serviciilor oferite de router-ul integrat.

Firewall-ul (Fig. 2.14) este un dispozitiv hardware sau software utilizat pentru a proteja rețeaua de acces neautorizat și atacuri cibernetice. Acesta poate bloca traficul nedorit și poate permite accesul doar utilizatorilor autorizați.

Firewall-ul controlează accesul la resursele disponibile în rețea. Pentru a crea o rețea cât mai sigură posibil, majoritatea dispozitivelor utilizate în cadrul rețelelor pun la dispoziție un firewall de tip software unde pot fi setate o serie de reguli cu privire la modul și porturile de accesare a dispozitivelor.


Fig. 2.14. Firewall ZYXEL [34]

Modem-ul (fig. 29) este un dispozitiv care transformă semnalele digitale în semnale analogice și invers, permițând astfel conectarea la internet prin linii telefonice sau cablu.

Aceste echipamente pot fi conectate la o rețea de telefonie fixă (Fig. 2.15a) sau la o rețea de telefonie mobilă (Fig. 2.15b). Aceste echipamente sunt

similare cu media convertorul deoarece au rolul de a converti un semnal, doar că în acest caz nu se schimbă tipul semnalului.


a)


b)

Fig. 2.15. Modem [35,36]

Access Point (AP) 9 (Fig. 2.16)este un dispozitiv utilizat pentru a permite conectarea dispozitivelor wireless la rețeaua locală. Aceste dispozitive pot fi comparate cu o antenă sau cu un router wireless care comunică doar cu dispozitive care permit conectarea fără fir. Avantajul utilizării dispozitivelor de tip acces point este acela că o subrețea poate fi gestionată de mai multe dispozitive AP astfel mărind zona activă a rețelei. Un dispozitiv portabil va selecta echipamentul AP care oferă cel mai bun semnal, iar în momentul în care utilizatorul își schimbă locația și alt echipament AP oferă semnal mai bun, dispozitivul portabil se va conecta la acesta fără a întrerupe comunicarea.

De asemenea toate dispozitivele AP care creează o rețea locală sunt gestionate dintr-o singură interfață (Fig. 2.17). Se poate observa că în cadrul acestei subrețele sunt disponibile trei echipamente de tip AP care fiecare este conectat la rețea prin cablu.

Fig. 2.16. Acces point Ubiquiti[37]

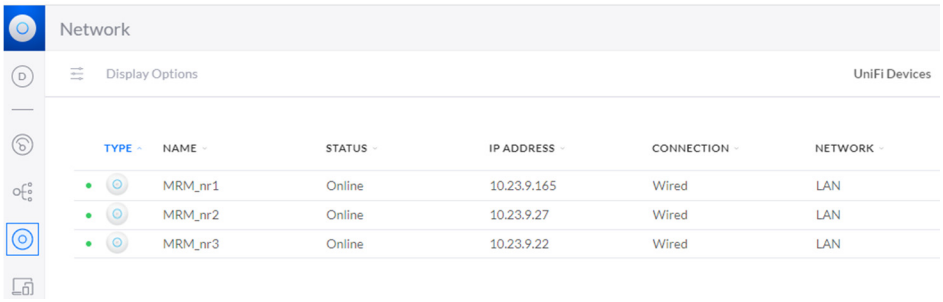



Fig. 2.17. Interfață dispozitive Acces Point

Cu toate că sunt trei dispozitive AP în cadrul subrețelei, se observă că există doar o singură rețea WiFi (Fig. 2.18) care este disponibilă pe toate cele trei dispozitive.

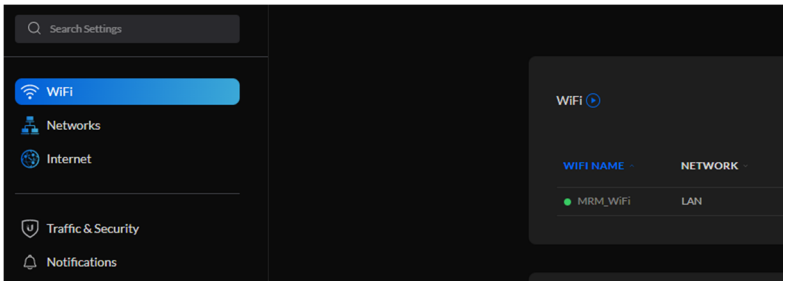



Fig. 2.18. Setare WiFi Acces Point

Server-ul (Fig. 2.19) este un dispozitiv utilizat pentru a stoca și gestiona date și resurse în cadrul rețelei. Serverul poate fi utilizat pentru a găzdui site-uri web, pentru a gestiona bazele de date sau pentru a stoca fișierele partajate în cadrul rețelei.

a) b)

Fig. 2.19. Server [38,39]

Un server poate fi un calculator pe care este instalat un sistem de operare tip server, sau un dispozitiv dedicat (Fig. 2.19 a) care arată și se comportă ca un calculator dedicat. Pentru serverele care gestionează rețele mari este nevoie de

resurse hardware suplimentare, astfel aceste dispozitive putând ajunge la dimensiuni impresionante (fig. 2.19 b).

Repeater-ul (Fig. 2.20) este un dispozitiv utilizat pentru a extinde distanța de transmisie a semnalului într-o rețea.

Fig. 2.20. Repeter[40]

Un astfel de dispozitiv este adesea utilizat dacă semnalul WiFi nu acoperă aria dorită sau dacă datorită structurii imobilului semnalul nu este suficient de puternic. Un astfel de dispozitiv preia semnalul WiFi, îl amplifică apoi îl transmite mai departe.

Acestea sunt doar câteva dintre echipamentele utilizate în cadrul rețelelor informatice, existând și altele precum bridge-uri, gateway-uri, proxy-uri și multe altele, care pot fi utilizate în funcție de nevoile specifice ale rețelei.

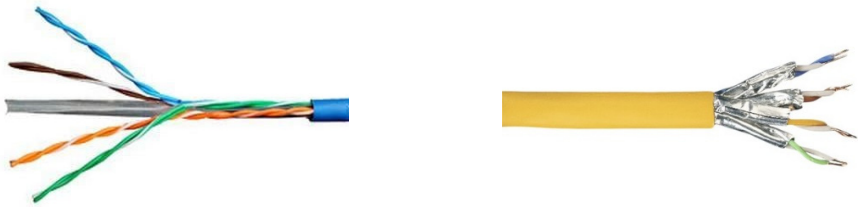
## 2.4 Tipuri de conexiuni între dispozitivele dintr-o rețea informatică

Conexiunea (legătura) dintre dispozitivele unei rețele informatice se poate realiza prin cabluri sau wireless (fără fir). Funcție de semnalul transmis pe cablu acestea poate fi electric sau optic.

Cablurile utilizate pentru conectarea dispozitivelor într-o rețea informatică pot fi:

- Cablul UTP/STP (Unshielded/Shielded twisted pair);
- Cablu coaxial;
- Fibre optice.

Cablul UTP/STP (Fig. 2.21) este format din 4x2 perechi de fire de cupru cu un diametru între 0.2 și 1 milimetri izolate individual, răsucite în perechi și învelite în teflon. Rata de transfer pe un astfel de cablu este de 100 Mbps la 100 metri. Se simbolizează TbaseT.



a) UTP b) STP

Fig. 2.21. Cablu rețea

Diferența între cablul UTP și STP este ecranarea și anume în cazul cablului STP fiecare pereche de fire este învelită într-o folie care protejează împotriva interferențelor, pe când în cazul cablului UTP ecranarea de realizează pentru toate firele la comun.

Mufa utilizată pentru conectarea dispozitivelor prin cablu UTP/STP este de tip RJ45 (Fig. 2.22).

Fig. 2.22. Mufă RJ45 [42]

Pentru conectarea mufei RJ45 există două scheme de conectare T568A și T568B (Fig. 2.23). Cablurile care au la capete aceeași schemă de conectare indiferent care este sunt identice. De fapt legăturile sunt unu la unu și funcționează identic doar secvența de culori diferă. Aceste cabluri sunt utilizate pentru conectarea dispozitivelor la o rețea informatică. Cea mai des utilizată schemă este T568 B.

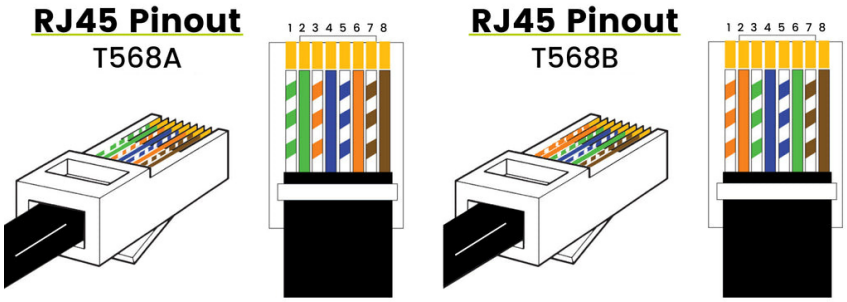



Fig. 2.23. Schema conectare mufa RJ45 [43]

Dacă se dorește conectarea directă a două dispozitive între ele fără a face parte dintr-o rețea atunci cablul trebuie mufat la un capăt după schema T568A iar la celălalt după schema T568B. Se observă că în acest caz legătura nu mai este unu la unu ci se schimba perechea de fire verde cu cea portocalie. Un astfel de cablu se numește cablu cross-over (adică încrucișat) și de fapt se schimbă între ele conexiunile RX (recepție) și TX (transmitere)(Fig. 2.24).

Într-o rețea informatică echipamentele care conectează mai multe dispozitive (router, switch, etc) realizează ele comutarea semnalului de la TX la RX și invers nefiind necesară comutarea fizică prin intermediul cablului. Dar dacă conectăm direct două dispozitive atunci este obligatorie utilizarea unui cablu cross-over deoarece nu avem nici un echipament care să realizeze comutarea, aceasta fiind realizată fizic prin intermediul cablului.

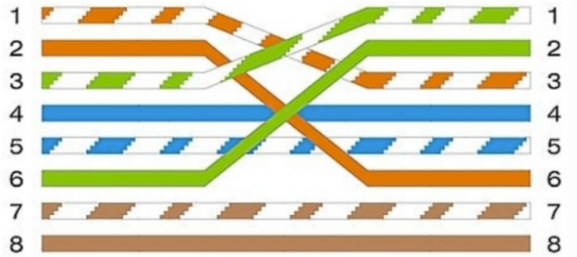



Fig. 2.24. Schemă cablu cross-over [44]

O altă modalitate de conectare a dispozitivelor într-o rețea de informatică este utilizând un cablu coaxial (Fig. 2.25).

Fig. 2.25. Cablu coaxial [45]

Cablu coaxial există în doua variante:

- Subțire simbolizat TBase2 cu o rată de transfer de 10 MBps la 200 metri;
- Gros simbolizat TBase5 cu o rata de transfer 10 MBps la 500 metri.

Cablurile coaxiale sunt utilizate cel mai des pentru transmiterea semnalelor video și audio. Au fost utilizate la începuturi pentru transmiterea semnalului de internet dar datorită vitezei reduse au fost înlocuite cu cablurile UTP.

Fibra optică (Fig. 2.26) este un tip de cablu care utilizează impulsuri de lumină pentru a transmite semnale de date. Este un fir subțire de sticlă sau plastic, denumit fibră, care este folosit pentru a transporta semnale de lumină între două puncte. Acesta este utilizat pentru a transmite semnale de date în mare viteză pe

distanțe lungi și este un mijloc popular de comunicare în rețelele de telecomunicații.

Fig. 2.26. Fibră optică [46]

Fibre optice pot fi:

- mono-mod echipate cu diode laser, transfera un singur fir optic pe distante mari;
- multi-mod echipate cu diode led, transfera mai multe fire optice pe distante mai reduse.

Rata de transfer este de 10 GBps la 2km (pe distante mai mici rata creste pana la 111 Gbps). Avantaj esential la greutate si rata de transfer: 100 km pe UTP cantareste 8 t iar pe fibra doar 100 kg; Se simbolizeaza cu TBaseFX

## 2.5 Configurarea unui router

Echipamentele cel mai frecvent utilizate pentru conectarea mai multor dispozitive în rețea sunt router-ele și switch-urile. Pentru subrețele locale cel mai facil este utilizarea unui router. Deoarece aceste echipamente sunt frecvent utilizate de majoritatea persoanelor pentru a crea o rețea locală privată la domiciliu, în continuare se vor prezenta pașii ce trebuie urmați pentru a configura o astfel de rețea.

Dispozitivele tip router pot fi achiziționate din orice magazin de electronice sau IT. Companiile de distribuție a internetului oferă spre închiriere astfel de dispozitive. O dată achiziționat acest dispozitiv pentru a ne putea conecta la acesta trebuie să cunoaștem credențialele inițiale. Acestea se găsesc afișate de obicei pe spatele echipamentului (Fig. 2.27).


Fig. 2.27. Credențiale router TP-Link

Aceste credențiale diferă funcție de compania producătoare și funcție de modelul de router ales. De asemenea și interfața de configurare este diferită dar pașii ce trebuie urmați pentru a configura dispozitivul sunt aceiași.

În continuare se vor prezenta etapele de configurare a unui router TP-Link model Archer C6. După ce echipamentul este alimentat cu energie electrică putem să ne conectam la interfața de configurare (Fig. 2.28) atât prin cablu cât și wireless. Prin introducerea credențialelor în fereastra de logare vom avea acces la pagina de configurare. În cazul echipamentului prezentat este necesar doar introducerea parolei. Alte dispozitive pot cere și utilizator și parolă.

Primul pas este să schimbam această parolă universala (toate dispozitivele de același fel au aceleași date generale pentru logare).

Utilizatorii care nu au experiență în configurarea acestor dispozitive pot utiliza interfața rapidă de configurare (quick setup) a dispozitivului (Fig. 2.29). Prin această interfață se vor cere toate informațiile necesare configurări de bază, nefiind necesar ca utilizatorul să caute prin meniu unde trebuie introduse acestea.

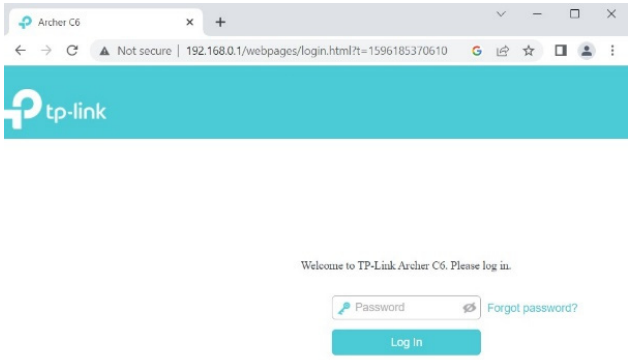



Fig. 2.28. Interfață logare router

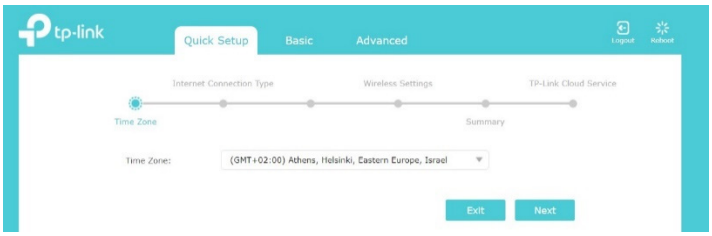



Fig.2.29. Interfață de configurare rapidă a dispozitivului

Pentru început trebuie să configuram accesul la internet al router-ului (Fig. 2.30). Pentru accesul la internet putem utiliza Static IP atunci când știm ce IP dorim să alocam dispozitivului. Această configurare este utilă când dorim accesul de la distanță la acest dispozitiv. O altă modalitate de configurare este Dynamic IP, caz în care dispozitivul primește IP alocat dinamic de un server

DHCP. Diferența dintre aceste două configurări este doar modul de alocare al adresei IP.

În cazul utilizării router-ului pentru o rețea locală cu conectare la internet printr-un serviciu pus la dispoziție de un distribuitor de internet cel mai probabil conexiunea la internet este de tip PPPoE (Point-to-Point Protocol over Ethernet), adică utilizând un user și o parolă. Aceasta este soluția de bază și cea mai simplă pusă la dispoziție de companiile de distribuție a internetului.

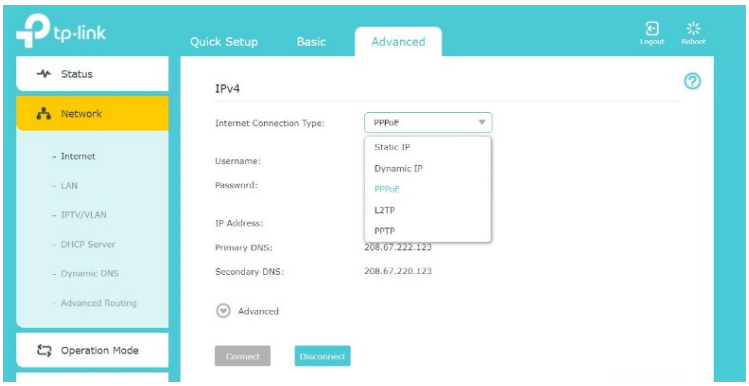



Fig. 2.30. Configurare acces internet

Dacă se dorește crearea unei rețele locale private atunci trebuie activat serverul de DHCP (Fig. 2.31). Aici se setează plaja adreselor IP ce vor fi utilizate pentru dispozitivele conectate la acest router, de asemenea se setează adresa locală a router-ului (Gateway) care reprezintă de fapt ieșirea spre internet. Tot aici se pot rezerva adrese IP pentru dispozitivele care necesită adresă IP statică. Aceste adrese IP sunt rezervate pentru dispozitive cum ar fi imprimantele, camerele de supraveghere, etc , unde dorim ca adresa IP să rămână aceeași.

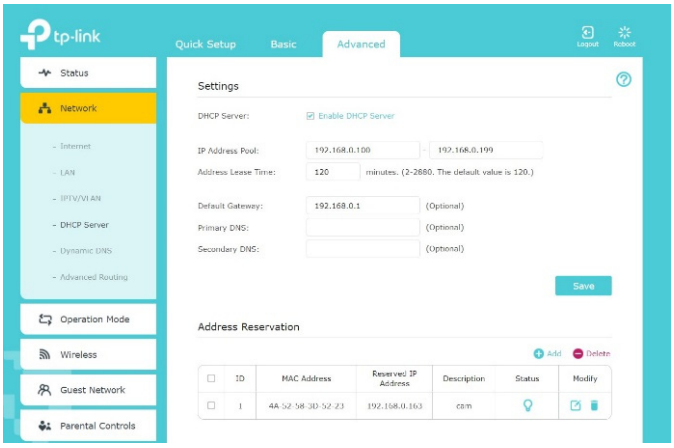



Fig. 2.31. Configurare server DHCP

Având acces la internet este necesar de a seta și o rețea wireless. Majoritatea router-elor oferă WiFi atât la 2.4GHz cât și la 5GHz. Putem configura fiecare rețea în parte (având două rețele WiFi distincte) sau putem avea aceeași rețea pentru ambele frecvențe, caz în care dispozitivele ce se conectează la rețea aleg automat frecvența. Evident că frecvența de 5GHz are o viteză de transfer a datelor mai mare, astfel dispozitivele care se pot conecta la 5GHz vor prefera să realizeze o astfel de conexiune.

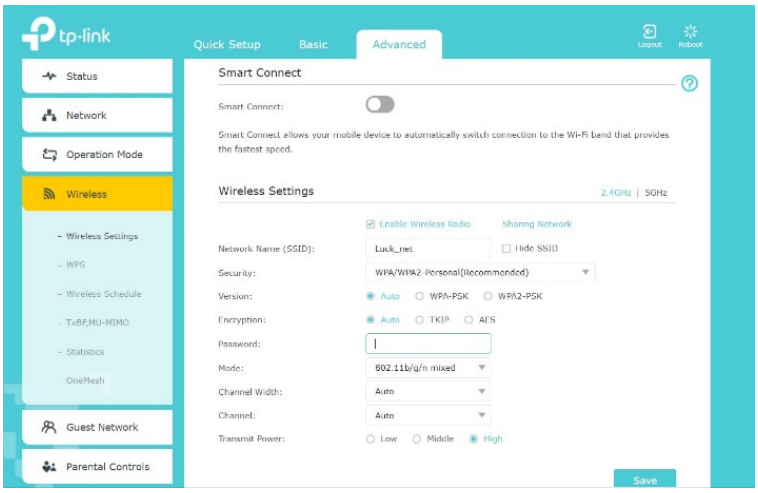



Fig. 2.32. Configurare rețelei WiFi

Rețeaua WiFi (Fig. 2.32) trebuie să aibă un nume (SSID) și o parolă. După completarea acestor câmpuri se poate modifica modul de criptare al parolei, respectiv se poate ascunde numele rețelei WiFi. Pe lângă modul de criptare al parolei care conferă securitate rețelei și ascunderea numelui rețelei crește securitatea acesteia deoarece rețeaua nu este descoperită de dispozitivele dimprejur.

După realizarea etapelor prezentate rețeaua locală privată este configurată și funcțională. Putem verifica funcționalitatea rețelei accesând opțiunea status din meniu (Fig. 2.33). Se poate observa că există si posibilitatea activării unei rețele WiFi pentru vizitatori. Această rețea dacă este activă atunci orice dispozitiv se poate conecta wireless la aceasta fără a fi nevoie de user și parolă, adică este o rețea cu acces liber.

Pe langă aceste configurări de bază necesare bunei funcționări a rețelei mai există unele configurări suplimentare disponibile. Dintre acestea vom menționa accesul controlat al dispozitivelor în rețea precum și controlul parental.

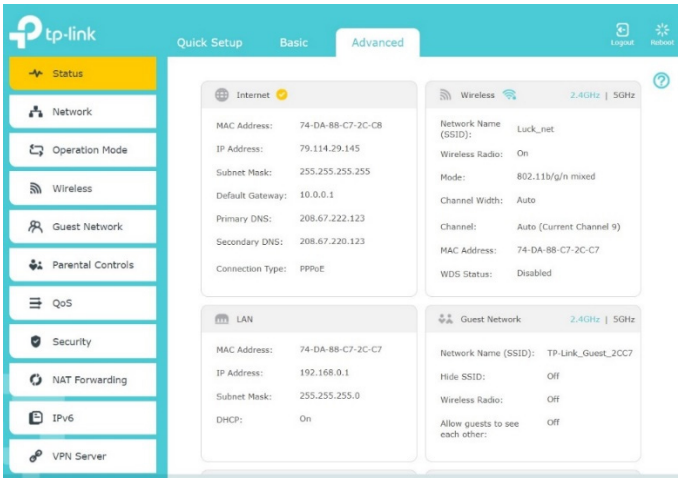



Fig. 2.33. Statusul rețelei locale

O configurare suplimentară este aceea de filtrare a dispozitivelor ce se conectează la această rețea pe baza adresei MAC. Adresa MAC (Media Access Control) este o adresă unică atribuită unui dispozitiv de catre producător. Această adresă este utilizată pentru identificarea unică a dispozitivului în rețea (exemplu de adresă MAC - . A8:5E:45:51:7D:B5). Utilizând această adresă MAC se pot defini liste ale dispozitivelor din rețea. Se pot crea listă cu dispozitive acceptate în rețea (white list) (Fig. 2.34) sau liste cu dispozitive neacceptate în rețea (black list).

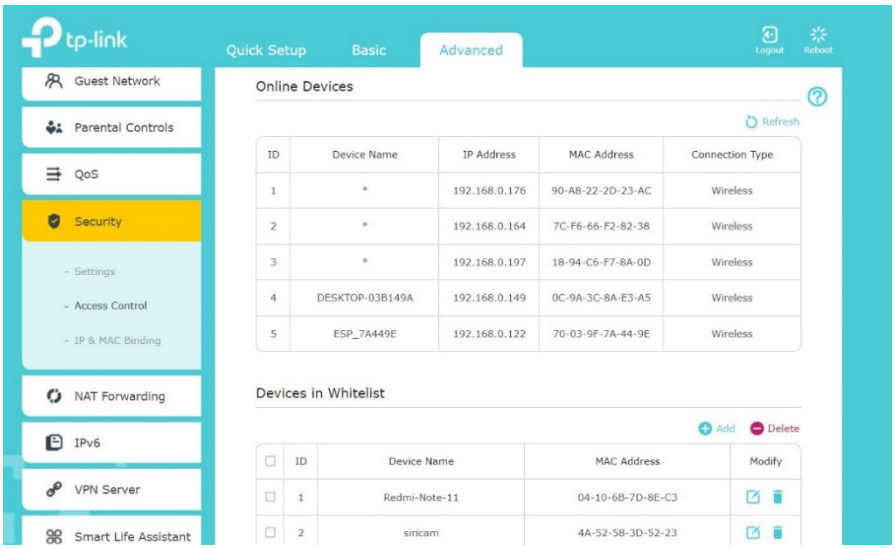



Fig. 2.34. Acces controlat prin listă adrese MAC

La opțiunea securitate din meniu se poate activa acest serviciu de control al accesului precum și optarea pentru utilizare a listei albe sau negre (Fig. 2.35).

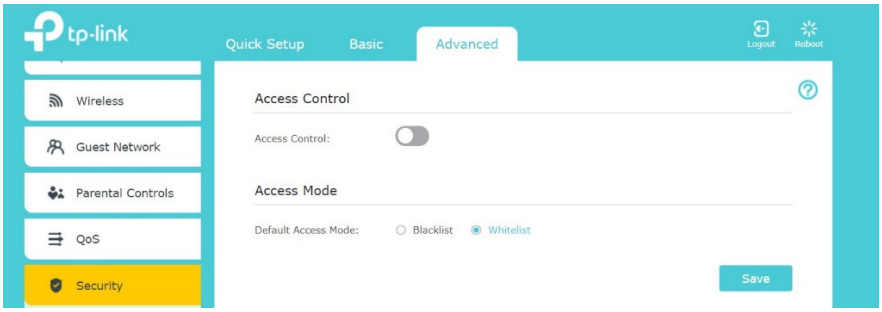



Fig. 2.35. Interfață acces controlat

Pentru securitatea accesului copiilor la internet majoritatea router-elor pun la dispoziție opțiunea de control parental (Fig. 2.36). Cu ajutorul acestei opțiuni părinții pot defini dispozitivele prin care au copii acces la internet, iar pentru aceste dispozitive se pot utiliza filtre pentru conținut (pe bază de cuvinte cheie) respectiv se pot seta limitări de timp.

Filtrarea conținutului pe baza de cuvinte cheie nu este o filtrare foarte sigură. În acest scop pot fi utilizate în paralel cu această filtrare și unele software-uri dedicate acestui scop cum ar fi Family Link pus la dispoziție de către google.

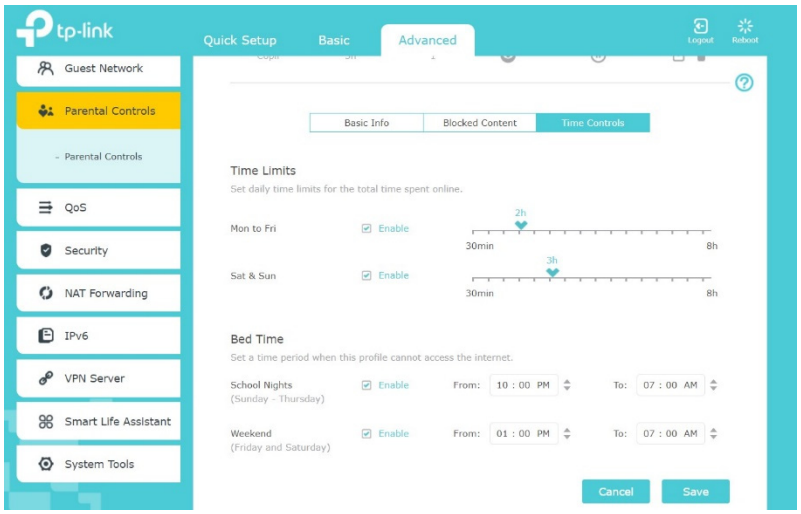



Fig. 2.36. Control parental

Limitarea accesului la internet al copiilor se poate face prin setarea unui timp maxim de utilizare al acestuia pentru fiecare zi a săptămâni respectiv al weekend-ului precum și al intervalului orar dedicat somnului când accesul la internet este oprit.

## 2.6 Stocarea datelor în rețelele informatice

Toate informațiile din rețelele de calculatoare sunt stocate pe hard disk-uri, care sunt dispozitive de stocare a datelor pe termen lung, și au fost prezentate în capitolul introductiv.

Aceste hard disk-uri sunt dispozitive electronice și ca atare se pot defecta astfel informația se poate pierde. Din acest motiv este indicat realizarea periodică a unor backup-uri.

În momentul actual stocarea datelor pe servere prin serviciile de cloud puse la dispoziție de furnizori de serviciu precum Google, Microsoft, Amazon sunt din ce in ce mai des utilizate. Acest mod de stocare a datelor este util datorită accesului facil la acestea de oriunde, doar vulnerabilitatea la atacuri cibernetice fiind un risc.

Pentru a asigura securitatea datelor, serverele prezintă un nivel ridicat de securitate, inclusiv criptarea datelor și/sau autentificarea în mai mulți pași. Aceste servicii asigură siguranța datelor față de atacurile cibernetice dar nu și față de defecțiuni ale echipamentelor hardware. În acest scop s-au dezvoltat sisteme de backup automat a informațiilor de pe hard disk. Un astfel de sistem fiind sistemul RAID.

Sistemul RAID (Fig. 2.37) (Redundant Array of Independent Disks) este o tehnologie de stocare a datelor pe mai multe disk-uri pentru a crea o unitate logică de stocare a datelor mai mare și mai fiabilă decât un singur disc. Sistemul RAID poate îmbunătăți performanța sistemului de stocare, protejând în același timp datele împotriva pierderii acestora cauzate de defecțiunile disk-urilor [47].

Fig. 2.37. Sistem RAID [47]

Există mai multe nivele ale sistemului RAID, cele mai uzuale fiind:

- RAID 0;
- RAID 1;
- RAID 5;
- RAID 10.

Nivelul RAID 0 (Fig. 2.38) utilizează minim două hard disk-uri, oferă performanță excelentă, nu are redundanță deci nu asigură siguranța datelor în caz de erori. Prin împărțirea datelor pe mai multe discuri se asigură o performanță ridicată deoarece disk-urile pot fi accesate simultan. Neavând backup a datelor acest sistem RAID 0 este vulnerabil la erorile hardware.

Și în cazul sistemului RAID 1 (Fig. 2.39) utilizează minim două discuri. Performanța și în acest caz este bună, iar redundanța este excelentă deoarece discurile sunt în oglindă. Sistemul RAID1 se folosește mai ales când dorim o securitate sporită a informațiilor stocate. Dezavantajul acestei configurații este că spațiul de stocare este de jumătate din suma capacității discurilor.

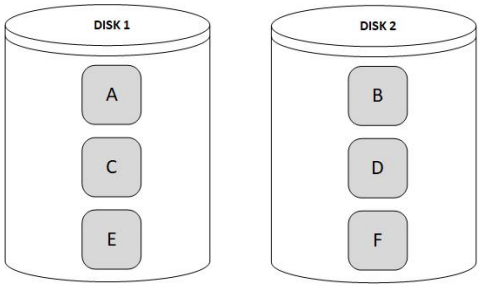



Fig. 2.38. Sistem RAID 0

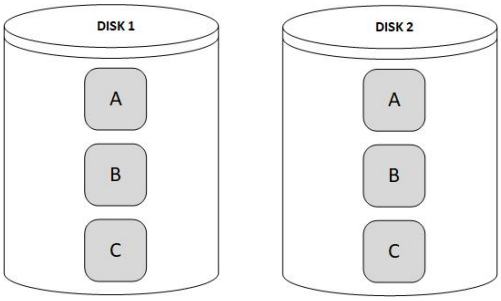



Fig. 2.39. Sistem RAID 1

Această configurație oferă în același timp performanța ridicată și o redundanță bună a datelor în cazul deteriorării unui disk. Configurația RAID5 are nevoie de minim trei discuri. Această soluție oferă cel mai scăzut cost raportat la performanță și redundanță. În cazul configurației RAID 5 cu trei discuri (Fig. 2.40) informația este distribuită pe toate cele 3 discuri (identic cu RAID 0) dar în același timp informației de pe două discuri i se asigură backup pe cel de al treilea disc acest lucru oferindu-i o foarte bună redundanță.

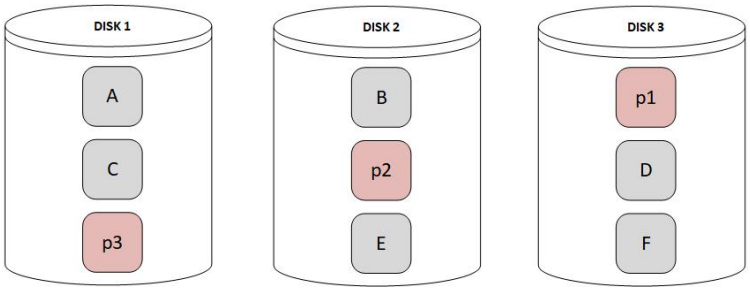



Fig. 2.40. Sistem RAID 5

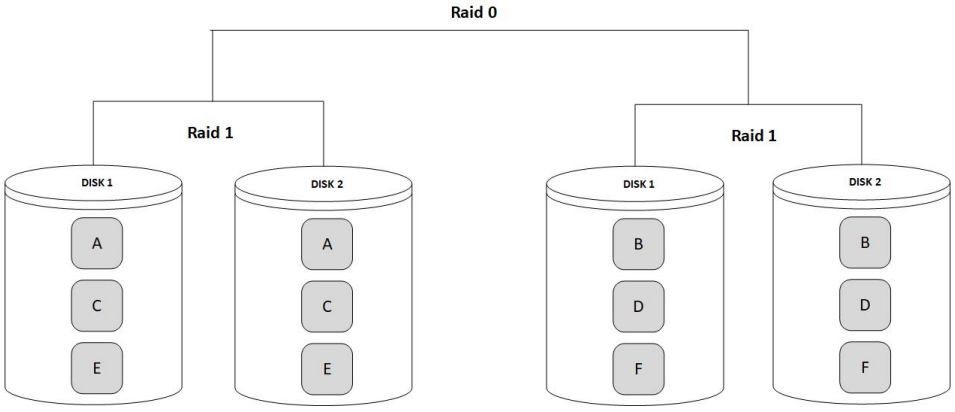



Fig. 2.41. Sistem RAID 10

Configurația RAID 10 (Fig. 2.41) este o combinație între configurațiile RAID 0 și RAID 1. Pentru această configurație avem nevoie de minim patru discuri, dar obținem o performanță și o redundanță excelentă.

Astfel beneficiem de viteza configurației RAID0 și de siguranța configurației RAID1. Dezavantajul acestui sistem RAID 10 este că se poate folosi numai jumătate din capacitatea discurilor, acest lucru căscând bugetul. Această configurație este perfectă pentru servere ce conțin baze de date.

# 3 BAZE DE DATE

Bazele de date sunt colecții organizate de informații, care permit stocarea, actualizarea și regăsirea informațiilor. Acestea sunt proiectate pentru a permite accesul rapid și eficient la informație și pentru a asigura gestionarea și actualizarea acestora. Bazele de date pot fi utilizate în toate domeniile.

În general, o bază de date este compusă din tabele, fiecare dintre acestea având un set specific de coloane și rânduri.

Bazele de date sunt gestionate și accesate de către programe de software numite sisteme de gestiune a bazelor de date. Aceste sisteme permit utilizatorilor să creeze, să modifice și să interogheze baze de date cu ușurință și eficiență [48].

Există multe tipuri de baze de date:

- **Baze de date relaționale**: elementele dintr-o bază de date relațională sunt organizate ca un set te tabele, cu rânduri și coloane;
- **Baze de date orientate pe obiecte**: în care informațiile sunt reprezentate sub formă de obiecte
- **Baze de date distribuite**: sunt formate din două sau mai multe fișiere aflate în locații diferite.
- **Baze de date NoSQL**: denumite si baze de date nerelationale, permit stocarea și gestionarea datelor nestructurate sau semi-structurate;
- **Baze de date grafice**: stochează datele sub forma unor entități și a relațiilor dintre acestea.
- **Baze de date OLTP**: sunt baze de date rapide, pentru analize, destinate rulării unui număr mare de tranzacții efectuate de mai mulți utilizatori.
- **Baze de date ierarhice**: stochează informații într-o structură ierarhică care leagă înregistrările într-o structură arborescentă.
- **Baze de date open source**: au codul sursă de tip open source, cele mai cunoscute fiind cele SQL și NoSQL
- **Baze de date în cloud**: colectie de date, structurate sau nestructurate aflate în platformă cloud privată, publică sau hibridă.
- **Baze de date multi-model**: combină diferite tipuri de modele de baze de date într-un singur sistem back-end integrat cu care se pot acomoda diverse tipuri de date.
- **Baze de date pentru documente**: concepute pentru stocarea, preluarea și managementul informațiilor orientate pe documente.
- **Baze de date autonome**: sunt cele mai noi sisteme ce funcționează în cloud și utilizează machine learning.

Conform ANSI (Institutul National American pentru Standarde) și SPARC (Comitetul de planificare și cerințe privind standardele) arhitectura unui sistem de gestiune a bazelor de date se împarte pe trei nivele (Fig. 3.1):

- Extern (nivel grafic);
- Conceptual (nivel software);
- Intern (nivel fizic).

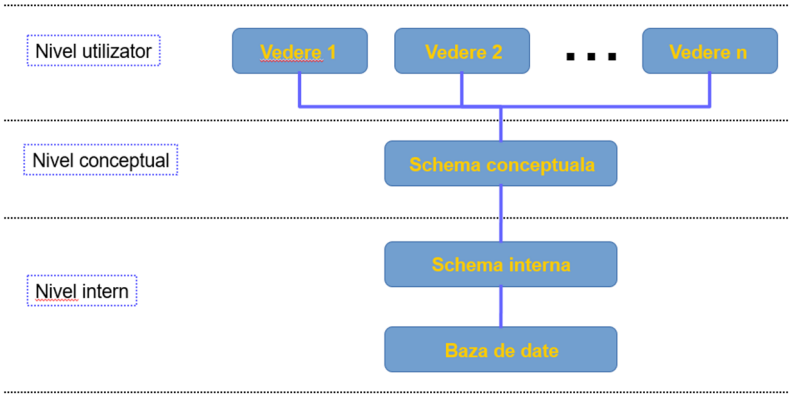



Fig. 3.1. Arhitectura unui sistem de baze de date

Nivelul extern sau utilizator cuprinde interfața grafică utilizator, adică modul de comunicare între utilizator și baza de date.

Nivelul conceptual realizează prin software de fapt legătura între utilizator și baza de date. Adică preia cerințele utilizatorului, și modifică baza de date în concordantă cu acestea și/sau oferă utilizatorului răspunsul interogării bazei de date.

Nivelul intern sau fizic cuprinde toate informațiile stocate în baza de date (aflate pe discuri) precum și structura și legăturile din cadrul bazei de date.

Informația din bazele de date se poate stoca local sau pe un server. Bazele de date cu caracter general sau cele la care se dorește accesul de la distanță trebuie stocate pe servere. Accesul la bazele de date (Fig. 3.2) se poate face local adică se accesează baza de date din rețeaua locală în care se află serverul sau din exterior prin intermediul internetului [49].

Această arhitectura numită client-server facilitează partajarea informației din baza de date. Ținând cont de multiplele posibilități de conectare la baza de date trebuie avut grija la securizarea informațiilor. Pe lângă diferitele firewall-uri care protejează serverul, se poate mări securitatea prin eliminarea unor privilegii acordate utilizatorilor externi. Printr-o astfel de măsură se poate de exemplu acorda privilegii de modificare a structuri bazei de date și de acces la informații confidențiale doar utilizatorilor locali, reducând astfel riscul unor atacuri cibernetice asupra bazei de date.

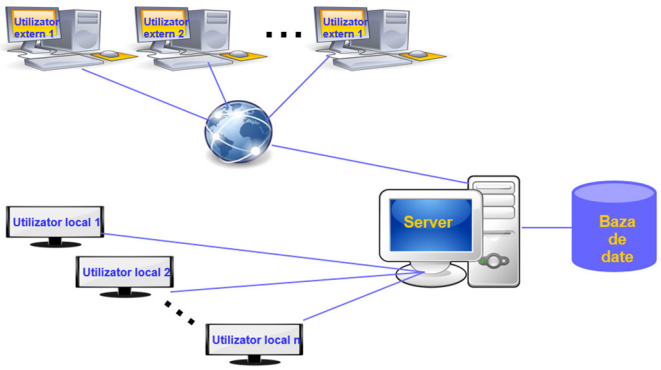



Fig. 3.2. Accesul la o bază de date

În arhitectura client-server, pentru conectarea la baza de date utilizatorul trebuie să introducă un user și o parolă. Funcție de nivelul de încredere acordat utilizatorului, managerul bazei de date poate acorda acestuia diferite drepturi [50].

## 3.1 Baze de date relaționale

Prin bază de date înțelegem un sistem proiectat pentru a oferi un mecanism organizat, capabil să stocheze, să actualizeze și să regăsească informații. În cadrul unei baze de date informația este stocată sub formă tabelară în unul sau mai multe tabele. Baza de date relațională (Fig. 3.3) reprezintă un set de tabele rațional definite. Adică în cazul unei baze de date relaționale informația este stocată în tabele între care există diferite relații de legătură [50].

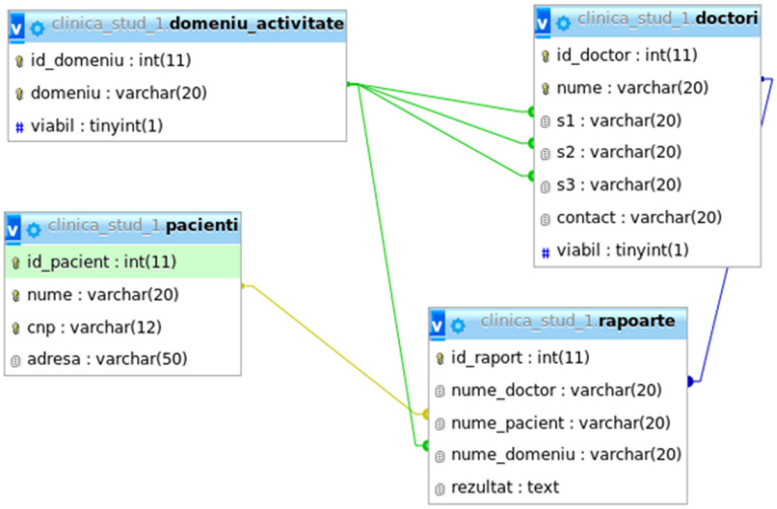



Fig. 3.3. Baza de date relațională

Terminologia de bază utilizată în cadrul unui sistem de gestiune a bazelor de date cuprinde următorii termeni:

- **entitate**: tabel cu linii și coloane (grup de date);
- **atribut**: coloana unei entități cu o anumită denumire și cu un anumit tip de conținut;
- **domeniu**: mulțimea valorilor permise pentru unul sau mai multe atribute;
- **tuplu**: un rand dintr-o entitate (înregistrare);
- **grad**: numărul de atribute al unei entități;
- **cardinalitate**: numărul de tupluri al unei entități;

Pentru o mai bună înțelegere a entităților (tabelelor) putem face comparației din Tabelul 3.1.

Tabelul 3.1. Termeni comparativi

| **Termeni formali** | **Alternativa 1** | **Alternativa 2** |
|---|---|---|
| Entitate | Tabel | Fişier |
| Tuplu | Rând | Înregistrare |
| Atribut | Coloană | Câmp |

Pentru a putea asigura integritatea datelor precum și un acces rapid și eficient la informațiile din baza de date s-a definit conceptul de cheie. Cu ajutorul acestui concept se organizează informațiile în entități și de asemenea asigură realizarea corespunzătoare a legăturilor dintre entități.

Principalele concepte de chei utilizate sunt:

- cheia relațională: care reprezintă modalitatea prin care un tuplu este identificat în mod unic în cadrul unei relații;
- super cheia: modalitatea prin care un atribut sau un grup de atribute asigură identificarea unui tuplu în mod unic;
- cheia candidat: este o super cheie pentru care fiecare atribuit este esențial;
- cheia primară: este una dintre cheile candidat;
- cheia externă: este un atribut al unei entități care are aceleași valori cu o cheie primară dintr-o altă entitate (aceste chei stau la baza legăturilor dintre entitățile unei baze de date).

Utilizând acest concept de cheie putem accesa într-un mod eficient informația din baza de date deoarece pe baza acestora putem organiza toată informația într-un mod optim.

## 3.2 Proiectarea unei baze de date

Proiectarea sau modelarea unei baze de date presupune în esență crearea entităților (tabelelor) și a relațiilor de legătură dintre acestea [51]. Modul de proiectare a unei baze de date poate fi împărțită în mai multe etape astfel:

- **stabilirea domeniului**: presupune stabilirea ariei (domeniului) unde utilizăm baza de date pentru a ştii ce fel de informaţii se vor stoca în această bază de date;
- **definirea parţilor specifice**: stabilirea numărului de entităţi (tabele);
- **identificarea atributelor**: stabilirea atributelor (coloanelor) necesare;
- **definirea atributelor şi gruparea pe entităţi**: stabilirea numelui şi tipului fiecărui atribut funcţie de informaţia stocată în acesta şi gruparea acestora în entităţi;
- **identificarea cheilor primare ale fiecărei entităţi**: găsirea unei chei primare care face posibilă identificarea în mod unic a unui tuplu în cadrul entităţi;
- **identificarea cheilor externe**: identificare legăturilor necesare între entităţi;
- **normalizarea entităţilor**: presupune normalizarea datelor astfel încât să minimizăm redundanţa.

Pentru modelarea unei baze de date pe lână etapele enumerate ar trebui să mai ţinem cont de trei reguli care ne ajută la realizarea unei baze de date eficiente şi fiabile. Aceste reguli sunt:

- atributele unui tabel trebuie alese astfel încât să permită un număr nelimitat de înregistrări fără modificarea structurii tabelului; adică definirea tipului atributului să fie astfel făcută încât să nu limiteze datele introduse. De exemplu dacă un atribut trebuie să înregistreze prenumele unor persoane şi definesc tipul acestuia ca un sir de maxim 5 caractere (varchar 5), atunci nu pot înregistra persoanele a căror prenume este format din mai mult de 5 caractere.
- cheia primară a unui tabel trebuie sa fie apreciata în totalitate (fiecare atribut trebuie sa se afle în dependenta funcţionala cu toate atributele din cheia primara compusă). Trebuie să fim atenţi, dacă cheia primară este compusă (formată prin utilarea mai multor atribute), atunci pentru eficienţă fiecare atribuit utilizat trebuie sa fie esenţial. De exemplu dacă avem o entitate cu toţi studenţi unei universităţi şi dorim identificare unică a studenţilor unei facultăţi este suficient ca pentru cheia primară să utilizăm o cheie compusă din atributele marca (id student) şi facultate. Dacă am fi adăugat pe lângă cele două atribute şi atributele cu nume şi prenume ale studentului, aceste din urmă nu erau necesare la identificarea unica a studentului astfel ele doar îngreunând procesul.
- Toate atributele unui tabel trebuie sa fie în aceeaşi entitate cu cheia primara. Nu este acceptată utilizarea în cadrul unei chei primare compuse a atributelor din mai mule entităţi.

După modelarea unei baze de date respectând etapele și regulile prezentate dacă se dorește ca și alți utilizatori să o poată accesa atunci trebuie să li se acorde drepturi asupra bazei de date. Acest lucru este posibil datorită sistemului de privilegii. Prin utilizarea unui astfel de sistem de privilegii se pot valida utilizatorii ce se conectează, apoi asocia cu baza de date la care au acces urmat de verificarea privilegiilor (drepturilor) ce le au asupra bazei de date.

Pentru validarea unui utilizator prima oară se verifică dacă îi este permisă conexiunea la serverul de baze de date, apoi pentru fiecare interogare a bazei de date din partea utilizatorului se verifică dacă acesta are privilegii pentru cererea din interogare.

Principalele privilegii ale unui utilizator asupra unei baze de date sunt [49. 50, 51]:

- **Usage** – dreptul de acces la DB fără nici un privilegiu suplimentar;
- **Select (tabele)** – rularea comenzilor de selectare a informațiilor din entități;
- **Insert (tabele)** – înserarea de valori noi în tabele;
- **Update (tabele)** – actualizarea valorilor din tabele;
- **Delete (tabele)** – ștergerea unor valori din tabel;
- **Index (tabele)** – posibilitatea de a crea și șterge index-uri;
- **Alter (tabele)** – dreptul de modifica structura unui tabel;
- **Create (db,tabele,index)** – dreptul de a crea DB, tabele, index-uri;
- **Grant (db,tabele)** – dreptul de a oferii altor utilizatori privilegii proprii asupra bazei de date;
- **Reload (db,tabele)** – dreptul de a folosi comanda FLUSH pentru reîncărcarea bazei de date, a tabelelor și a sistemului de privilegii;
- **Shutdown (administrarea serverului)** – dreptul de a oprii serverul;
- **Process (administrarea serverului)** – posibilitatea de a afișa toate procesele din server la un moment dat;
- **File (db,tabele)** – dreptul de a folosi fișiere externe pentru încărcarea sau salvarea datelor din server.

După modelarea bazei de date și crearea utilizatorilor și privilegiile acestora, trebuie avut grijă doar ca serverul să fie activ pentru a putea gestiona informațiile din baza de date.

### 3.3 Sistemul MySQL de gestiune a bazelor de date

Există pe piață mai multe sisteme de gestiune a bazelor de date (SGBD) cele mai cunoscute fiind: MySQL, SQL Server, Access, Oracle, Sybase, DB2, Microsoft SQL.

MySQL este un sistem de baze de date relațional dezvoltat de TcX DataKonsult AB în 1996. Soluția a revoluționat piața bazelor de date și a dus la

parteneriate importante cu RedHat, Veritas, Novell sau Rackspace. În 2008, MySQL a fost achiziționat de Sun Microsytems, care la rândul ei a fost cumpărata de Oracle în 2009.

MySQL este unul dintre cele mai populare sisteme de gestiune de baze de date (SGBD) în regim Open Source.

Cele mai importante caracteristici ale MySQL sunt:

- Este un SGBD care asigură gestiunea unei colecții structurate de date.
- Este un SGBD relațional. Datele sunt organizate în tabele. Intre tabele sunt stabilite relații și constrângeri de integritate. Pentru gestiunea datelor se folosește limbajul SQL (Structured Query Language) care este un standard în acest domeniu.
- Este un sistem client -server. Serverul de baze de date este separat logic și deseori fizic (pe calculatoare diferite) de programele client care asigura interfața operațiilor cu baza de date. Serverul MySQL suporta o paleta largă de programe client, scrise în diverse limbaje de programare.
- Este distribuit în formula Open Source Software. Aceasta formula permite oricărei persoane să utilizeze și să modifice programul, care poate fi luat gratuit de pe Internet. Formula utilizată pentru licențiere este GPL (GNU General Public License).

Argumente în favoarea utilizării SGDB-ului MySQL:

- **Este rapid**. În multe aplicații este chiar mai rapid decât alte sisteme foarte scumpe.
- **Este fiabil**. Asigura o bună protecție a datelor prin mecanisme specifice.
- **Este ușor de utilizat**. Chiar dacă oferă o paleta foarte bogată de funcții, organizarea acestora și modul în care au fost create facilitează folosirea lor.
- **Bazat pe fire de execuție nucleu**. Poate folosi mai multe procesoare daca sistemul este multi-procesor.
- **Suport pentru multe limbaje de programare**. Interfețe pentru clienți: C, C++, Eiffel, Java, Perl, PHP, Python.
- **Multiplatforma**. Sunt suportate următoarele sisteme de operare: Linux, Windows, AIX, DEC Unix, HP-UX, Mac OS XServer, OS2 Warp, Solaris, SCO UnixWare etc.
- **Securitate ridicată**. Utilizează parole criptate.
- **Scalabilitate bună**. Suporta baze de date de mari dimensiuni. Exista aplicații cu peste 5 miliarde înregistrări rulând pe MySQL.

Limbajul utilizat de către SGDB-ul MySql este SQL. SQL este acronimul pentru Structured Query Language și este un limbaj standard pentru comunicarea cu bazele de date relaționale.

In ciuda simplității sale, SQL este un limbaj foarte puternic, care:

- poate obține accesul la date stocate în mai multe tabele;
- poate filtra datele dorite;
- poate insera, sorta, rezuma și afișa rezultatele.

Acest limbaj are ca scop inserarea datelor, interogații, actualizare și ștergere, modificarea și crearea schemelor, precum și controlul accesului la date. A devenit un standard în domeniu (standardizat ANSI-ISO), fiind cel mai popular limbaj utilizat pentru crearea, modificarea, regăsirea și manipularea datelor de către SGBD-urile [52].

## 3.4 Limbajul pentru gestionarea bazelor de date MySQL

SQL (Structured Query Language) este un limbaj de programare standardizat folosit în gestionarea bazelor de date relaționale. SQL este folosit pentru a crea, manipula și interoga bazele de date, precum și pentru a gestiona accesul și securitatea datelor.

SQL a fost dezvoltat în anii 1970 de către IBM, dar a devenit un standard ANSI în anul 1986 și un standard ISO în anul 1987. SQL permite utilizatorilor să creeze tabele, să adauge date în tabele, să modifice date existente, să selecteze date pentru a fi afișate sau să fie procesate și să șteargă date din tabele [50,52].

SQL este un limbaj declarativ, ceea ce înseamnă că utilizatorii descriu ceea ce doresc să se întâmple cu datele, nu cum să se întâmple. În loc să se scrie cod pentru a efectua o operațiune, utilizatorii spun SGBD-ului ce să facă și baza de date se ocupă de restul [50,52].

În continuare se vor prezenta cele mai uzuale comenzi SQL utilizate de către sistemul de gestiune a bazelor de date MySQL.

Pentru conectarea la serverul MySQL este utilizată comanda:

**mysql –u user –p paorla;**

Prin această comandă se realizează conectarea la serverul unde se găsește baza de date. Deoarece pe server se găsesc mai multe baze de date pentru a vizualiza bazele de date disponibile pentru utilizatorul ce s-a conectat putem utiliza comanda:

**show databases;**

Această comandă va afișa toate bazele de date la care utilizatorul are anumite privilegii. În continuare trebuie selectată baza de date pe care dorim să o accesam. Pentru această operație se va folosi:

**use nume_db;**

Dacă utilizatorul nu dorește utilizarea unei baze de date existente ci crearea uneia noi, poate realiza acest lucru utilizând comanda:

**CREATE DATABASE nume_DB;**

Un utilizator poate crea baze de date doar dacă îi este acordat acest privilegiu. Etapa următoare ar fi crearea tabelelor (entităților) unde vor fi stocate informațiile. Crearea unui tabel de realizează prin comanda:

```
CREATE TABLE nume_tabel (
coloana1 tip_date,
coloana2 tip_date,
coloana3 tip_date,
……
);
```

În cadrul acestei comenzi trebuie specificat numele tabelului apoi fiecare atrubut (coloană cu nume și format). O dată creat tabelul, asupra structurii acestuia se poate reveni (dacă s-a omis sau greșit ceva) prin comanda:

```
ALTER TABLE nume_tabel
ADD/DROP/MODIFY COLUMN nume_coloana tip_date;
```

Numele tabelului căruia dorim să îi modificăm structura, este urmat de unul din cuvintele cheie ADD , DROP, MODIFY care precizează ce dorim să facem: să adăugam, să ștergem sau să modificăm coloanele tabelului. De asemenea dacă dorim ștergerea întregului tabel putem utiliza comanda:

**DROP TABLE nume_tabel;**

Dacă în cazul instrucțiunii drop, cuvantul cheie table este înlocuit cu database sau user, atunci se va sterge baza de date sau utilizatorul menționat.

De asemenea la crearea unui tabel sau ulterior se pot declara anumite constrângeri. Sintaxa pentru declararea constrângerilor este:

```
CREATE TABLE nume_tabel(
coloana1 tip_date constrangere,
coloana2 tip_date constrangere,
coloana3 tip_date constrangere,
……
);
```

În Tabelul 3.2 se vor menționa principale constrângeri ce pot vi aplicate atributelor unei entități.

Tabelul 3.2. Constrângeri

| | |
|---|---|
| NOT NULL | Nu permite valori nule în coloana respectivă |
| UNIQUE | Valorile din fiecare câmp al coloanei trebuie să fie unice |
| PRIMARY KEY | Cheie primara |
| FOREIGN KEY | Cheie externă |
| CHECK | Verifică dacă valoarea din coloană satisface o anumită condiție |
| DEFAULT | Setează o valoare implicită dacă nu se specifică altfel |
| CREATE INDEX | Crează index pentru accesul rapid la informație |
| AUTO_INCREMENT | Incrementare automată a valorii |

După crearea tabelelor, acestea trebuie populate. Pentru introducerea de valori în tabel utilizăm comanda:

**INSERT INTO nume_tabel (coloana1, coloana2, coloana, ...)**
**VALUES (valoare1,valoare2, valoare3, ...);**

Dacă se introduc valori în toate coloanele, atunci dacă acestea se introduc în ordinea în care se găsesc în tabel se poate omite paranteza cu numele coloanelor. O dată datele introduse în tabel se pot modifica utilizând comanda update.

**UPDATE nume_tabel**
**SET coloana1=valoare1, coloana2=valoare, ...**
**WHERE conditii;**

Prin intermediul comenzi update nu trebuie modificate toate valorile dintr-o înregistrare. De asemenea utilizând cuvântul cheie where, se pot seta un set de condiții care definesc înregistrările unde se doresc modificările.

Dacă se dorește ștergerea unei sau mai multor înregistrări se va folosi comanda delete.

**DELETE FROM nume_tabel WHERE conditii;**

Evident că și în cadrul comenzi delete, trebuie menționate un set de condiții deoarece altfel se vor șterge toate înregistrările.

Pentru regăsi informația introdusă în baza de date se utilizează comanda select. În cadrul acestei comenzi se menționează numele coloanelor și a tabelului care stochează informația și de asemenea se pot menționa un set de condiții pentru trierea informației.

**SELECT coloana1, coloana2, ...**

**FROM nume_tabel**
**WHERE conditii;**

Dacă se dorește regăsirea întregii înregistrări (adică toate coloanele din tabel) atunci se poate utiliza simbolul '*' pentru a nu menționa numele tuturor coloanelor din tabel.

Pentru realizarea setului de condiții suplimentare se pot folosi atât operatori de comparație (Tabelul 3.3) cât și operatorii logici (Tabelul 3.4).

Tabelul 3.3. Operatori de comparație

| | |
|---|---|
| = | egal |
| < | mai mic |
| > | mai mare |
| <= | mai mic sau egal |
| >= | mai mare sau egal |
| < > sau != | diferit |
| BETWEEN | într-un anumit interval |
| LIKE | asemănător |
| IN | aparține unei mulțimi |

Tabelul 3.4. Operatori logici

| | |
|---|---|
| AND | și |
| OR | sau |
| NOT | negație |

Comenzile MySQL prezentate servesc la crearea, popularea, modificarea, ștergerea și regăsirea informației din baza de date. Pe lângă comenzile de bază prezentate, sistemul de gestiune a bazelor de date MySQL mai pune la dispoziție comenzi pentru administrarea bazei de date cum ar fi crearea de utilizatori, acordarea de privilegii, crearea cheilor externe, importul și exportul bazelor de date precum și comenzi pentru gestionarea serverului de baze de date.

## 3.5 Gestionarea unei baze de date MySQL prin comenzi de tip text

Pentru gestionarea unei baze de date MySQL prin linii de comnadă, trebuie să avem acces la serverul bazei de date (fie local, fie de la distanță). Conectarea se poate realiza prin consolă (shell, terminal) care permite transferul datelor de tip text.

```
# mysql -u root -p
Enter password: *************
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Prin utilizarea comenzi *mysql* se va realiza conexiunea la serverul bazei de date. În cazul de față conectarea la serverul MySQL se va realiza cu utilizatorul *root* (administrator) deoarece avem nevoie ca utilizatorul să dețină drepturi pentru crearea unei baze de date respectiv a unui nou utilizator. Parametru *–u* este utilizat pentru a preciza numele utilizatorului, iar parametrul *–p* pentru a introduce parola.

Dacă datele introduse sunt corecte se va realiza conexiunea la serverul MySQL. Se observă că nu este selectată nici o bază de date (none). La acest moment nu este necesară selecția unei baze de date deoarece dorim doar crearea unui nou utilizator și a unei noi baze de date.

```
MariaDB [(none)]> CREATE USER 'testare'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.003 sec)
```

Pentru crearea unui nou utilizator am folosit comanda *CREATE USER* menționând numele noului utilizator (testare), modul de conectare (localhost) adică îi este permisă doar conectarea locală, și parola acestuia. Ulterior utilizatorul își poate schimba parola.

S-a creat și o bază de date numită *informatica* utilizând comanda *create database*.

```
MariaDB [(none)]> CREATE DATABASE informatica;
Query OK, 1 row affected (0.001 sec)
```

În continuare trebuie șa îi acordam privilegii utilizatorului asupra noi baze de date create. Pentru acordarea privilegiilor se utilizează comanda *GRANT* . In cazul de față acordăm toate privilegiile utilizatorului *testare* asupra bazei de date *informatica*. Se observă că doar pentru conexiunea locală îi sunt acordate privilegii. Dacă utilizatorul se conectează de la distanță atunci el nu are nici un privilegiu asupra acestei baze de date.

```
MariaDB [(none)]> GRANT ALL ON informatica.* TO 'testare'@'localhost';
Query OK, 0 rows affected (0.013 sec)
```

În acest moment ne vom deconecta de la server. Deconectarea se realizează prin intermediul comenzi *exit*.

```
MariaDB [(none)]> exit
Bye
```

Reconectarea la server se va face cu noul utilizator. După introducere credențialelor se observă că s-a realizat conectarea la server dar încă nu s-a ales nici o bază de date.

```
# mysql -u testare -p
Enter password: ********
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Se pot afișa toate bazele de date asupra cărora utilizatorul are privilegii prin utilizarea comenzi *show databases.* Se observă că utilizatorul curent are acces la trei baze de  date: informatica (creată anterior), information_schema și test, care sunt baze de date generale la care toți utilizatori primesc privilegii.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| informatica        |
| information_schema |
| test               |
+--------------------+
3 rows in set (0.001 sec)

MariaDB [(none)]>
```

Se alege utilizarea bazei de date informatica utilizând comanda use. O dată selectată baza de date se observă că și prompter-ul se modifică afișând numele bazei de date alese.

```
MariaDB [(none)]> use informatica;
Database changed
MariaDB [informatica]>
```

Petru a afișa tabelele conținute în baza de date curentă se utilizează comanda *show tables*. Se observă că nu există tabele în baza de date curentă.

```
MariaDB [informatica]> show tables;
Empty set (0.001 sec)

MariaDB [informatica]>
```

Pentru crearea unui tabel se folosește comanda *create table*. Se va crea tabelul numit *studenti* care are patru coloane (atribute) și anume:

- *id* care este de tip întreg, nu poate fi nul, și se auto incrementează;
- *nume* care este de tip sir de maxim 20 caractere și nu poate fi nul;
- *prenume* care este de tip sir de maxim 20 caractere;
- *specializare* tot de tip sir de maxim 20 caractere.

```
MariaDB [informatica]> CREATE TABLE Studenti (id int(5) NOT NULL AUTO_INCREMENT,
    -> nume varchar(20) NOT NULL,
    -> prenume varchar(20),
    -> specializarea varchar(20),
    -> PRIMARY KEY (id),
    -> UNIQUE (nume)
    -> );
Query OK, 0 rows affected (0.019 sec)
```

Se observă că atributului *id* îi este aplicată o cheie primară, iar atributului *nume* îi este aplicată o cheie unică.

S-a creat astfel tabelul *Studenti*. Dacă dorim să vizualizăm structura acesturia se va folosi comanda *describe nume_tabel.* Se observă că tabelul creat corespunde din punct de vedere structurii cu ce am dorit.

```
MariaDB [informatica]> show tables;
+-----------------------+
| Tables_in_informatica |
+-----------------------+
| studenti              |
+-----------------------+
1 row in set (0.001 sec)

MariaDB [informatica]> describe studenti
    -> ;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id            | int(5)      | NO   | PRI | NULL    | auto_increment |
| nume          | varchar(20) | NO   | UNI | NULL    |                |
| prenume       | varchar(20) | YES  |     | NULL    |                |
| specializarea | varchar(20) | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
4 rows in set (0.015 sec)

MariaDB [informatica]> select * from studenti;
Empty set (0.016 sec)
```

Pentru introducerea de informaţi în tabel se va folosi comanda *insert into.* Se vor insera două rânduri (tupluri).

```
MariaDB [informatica]> INSERT INTO studenti Values('','POPESCU','ION','Inginerie Medicala');
Query OK, 1 row affected, 1 warning (0.013 sec)

MariaDB [informatica]> INSERT INTO studenti Values('','IONESCU','MARIA','Inginerie Mecanica');
Query OK, 1 row affected, 1 warning (0.012 sec)

MariaDB [informatica]> select * from studenti;
+----+---------+---------+--------------------+
| id | nume    | prenume | specializarea      |
+----+---------+---------+--------------------+
|  1 | POPESCU | ION     | Inginerie Medicala |
|  2 | IONESCU | MARIA   | Inginerie Mecanica |
+----+---------+---------+--------------------+
2 rows in set (0.000 sec)
```

Se observă că pentru fiecare înregistrare datele au fost introduse în ordinea din tabel, astfel nefiind necesar a menționa numele coloanelor. Trebuie precizat însă, cu toate că atributul *id* are setat auto increment, este necesar să menționăm poziția acestuia. Acest lucru este obligatoriu deoarece numărul de valori introduse trebuie să coincidă cu numărul de atribute ale tabelului.

Pentru a vizualiza informația introdusă în tabel s-a utilizat comanda *select.* S-a utilizat simbolul '*' pentru a selecta informația din toate coloanele tabelului *studenti* fără a meționa numele atributelor.

Pentru a modifica informațiile dintr-un tabel se folosește comanda *update.* Se va modifica în tabelul *studenti* , *prenumele* pentru înregistrarea care are *id*-ul identic cu 2, noua valoare fiind 'MARIA IOANA'.

```
MariaDB [informatica]> UPDATE studenti SET prenume='MARIA IOANA' WHERE id=2;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [informatica]> select * from studenti;
+----+---------+-------------+--------------------+
| id | nume    | prenume     | specializarea      |
+----+---------+-------------+--------------------+
|  1 | POPESCU | ION         | Inginerie Medicala |
|  2 | IONESCU | MARIA IOANA | Inginerie Mecanica |
+----+---------+-------------+--------------------+
2 rows in set (0.000 sec)
```

Dacă ar fi fost mai multe înregistrări (tupluri) cu atributul *id*=2, atunci la toate înregistrările atributul *prenume* va avea noua valoare. Pentru exemplul prezentat nu ar fi fost posibil să avem mai multe înregistrări cu același *id* deoarece acest atribut are setat auto increment precum și cheie primară.

Pentru a șterge o înregistrare se utilizează comanda *delete.* Acestă comandă obligă la utilizarea condițiilor pentru a nu șterge toate datele S-a șters din tabelul *studenti* doar înregistrarea a cărei *nume* era 'POPESCU'.

```
MariaDB [informatica]> DELETE FROM studenti WHERE nume='POPESCU';
Query OK, 1 row affected (0.013 sec)

MariaDB [informatica]> select * from studenti;
+----+---------+-------------+--------------------+
| id | nume    | prenume     | specializarea      |
+----+---------+-------------+--------------------+
|  2 | IONESCU | MARIA IOANA | Inginerie Mecanica |
+----+---------+-------------+--------------------+
1 row in set (0.000 sec)
```

Și în acest caz s-a stres o singură înregistrare deoarece atributul *nume* are setată cheie unică, adică nu pot exista două înregistrări care să aibă același *nume.*

Pentru a modifica structura tabelului se va folosi comanda *alter table.* Pentru exemplificare se va modifica structura tabelului *studenti* prin eliminarea coloanei *specializare.*

Pentru a elimina un atribut, o entitate, o bază de date sau un utilizator cuvântul cheie utilizat este *drop.*

Utilizând comanda *alter table* în tandem cu cuvântul cheie *drop* s-a eliminat coloana *specializare* din tabelul *studenti*.

Pentru eliminarea unui tabel, baze de date sau utilizator cuvântul cheie *drop* este urmat de cuvântul *table, database* sau *user*.

```
MariaDB [informatica]> ALTER TABLE studenti DROP specializarea;
Query OK, 0 rows affected (0.017 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [informatica]> select * from studenti;
+----+---------+-------------+
| id | nume    | prenume     |
+----+---------+-------------+
|  2 | IONESCU | MARIA IOANA |
+----+---------+-------------+
1 row in set (0.000 sec)
```

După ce s-a utilizat comanda *drop table* pentru a elimina tabelul *studenti* se observă că o dată cu eliminarea tabelului s-a pierdut și toată informația existentă în acesta.

```
MariaDB [informatica]> DROP TABLE studenti;
Query OK, 0 rows affected (0.017 sec)

MariaDB [informatica]> show tables;
Empty set (0.001 sec)
```

Pentru ștergerea utilizatorilor și a bazelor de date este necesar a avea privilegii administrative. Exemplificarea comenzilor de eliminare a bazei de date

*informatica* precum și a utilizatorului *testare* s-a revenit la utilizatorul *root* care are drepturi administrative.

```
MariaDB [(none)]> DROP DATABASE informatica;
Query OK, 0 rows affected (0.013 sec)
```

```
MariaDB [(none)]> DROP USER 'testare'@'localhost';
Query OK, 0 rows affected (0.012 sec)
```

Cuvintele cheie utilizate în comenzile prezentate anterior sunt scrise cu majuscule. Nu este obligatoriu acest lucru dar majoritatea utilizatorilor folosesc această convenție deoarece sintaxa liniilor de comandă se înțelege mai ușor în acest fel.

Se poate observa de asemenea utilizarea simbolului ';' la terminarea fiecărei linii de comanda. Sistemul de gestiune MySQL permite scriere liniilor de comandă pe mai multe linii. Astfel executarea unui comenzi se realizează doar în momentul întâlniri simbolului ';'. Acest lucru este benefic deoarece comanda poate fi împărțită pe mai multe rânduri astfel încât să efectuăm cât mai ușor eventualele corecturi de sintază.

## 3.6 Gestionarea unei baze de date utilizând interfața phpMyAdmin

phpMyAdmin este o aplicație de gestionare a bazelor de date MySQL, dezvoltată în PHP, de tip open-source. Acesta oferă o interfață web pentru a gestiona bazele de date MySQL și pentru a efectua sarcini precum crearea, modificarea și ștergerea atât a tabelelor cât și a informației introduse. De asemenea permite și, gestionarea utilizatorilor și a privilegiilor acestora.

phpMyAdmin poate fi instalat pe un server web astfel putând fi accesat prin intermediul unui browser web. Acesta interfață grafică pentru gestionarea bazelor de date MySQL, facilitează accesul utilizatorii care nu sunt familiarizați cu comenzi SQL.

Principalele caracteristici ale phpMyAdmin sunt:

- permite crearea, modifica și ștergerea tabelelor (structura tabelelor);
- permite gestionarea relațiilor de legătură dintre tabele;
- permite gestiona utilizatorii și privilegiilor acestora;
- permite importul și exportul înregistrărilor din baza de date prin intermediul fișierelor externe;
- permite execuția interogărilor SQL și vizualiza rezultatelor interogări;
- permite efectuarea de operații avansate, cum ar fi optimizarea bazelor de date și vizualizarea informațiilor de performanță.

phpMyAdmin este un instrument utilizat de dezvoltatorii de aplicații web și administratorii de baze de date pentru a gestiona bazele de date MySQL.

Pentru a accesa aplicația phpMyAdmin trebuie să cunoaștem numele sau adresa serverului unde a fost instalată aplicația. Într-un browser web (nu contează ce browser se utilizează), se va introduce adresa sau numele serverului, urmând să apară pagina de conectare la aplicație (Fig. .3.4).

Pentru conectare trebuie introduse numele și parola utilizatorului. Tuturor utilizatorilor declarați pe serverul de MySQL le este permis accesul și la interfața web a aplicației phpMyAdmin.

După conectare fiecare utilizator va avea acces la bazele de date asupra cărora are privilegii. Pentru siguranță, utilizatorului root (administrator) îi este permisă conectarea la serverul MySQL doar local (nu prin intermediul internetului). Acest lucru poate fi modificat, dar nu este recomandat deoarece crește riscul unui atac cibernetic asupra serverului bazelor de date.

Utilizatorul root are privilegii pentru gestionarea altor utilizatori. Astfel se va crea un nou utilizator denumit *testare* (Fig. 3.5).


Fig. 3.4. Interfață conectare aplicație phpMyAdmin

Pentru un nou utilizator trebuie să introducem un nume și o parolă. Forma utilizată pentru numele utilizatorului și parolă este impusă de serverul de baze de date. Dacă în câmpul *host* introducem simbolul % atunci utilizatorul se poate conecta la serverul MySQL de oriunde. Dacă dorim să limităm accesul utilizatorului atunci se va menționa în câmpul *host* adresa sau grupul de adrese de la care îi permitem conectare.

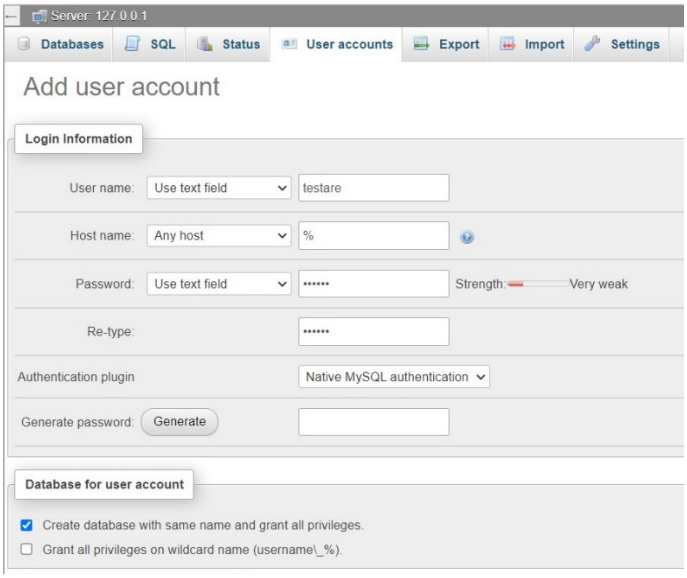



Fig. 3.5. Crearea unui nou utilizator

In fereastra de creare utilizator (Fig. 3.5) există posibilitatea de a bifa opțiunea de a crea o bază de date cu același nume ca utilizatorul și acordarea tuturor privilegiilor asupra acesteia. Având creat noul utilizator, putem utiliza credențialele acestuia (Fig. 3.6) pentru gestiona baza de date.


Fig. 3.6. Pagina web de conectare la phpMyAdmin

Se observă că pentru noul utilizator testare avem disponibile doar bazele de date generice disponibile tuturor utilizatorilor precum și baza de date creată asupra căreia are toate privilegiile (Fig. 3.7).

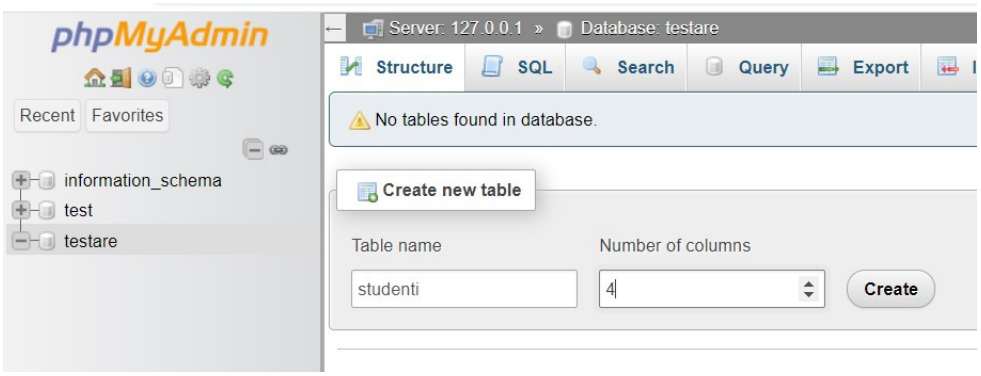



Fig. 3.7. Interfața utilizatorului *testare*

Baza de date care are același nume cu utilizatorul a fost creată dar nu are nici o informație introdusă. Primul pas ce trebuie efectuat este crearea de tabele. Se vor crea două tabele numite *studenti* respectiv *specializare.*

Se creează două tabele pentru a putea demonstra și modul în care se definesc relațiile de legătură între acestea. Tabelul *studenti* va avea un atribut numit *specializare* care va fi conectat cu specializările disponibile în tabelul *specializari*.

Tabelul *studenti* va avea patru atribute (coloane)(Fig. 3.8):

- id – numar intreg maxim 11 caractere, auto increment, cheie primară;
- nume – șir de maximum 20 de caractere , cheie unică
- prenume - șir de maximum 20 de caractere
- specializare - șir de maximum 20 de caractere, indexat

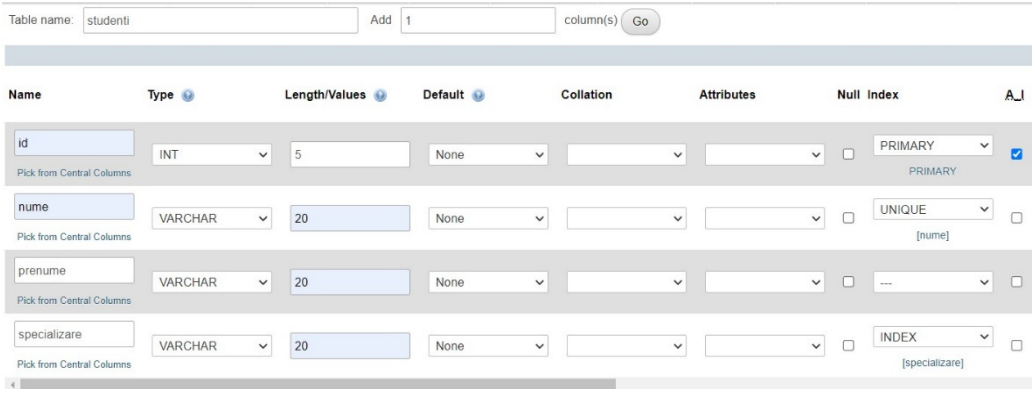



Fig. 3.8. Crearea tabelului (entitate) studenti

La crearea unui tabel trebuie să denumim fiecare atribut (coloană) precum să și precizăm tipul acestuia.

În Tabelul 3.5 sunt prezentate toate opțiunile disponibile pentru tipul atributelor.

Tabelul 3.5. Tipuri de atribute

| **Tip - Caracter** | |
|---|---|
| CHAR(lungime) | un șir de caractere cu o *lungime* cuprinsă între 0 și 255 caractere; implicit lungimea este 1; poate conține litere, numere sau caractere speciale. |
| VARCHAR(lungime) | un șir de caractere cu o *lungime* cuprinsă între 0 și 65535 caractere; implicit lungimea este 1; poate conține litere, numere sau caractere speciale. |
| BINARY(lungime) | echivalent cu char(), doar că stochează șiruri de octeți; *lungime* reprezentând lungimea coloanei în octeți; implicit lungimea este 1. |
| VARBINARY(lungime) | echivalent cu varchar(), doar că stochează șiruri de octeți; *lungime* reprezentând lungimea coloanei în octeți; |
| TINYBLOB | utilizat pentru BLOBs (Binary Large OBjects); lungimea maximă fiind 255 octeți |
| TINYTEXT | șir de maxim 255 caractere; acceptă orice caractere. |
| TEXT(lungime) | șir de maxim 65535 caractere; acceptă orice caractere. |
| BLOB(lungime) | utilizat pentru BLOBs (Binary Large OBjects); lungimea maximă fiind 65535 octeți. |
| MEDIUMTEXT | șir de maxim 16777215 caractere; acceptă orice caractere. |
| MEDIUMBLOB | utilizat pentru BLOBs (Binary Large OBjects); lungimea maximă fiind 16777215 octeți. |
| LONGTEXT | șir de maxim 4294967295 caractere; acceptă orice caractere. |
| LONGBLOB | utilizat pentru BLOBs (Binary Large OBjects); lungimea maximă fiind 4294967295 octeți. |
| ENUM(val1, val2, val3, ...) | enumerare; se definește un tabel cu maxim 65535 de valori; dacă valoarea introdusă nu se află în listă atunci câmpul va fi setat pe nul. |
| SET(val1, val2, val3, ...) | se definește o listă cu maxim 64 de valori din care trebuie aleasă o valoare. |
| **Tip numeric** | |
| BIT(lungime) | o variabila de tip bit; *lungimea* poate fi între 1 și 64; implicit lungimea este 1. |
| TINYINT(lungime) | un număr întreg mic cuprins între -128 și 127; dacă intervalul este fără semn atunci valorile sunt |

| | |
|---|---|
| | cuprinse între 0 și 255; parametrul *lungime* are valoarea maximă este de 255. |
| BOOL | zero este considerat fals; orice valoare diferită de zero este considerată adevărată. |
| BOOLEAN | echivalent cu BOOL |
| SMALLINT(lungime) | un număr întreg mic cuprins între -32768 și 32767; dacă intervalul este fără semn atunci valorile sunt cuprinse între 0 și 65535; parametrul *lungime* are valoarea maximă este de 255. |
| MEDIUMINT(lungime) | un număr întreg cuprins între -8388608 și 8388607; dacă intervalul este fără semn atunci valorile sunt cuprinse între 0 și 16777215; parametrul *lungime* are valoarea maximă este de 255. |
| INT(lungime) | un număr întreg mic cuprins între -2147483648 și 2147483647; dacă intervalul este fără semn atunci valorile sunt cuprinse între 0 și 4294967295; parametrul *lungime* are valoarea maximă este de 255. |
| INTEGER(lungime) | echivalent cu int(). |
| BIGINT(lungime) | un număr întreg mic cuprins între -9223372036854775808 și 9223372036854775807; dacă intervalul este fără semn atunci valorile sunt cuprinse între 0 și 18446744073709551615; parametrul *lungime* are valoarea maximă este de 255. |
| FLOAT(lungime, d) | un număr cu virgulă mobilă; parametru *lungime* reprezintă câte cifre are numărul, iar parametru *d* câte cifre sunt după virgulă. |
| FLOAT(p) | un număr cu virgulă mobilă; funcție de parametrul *p* MySQL-ul decide dacă numărul este *float* dau *double*; dacă *p* cuprins între 0 și 24 numărul va fi *float* iar dacă *p* este cuprins între 25 și 53 atunci numarul va fi *double.* |
| DOUBLE(lungime, d) | un număr cu virgulă mobilă; parametru *lungime* reprezintă câte cifre are numărul, iar parametru *d* câte cifre sunt după virgulă. |
| DECIMAL(lungime, d) | un număr cu virgulă fixă; parametru *lungime* reprezintă câte cifre are numărul, iar parametru *d* câte cifre sunt după virgulă; valoarea maximă |

| | |
|---|---|
| | pentru *lungime* este 64 iar pentru *d* este 30; implicit *lungime*=10 și *d*=0. |
| DEC(lungime, d) | echivalent cu DECIMAL() |
| **Tip dată și timp** | |
| DATE | data în format *YYYY-MM-DD*; acceptă date cuprinse în intervalul '1000-01-01' până la '9999-12-31' |
| DATETIME(fsp) | o combinație între dată și timp în formatul *YYYY-MM-DD hh:mm:ss*; acceptă valori cuprinse între '1000-01-01 00:00:00' și '9999-12-31 23:59:59'. |
| TIMESTAMP(fsp) | stochează data și timpul printr-un număr ce reprezintă numărul de secunde din '1970-01-01 00:00:00' UTC; acceptă valorile cuprinse în intervalul '1970-01-01 00:00:01' UTC până la '2038-01-09 03:14:07' UTC. |
| TIME(fsp) | acceptă o variabilă de tip timp în formatul *hh:mm:ss*. acceptă valori cuprinse între -838:59:59 și 838:59:59 |
| YEAR | acceptă o valoare de tip an formată din 4 cifre; acceptă valori cuprinse între 1901 și 2155, precum și 0000. |

Structura tabelului studenti (Fig. 3.9) poate fi modificată prin apăsarea butonului de modificare (change) sau șterge prin apăsarea butonului *drop*. Atributul *id* este setat ca cheie primară. Pentru a nu avea studenți înregistrați cu acelaşi nume, atributul *nume* are setată cheia unică. Atributul *specializare* este indexat deoarece se dorește stabilirea unui legături a acestuia cu specializările disponibile în tabelul *specializari*.

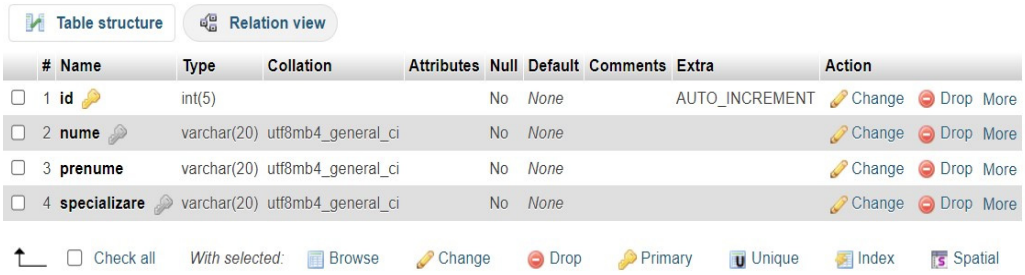



Fig. 3.9. Structura tabelului studenți

Structura tabelului specializări (Fig. 3.10) cuprinde trei atribute:

- *id* – de tip întreg; cheie primară; auto increment;
- *specializare* – sir de maxim 20 caractere; cheie unică;
- *descriere* – tip text.

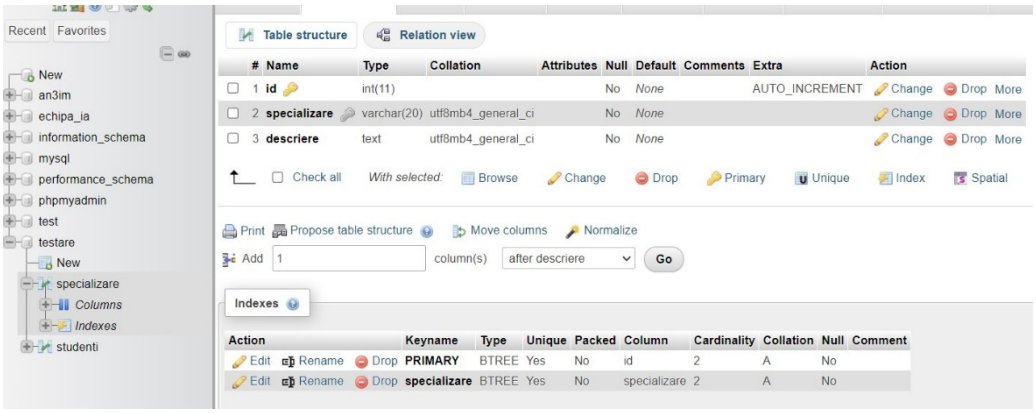



Fig. 3.10. Structura tabelului specializare

De asemenea se poate observa că sunt afișate toate cheile setate pentru tabel. Fiecare cheie poate fi modificată sau ștearsă.

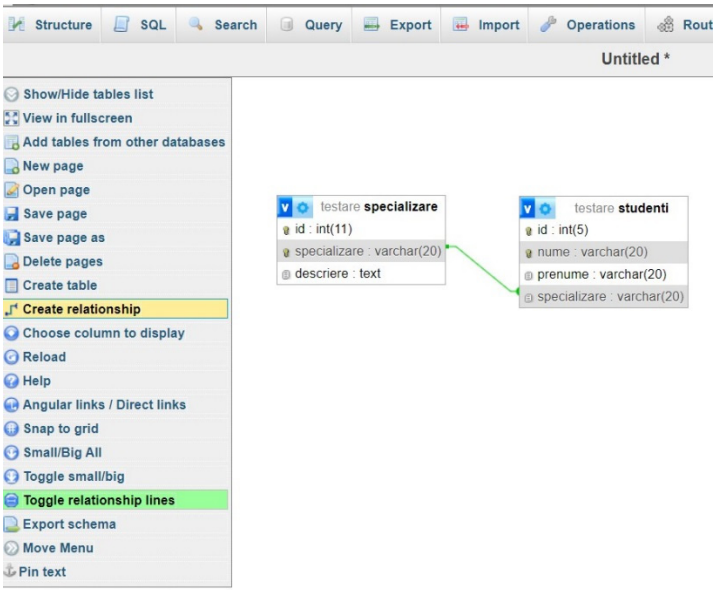



Fig. 3.11. Relația de legătură dintre tabele

Pentru a stabili relațiile de legătură dintre tabelele bazei de date trebuie selectată în primul rând baza de date, apoi se accesează opțiunea design. Se va

deschide fereastra de design (Fig. 3.11) unde vor apărea toate tabelele conținute în baza de date selectată.

Pentru a realiza o relație de legătură, se accesează din meniul ce este disponibil în stânga opțiunea *create relationship*. Apoi cu ajutorul mouse-ului se vor selecta din tabele atributele (coloanele) între care se dorește crearea relației de legătură. Prima oară trebuie selectat atributul părinte apoi cel copil.

La crearea unei relații de legătură trebuie stabilit ce se întâmplă cu informația stocată în tabelele între care se realizează legătura dacă această relație de legătură este modificată sau ștearsă

La modificarea sau ștergerea unei relații de legătură (Fig. 3.12) avem următoarele opțiuni disponibile:

- cascade – permite modificarea sau ștergerea rândului din tabelul părinte și modifică în concordanță cu acestea informația din tabelul copil;
- set null - permite modificarea sau ștergerea rândului din tabelul părinte și setează ca nul câmpul aferent din tabelul copil;
- no action – nu permite modificarea relației de legătură;
- restrict – identic cu opțiunea no action doar că restricția este impusă de serverul MySQL.


Fig.3.12. Opțiuni ale relațiilor de legătură

Este recomandat ca înainte de introducerea informațiilor în tabele să fie definite toate relațiile de legătură. Dacă ulterior se dorește creare altor relații de legătură (acest lucru este posibil) trebuie avut grija ca informația existentă în tabele să corespundă cu noua relație de legătură.

De asemenea este esențial ca tipul atributelor, între care se face legătura, să fie identice. Dacă atributele au tipuri diferite nu este permisă realizarea unei legături externe.

În cazul de față introducerea informației în tabele s-a efectuat după realizarea relației de legătură. Tabelul *specializare* este de tip părinte iar legătura se stabilește între atributul *specializare* din acest tabel și atributul *specializare* din tabelul *studenti* (tabel copil).

| | id | specializare | descriere |
|---|---|---|---|
| Edit Copy Delete | 1 | Inginerie Medicala | Specializare la intersectia domeniilor de medicina... |
| Edit Copy Delete | 2 | Inginerie Mecanica | Inginer mecanic |

Check all With selected: Edit Copy Delete Export

Fig. 3.13. Informația din tabelul specializare

Astfel primele informații introduse au fost în tabelul *specializare* (Fig. 3.13) deoarece, în acest mod informația introdusă este accesibilă pentru a fi selectată în momentul în care introducem date în tabelul *studenti*. Cu alte cuvinte în tabelul *specializare* avem o listă a specializărilor disponibile iar în momentul introduceri datelor unui nou student în tabelul *studenti* la atributul (coloana) *specializare* această listă va fi disponibilă, utilizatorul selectând opțiunea dorită.

În tabelul *studenti* (Fig. 3.14) au fost introduse două persoane. Numele și prenumele a fost introduse de la tastatură, specializarea a fost selectată iar marca s-a completat automat fiind cu auto increment.


Fig. 3.14. Informația din tabelul studenti

Dacă pentru un atribut (Fig. 3.15) este selectată opțiunea auto increment atunci nu este necesar completarea acestui câmp deoarece este incrementat automat. De asemenea pentru toate atributele care au chei externe, informația este disponibilă prin meniuri tip *DropDown* (Fig. 3.15).

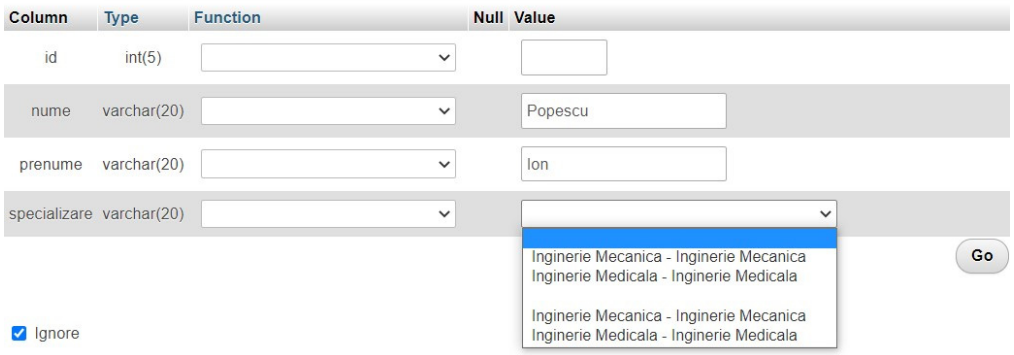



Fig. 3.15. Adăugarea unei înregistrări

Aplicația phpMyAdmin permite de asemenea și modificarea datelor introduse în tabele. Evident că modificările trebuie să fie în concordanță cu structura tabelelor și cu relațiile de legătură existente.


Fig. 3.16. Informația modificată din tabelul studenti

Pentru exemplul prezentat se observă că s-a realizat modificarea prenumelui studentului precum și a specializări (Fig. 3.16) Prenumele putea fi modificat sub orice formă, doar specializarea trebuie să existe în lista cu specializări. Pentru a completa câmpul specializare nu este obligatoriu selecția acesteia, fiind posibilă și introducerea de la tastatură doar că fiind un limbaj *case sensitive* (face diferență între litere mici și mari) textul introdus trebuie sa corespundă exact cu cel din lista de specializari.

# 4 DEZVOLTAREA UNEI APLICAȚII DE GESTIONARE A UNEI BAZE DE DATE

Există mai multe limbaje de programare în care se pot dezvolta aplicații care pot gestiona o bază de date. Majoritatea limbajelor de programare pun la dispoziție funcții care permit gestionarea bazelor de date. Cele mai utilizate limbaje de programare care facilitează accesul la baze de date sunt:

- **PHP** (Hypertext Preprocessor) - este un limbaj de programare open-source utilizat pentru a dezvolta aplicații web și este frecvent utilizat împreună cu baze de date MySQL. PHP poate fi utilizat pentru a interoga bazele de date MySQL, pentru a adăuga, actualiza sau șterge date și pentru a gestiona utilizatorii și permisiunile de acces [54].
- **Python** - este un alt limbaj de programare popular utilizat pentru dezvoltarea de aplicații web și pentru gestionarea bazelor de date MySQL. Python oferă module și librării pentru a interacționa cu bazele de date MySQL [55].
- **Java** - este un alt limbaj de programare popular utilizat pentru a dezvolta aplicații web și pentru a gestiona bazele de date MySQL. Java oferă clase și biblioteci pentru a interacționa cu bazele de date MySQL, inclusiv clasele JDBC (Java Database Connectivity) și MySQL Connector/J.[56]
- **Ruby** - este un limbaj de programare open-source utilizat pentru dezvoltarea de aplicații web și pentru gestionarea bazelor de date MySQL. Ruby oferă suport pentru interogarea bazei de date MySQL prin intermediul gem-ului MySQL2.[57]

Pe lângă limbajele de programare, majoritatea programelor (software) inginerești pun la dispoziție pachete de funții pentru comunicarea cu bazele de date.

Unul dintre cele mai puternice și cunoscute programe inginerești este Matlab, care pune la dispoziția utilizatorilor pachetul *database toolbox* pentru gestionarea bazelor de date.

În cadrul acestui capitol se vor prezenta etapele necesare pentru dezvoltarea unei aplicați cu interfață grafică utilizator dezvoltată în modulul *appdesigner* din programul Matlab.

Baza de date va fi salvată pe un server MySQL iar legătura dintre aplicația Matlab și baza de date MySQL se va realiza prin intermediul unui driver JDBC (Java Database Connectivity).

Aplicația va permite introducerea, modificarea și regăsirea informațiilor într-o baza de date MySQL, bază de date creată cu ajutorul aplicației phpMyAdmin.

## 4.1 Conectarea aplicației Matlab la baza de date MySQL

Pentru conectarea la baza de date se va folosi un driver JDBC care poate fi descărcat gratuit de pe internet. Implicit programul Matlab nu conține acest driver astfel că este obligatoriu descărcarea acestuia.

Aplicația Matlab necesită acces la acest driver. Se poate utiliza interfața grafică (Fig. 4.1) unde se definește locația driver-ului, precum și credențialele pentru conectarea la baza de date.

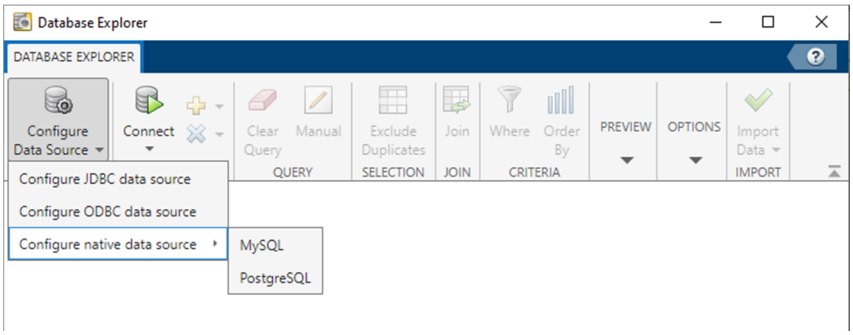



Fig. 4.1. Interfată grafică conectare la baza de date

O altă soluție de conectare la baza de date menționarea locației driver-ului java descărcat în fișierul *classpath.txt*. La pornirea programului Matlab se va încărca driver-ul și poate fi utilizat. Pentru aplicația ce va fi prezentată în continuare se folosi această soluție.

Pe lângă driver (face legătura dintre Matlab și MySQL) mai este necesar și pachetul *database toolbox* care pune la dispoziție funcțiile Matlab pentru gestionarea bazei de date.

Pachetul *database toolbox* (Fig. 4.2) poate fi descărcat de pe pagina oficială a aplicației Matlab [58].

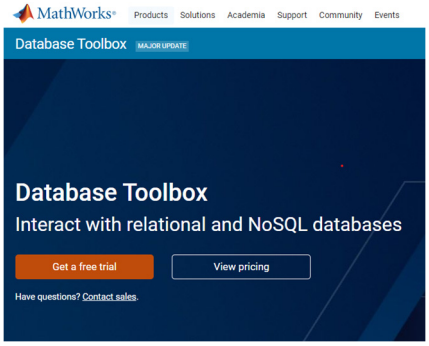



Fig. 4.2. Dabase Toolbox

După realizarea acestor setări s-a stabilit modul de comunicare cu baza de date.

## 4.2 Creare bazei de date în MySQL

Pentru crearea bazei de date se va folosi interfața grafică phpMyAdmin care facilitează crearea tabelelor și a legăturilor (chei externe) între acestea.

Baza de date va deservi o clinică. Informațiile stocate în baza de date fac referire la personalul medical, pacienți și rezultatele investigațiilor medicale.

Se vor crea cinci tabele (Fig. 4.3):

- specializare – conține specializările disponibile în cadrul clinici;
- pacienti – conține lista pacienților înscriși la clinică;
- medici_asistente – lista personalului medical;
- parole – tabel cu parolele angajatilor care le permite accesul acestora la aplicația dezvoltată;
- rapot – tabel cu rezultate investigațiilor medicale.

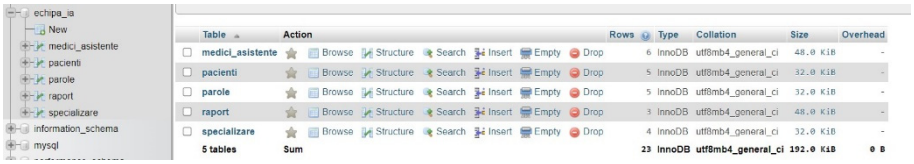



Fig. 4.3. Tabelele bazei de date

Tabelul *medici_asistente* are următoarea structură (Fig. 4.4):

- *id* – număr întreg; cheie primară; auto increment;
- *nume* – șir de maxim 20 caractere; cheie unică;
- *adresa* – șir de maxim 20 caractere;
- *specializare* – șir de maxim 20 caractere; indexat;
- *ma* – nr intreg de o cifră.

Coloana *id* reprezintă marca (id-ul) fiecărui angajat. Se utilizează setarea auto increment pentru a obține automat (nu trebuie introdus manual) marca. Deoarece mărcile reprezintă un număr unic atașat fiecărui membru se poate seta ca și cheie primară.

*Nume* este un șir de maxim 20 caractere care reprezintă numele fiecărui angajat. Cheia unică aplicată acestui atribut nu ne permite să introducem mai multe persoane cu același nume.

Coloana *adresa* este de tip șir de maxim douăzeci de caractere unde este stocată adresa corespunzătoare fiecărui membru.

*Specializare* este un șir de maxim 20 de caractere ce reprezintă specializarea personalului medical și este indexată deoarece se dorește realizarea unei relații externe cu lista specializărilor disponibile în tabelul *specializări*.

Coloana *ma* definește tipul de personal angajt și va avea valoarea 0 pentru asistentă și valoarea 1 pentru medic.

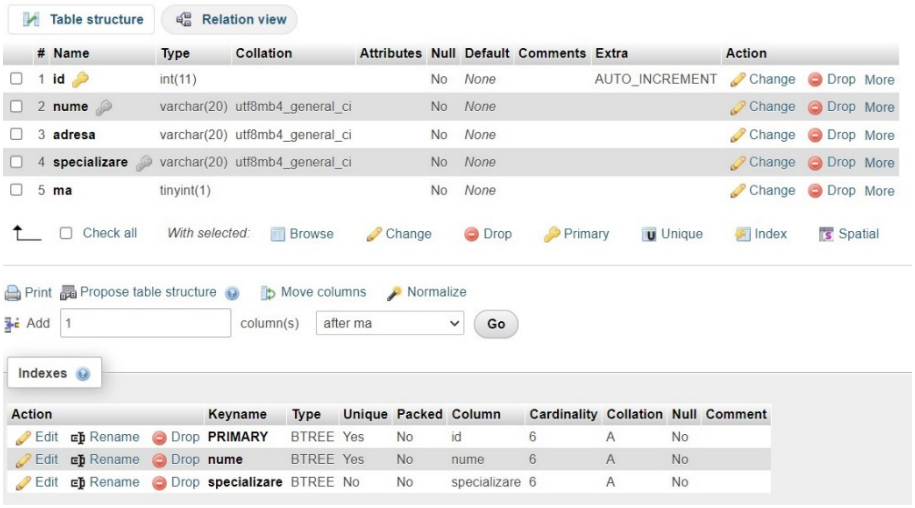



Fig. 4.4. Structura tabelului *medici_asistente*

Tabelul *pacienti* are următoarea structura (Fig. 4.5)

- *id* – număr întreg; cheie primară; auto increment;
- *nume* – şir de maxim 20 caractere; cheie unică;
- *adresa* – şir de maxim 20 caractere.

Numele coloanelor nu trebuie să fie neapărat diferite deoarece ele se află în tabele diferite. Doar în cazul coloanelor din același tabel nu este posibil să utilizăm același nume. Cele trei coloane ale tabelului *pacienti* sunt de același tip ca și cele din tabelul *medici_asistente.*

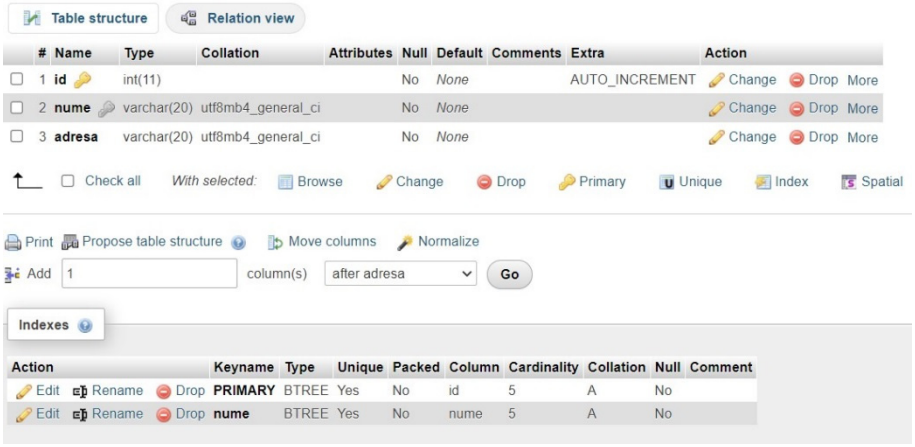



Fig. 4.5. Structura tabelului *pacienţi*

Tabelul *pacienţi* va conține lista pacienților înscriși la clinică precum și adresa de contact a acestora. Fiecare pacient va primi și o marcă (id) pentru a fi identificat în mod unic.

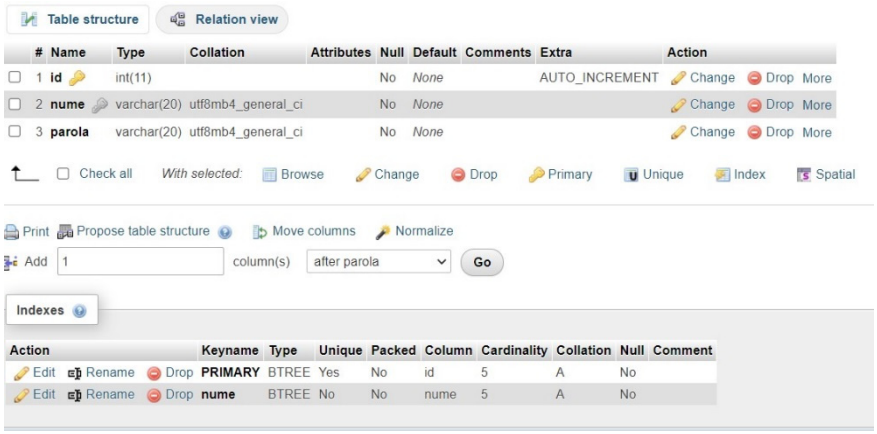



Fig. 4.6. Structura tabelului parole

Conectarea la interfața grafică utilizator a aplicației dezvoltate, se va realiza printr-o parolă. Stocarea acestor parole se va realiza în tabelul *parole.* Există posibilitatea ca parolele stocate să fie criptate dar acest lucru nu va fi implementat în cazul de față.

Tabelul *parole* are următoarea structură (Fig. 4.6):

- *id* – număr întreg; cheie primară; auto increment;
- *nume* – șir de maxim 20 caractere; cheie unică;
- *parola* – șir de maxim 20 caractere.

Coloanele *id* și *nume* sunt identice cu coloanele omoloage din tabelele prezentate. Coloana *parola* este un șir de maxim 20 caractere. Poate conține orice caracter, cifră sau caracter special. Nu sunt impuse reguli pentru formarea parolei (lungime, caractere speciale, litere mari sau mici).

Tabelul *specializare* are următoarea structură (Fig. 4.7):

- *id* – număr întreg; cheie primară; auto increment;
- *specializare* – șir de maxim 20 caractere; cheie unică.

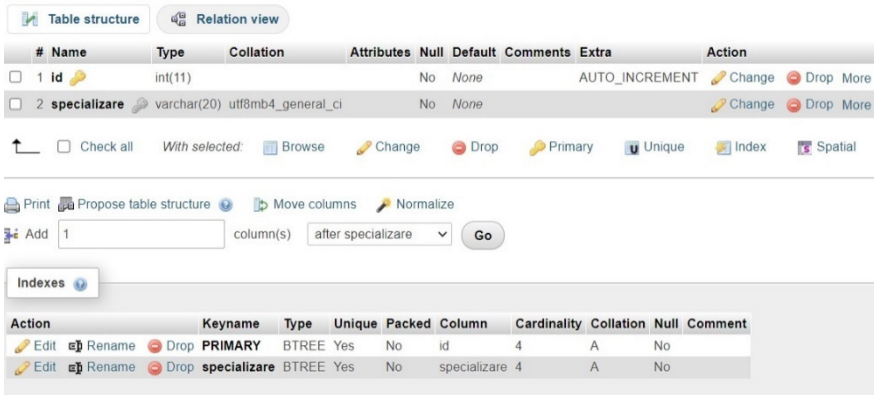



Fig. 4.7. Structura tabelului specializare

Tabelul *specializare* va conține lista cu specializări disponibile în clinică. Tabelul *raport* toate rezultate investigațiilor efectuate în cadrul clinici. Structura tabelului raport (Fig. 4.8) este:

- *id* – număr întreg; cheie primară; auto increment;
- *pacient* – șir de maxim 20 caractere; index;
- *medic* – șir de maxim 20 caractere; index;
- *data* – tip data;
- *diagnostic* – șir de maxim 20 caractere;
- *recomandari* – tip text.

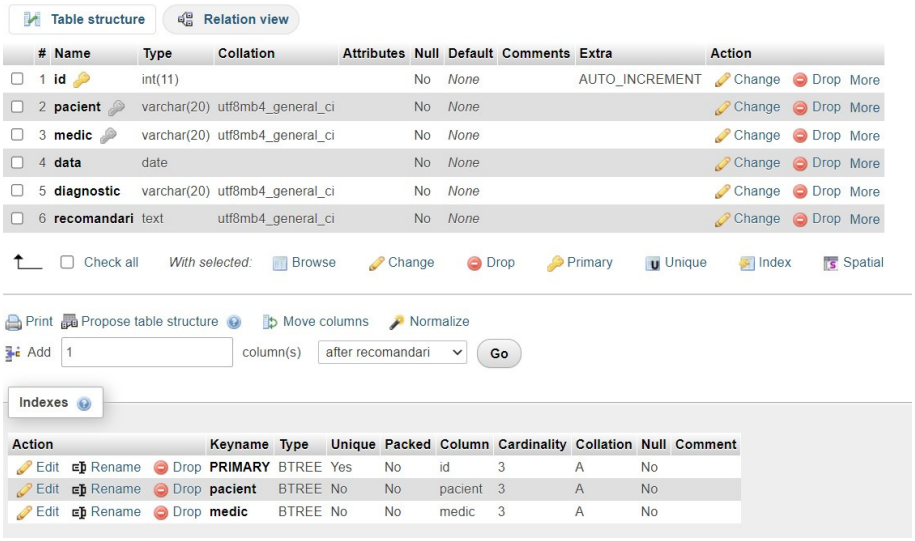



Fig. 4.8. Structura tabelului raport

În tabelul *raport* coloana *id* reprezintă numărul de înregistrare a raportului. Coloanele *pacient* și *medic* sunt indexate deoarece se dorește realizarea unei legături între acestea și lista pacienților respectiv a medicilor înscriși în clinică. Coloana data reprezintă data efectuării investigației. Diagnosticul și recomandările de tratament precizate de medic se vor stoca în coloanele *diagnostic* respectiv *recomandări*. Ținând cont că recomandări pot fi mai multe, tipul coloanei *recomandări* a fost ales ca text.

După generarea tuturor tabelelor trebuie realizate și legăturile dintre acestea. Pentru exemplul de față se stabilesc următoarele patru legături:

- *spacializare*.**specializare** – *medic_asistente*.**specializare**;
- *pacient*.**nume** – *raport*.**pacient**;
- *medic_asistente*.**nume** – *raport*.**medic**;
- *medic_asistente*.**nume** – *parole*.**nume**.

La enumerarea legăturilor, pentru evidențiere, s-a utilizat un text înclinat pentru numele tabelului și un text îngroșat pentru numele coloanei din tabel.

Pentru realizarea legăturilor s-a accesat opțiunea design (Fig. 4.9) din aplicația phpMyAdmin.

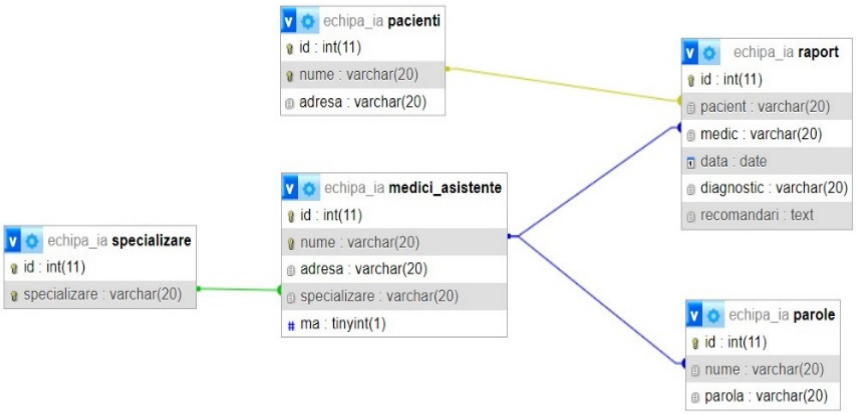



Fig. 4.9. Structura și relațiile de legătură a tabelelor din baza de date

De menționat, pentru realizarea legăturilor primul click se face pe coloana părinte apoi pe coloana copil. Se poate observa de asemenea că tipul coloanelor între care se stabilește relația de legătură este identic. Este esențial de ținut cont de aceste două observații pentru a realiza corespunzător legăturile dintre tabele.

## 4.3 Realizarea de interfețe grafice utilizator în Matlab

După creare bazei de date utilizând *phpMyAdmin* și stabilirea conexiuni *Matlab – MySQL* utilizând driver-ul *JDBC*, rămâne doar de realizat aplicația în programul Matlab.Pentru realizarea unei interfețe grafice utilizator, programul *Matlab* pune la dispoziție modulul *AppDesigner*. Acest modul facilitează realizarea interfețelor grafice utilizator prin plasarea simplă (drag & drop) a obiectelor (butoane, meniuri câmpuri, etc).Modului *AppDesigner* se poate fi apelat direct prin apăsarea butonului *Desin App* din tabul *APPS* din meniului principal al programului *Matlab*, sau prin comanda *appdesigner* scrisă în fereastra de comenzi.

Pentru această aplicație s-au creat patru interfețe grafice utilizator:

- interfața de conectarea la baza de date;
- interfața de conectare la aplicație;
- interfața de administrare;
- interfața pentru rapoarte.

Legătura între interfețele create este prezentată în Fig. 4.10. La pornirea aplicației prima interfață deschisă este cea de conectare la baza de date. Aici sunt introduse credențialele pentru conectarea la baza de date. Dacă s-a realizat conectarea la baza de date atunci se deschide interfața principală unde trebuie ca personalul să introducă parola de acces la aplicație și se închide interfața de conectarea la baza de date. Dacă nu s-a realizat conexiunea la baza de date atunci apare un mesaj de eroare și trebuie introduse credențialele din nou.

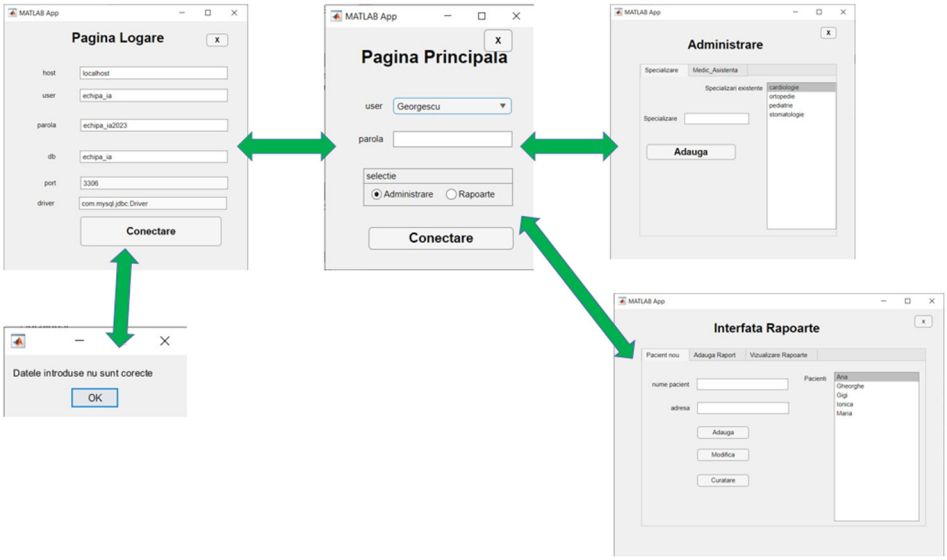



Fig. 4.10. Structura aplicației

În interfața principală funcție de selecția făcută (radio button) se va deschide interfața de administrare sau cea de rapoarte, dacă parola introdusă este corectă. La închiderea acestei interfețe se revine la interfața de conectare la baza de date.

În interfața de administrare pot fi adăugate noi specializări sau cadre medicale. Dacă se închide această interfață se va reveni la interfața principală.

Interfața rapoarte permite adăugarea de noi pacienți sau de rapoarte, precum și vizualizarea sau modificarea rapoartelor. La închiderea acestei interfețe se revine la interfața principală.

În continuare se vor prezenta fiecare interfață grafică utilizator, menționând obiectele utilizate la fiecare interfață precum și codurile utilizate.

Interfața de pornire este *Pagina Logare* (Fig. 4.11) unde trebuie introduse datele necesare conectări la baza de date. Această interfață conține șase câmpuri de tip *EditFieldText* și două butoane de tip *PushButton*. Cele șase câmpuri sunt editabile (utilizatorul poate introduce date) și sunt utilizate pentru introducerea următorilor parametri:

- host – adresa serverului MySQL;
- user – numele utilizatorului
- parola – parola corespunzătoare utilizatorului
- db – numele bazei de date
- port – portul de conectare la MySQL
- driver – numele driver-ului utilizat pentru realizarea legături dintre Matlab și MySQL.


Fig. 4.11. Interfața conectare la baza de date

Interfața de conectare la baza de date conține și două obiecte de tip *PushButton,* unul notat cu *X* pentru închiderea aplicației și unul numit *Conectare* care la apăsare stabilește conexiunea cu baza de date. Codul sursă al funcțiilor din spatele acestor butoane este:

```
% butonul de inchidere a interfetei
function XButtonPushed(app, event)
        closereq(); % inchidem interfata curenta
end

% butonul de conectare la baza de date
function ConectareButtonPushed(app, event)
        % preluam datele din campurile dedicate
        host=app.hostEditField.Value;
        user=app.userEditField.Value;
        pass=app.parolaEditField.Value;
        db=app.dbEditField.Value;
```

```
        driver=app.driverEditField.Value;
        port=app.portEditField.Value;
        % concatenam pentu a obtine url-ul
        url=strcat('jdbc:mysql://',host,':',port,'/',db);
        % realizam conectarea la baza de date.
        % variabila conectare va contine credentialele pentu conectare % la baza
        de date
        conectare=database(db,user,pass,driver,url);
        % verificam daca legatura la baza de date s-a realizat cu
        % succes
        if strcmp(conectare.Driver,app.driverEditField.Value)==1
                % transferam variabila cenectare in appdata pentru a fi
                % accesibila si pentru alte interfete
                setappdata(0,'conectare',conectare);
                principala; % pornim interfata denumita principala
                closereq(); % inchidem interfata curenta
        else
                % afisam mesaj daca datele pentru logare sunt gresite
                msgbox('Datele introduse nu sunt corecte');
        end
end
```

Prin rularea comenzi *database()* se va obține variabila *conectare* care conține credențialele pentru conectarea la baza de date. Această variabilă este de tip structură. Dacă s-a realizat cu succes conectarea la baza de date câmpul *conectare.Diver* al variabilei va conține numele driverului. Dacă din vari motive (utilizator, parolă, port, nume bază de date introduse greșit) nu s-a putut stabili conexiunea la baza de date atunci câmpul *conectare.Driver* va conține un mesaj de eroare.

Din acest motiv în codul sursă s-a utilizat instrucțiunea condițională *if* care verifică valoarea câmpului *conectare.Driver*. Dacă conectare s-a realizat cu succes atunci se transmite variabila *conectare* în *workspace-ul appdata* (acest lucru este necesar pentru a avea acces la această variabilă și în cadrul altor interfețe grafice utilizator), se deschide interfața *principala* și se închide interfața curentă.

Dacă nu s-a realizat legătura la baza de date atunci va apărea mesajul de eroare “ Datele introduse nu sunt corecte” (Fig. 4.12). Acest mesaj va apărea de câte ori datele introduse nu sunt corecte și nu continuă în aplicație până nu se reușește stabilirea conexiuni cu baza de date

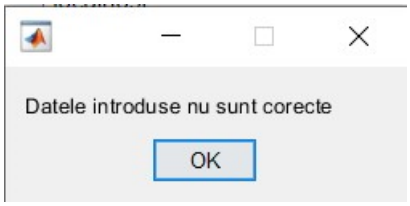



Fig. 4.12. Mesaj de eroare

Interfața principală (Fig. 4.13) este utilizată pentru conectarea utilizatorilor aplicației. Acești utilizatori reprezintă personalul medical (este diferit de utilizatorul folosit la conectarea la baza de date). După introducerea parolei și selectarea următoarei interfețe ( administrare sau rapoarte) prin apasarea butonului *Conectare* se deschide interfața selectată.

Dacă parola este introdus greșit, va apărea mesajul de eroare "Parolă greșită" și nu se continuă aplicația până când nu este introdusă corect parola pentru utilizatorul selectat.

Interfața principală conține următoarele obiecte:

- user – *DropDownMenu* cu utilizatorii disponibili (personalul medical);
- parola – *EditFieldText* – se va introduce parola;
- selectie – *RadioButton* – se selectează ce interfață grafică utilizator se va deschide;
- conectare – *PushButton* – verificăm datele și dacă sunt corecte continuam cu interfața aleasă
- x – *PushButton* – deschide interfața de conectare la baza de date și închide interfața curentă (*principala*)


Fig. 4.13. Interfața principală de conectare la aplicație

Pentru a avea acces la variabila *conectare* ce conține credențialele conectări la baza de date definim proprietatea *conectare*, astfel în orice funcție din această interfață să avem acces la acesta apelând *app.conectare*.

```
% definim variabilele ca proprietati privare
properties (Access = private)
        conectare % definim variabila de conectare la baza de date
end
```

S-a creat și funcția ce rulează la pornirea interfeței. În cadrul acestei funcții se citește variabila de conectare din *appdata* și se va afișa lista cu personalul medical în meniul de tip *DropDown*.

```
% functia ce ruleaza la pornirea interfetei
function startupFcn(app)
        % preluam din appdata variabila de conectare la db
        app.conectare=getappdata(0,'conectare');
        % comanda sql pentru selectia din baza de date a numelor
        % personalului medical
        query='select nume from medici_asistente;';
        rezultat=fetch(app.conectare,query); % se executa comanda sql
        % raspunsul bazei de date se afiseaza in meniul DropDown
        app.userDropDown.Items=rezultat.nume;
end
```

Butonul de închidere va deschide interfața de conectare la baza de date și va închide interfața curentă. Comanda utilizată pentru închiderea interfeței curente este *closereq()*.

```
% butonul de inchidere
function XButtonPushed(app, event)
        logare; % se acceseaza interfata logare
        closereq(); % se inchide interfata curenta
end
```

Funcția principală a acestei interfețe se regăsește în spatele butonului de conectare. În cadrul acestei funcții se preia numele utilizatorului selectat și parola introdusă de acesta și se va verifica dacă parola introdusă coincide cu cea din baza de date. Dacă parola nu corespunde va apărea un mesaj de eroare.

Dacă parola introdusă este corectă se verifică selecția făcută pentru obiectul de tip *RadioButton* .Funcție de această selecție se deschide interfața de administrare sau cea de rapoarte și se închide interfața curentă

```
% butonul de conectare
    % Button pushed function: ConectareButton
    function ConectareButtonPushed(app, event)
        % preluam numele selectat si parola introdusa de utilizator
        nume=app.userDropDown.Value;
        parola=app.parolaEditField.Value;
        % preluam din baza de date parola utilizatorului selectat
        query=strcat('select parola from parole where nume=''',nume,''';')
        rezultat=fetch(app.conectare,query);
        % verificam daca parola introdusa este corecta
        if strcmp(rezultat.parola{1},parola)==1
            % verificam selectia facuta in RadioButton
            if
strcmp(app.selectieButtonGroup.SelectedObject.Text,'Administrare')==1
                administrare; % rulam interfata administrare
                closereq(); % inchidem interfata curenta
            else
                rapoarte; % rulam interfata raloarte
                closereq(); % inchidem interfata curenta
            end
        else
            % apare mesaj daca parola introdusa e gresita
            msgbox('Parola gresita');
        end
    end
end
```

Interfața grafică utilizator pentru administrare conține 2 tab-uri, unul pentru a introduce noi specializări în cadrul clinici (Fig. 4.14), iar celălalt pentru introducerea de unor noi membri (personal medical) (Fig. 4.15). De asemenea mai are un buton de închidere.

La fel ca în cadrul interfeței principale, și în această interfață este nevoie de definirea variabilei de conectare la baza de date ca proprietate privată precum și inițializarea acesteia cu valoarea setată în *appdata*.

Pentru popularea inițială a obiectelor cu informați din baza de date este necesară crearea funcției ce rulează la pornirea interfeței.

Obiectele conținute de această interfață sunt:

- **tab – Specializare**;
  - specializari existente – *ListBox* – lista specializărilor disponibile;
  - specializare – *EditFieldText* – se introduce o nouă specializare;
  - adauga – *PushButton* – adaugă specializarea în baza de date;
- **tab – Medic_Asistenta**;
  - nume – *EditFieldText* – numele noului utilizator (personal medical);
  - adresa – *EditFieldText* – adresa noului utilizator;
  - specializare *DropDownMenu* – lista specializarilor disponibile;
  - parola – *EditFieldText* - parola noului utiliator
  - functie – *RadioButton* – se selectează dacă este medic sau asistentă;
  - adauga – *PushButton* – se adaugă noul utilizator în baza de date.

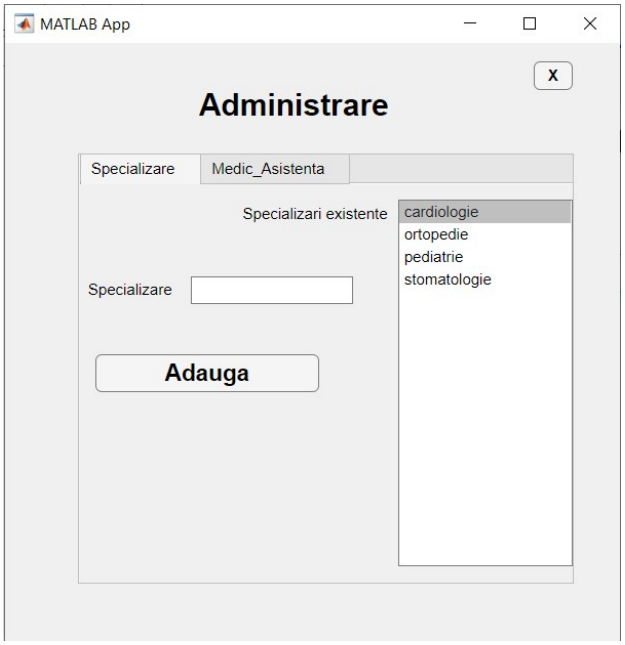



Fig. 4.14. Interfața administrare – tab specializare

```
% definirea proprietatilor
properties (Access = private)
      conectare % definim variabila de conectare la baza de date
   end
```

În cadrul funcției ce rulează la pornire, pe lângă inițializarea variabilei pentru conectarea la baza de date, s-a populat și *ListBox*-ul cu specializările disponibile în clinică.

```
% definirea functiei de pornire
function startupFcn(app)
        % preluam din appdata variabila de conectare la db
        app.conectare=getappdata(0,'conectare');
        % comanda sql pentru preluarea specializarilor din baza de date
        query='select specializare from specializare;';
        rezultat=fetch(app.conectare,query); % executia comenzi sql
        % afisarea specializarilor atat in ListBox cat si in DropDown
        app.SpecializariexistenteListBox.Items=rezultat.specializare;
        app.specializareDropDown.Items=rezultat.specializare;
    end
```

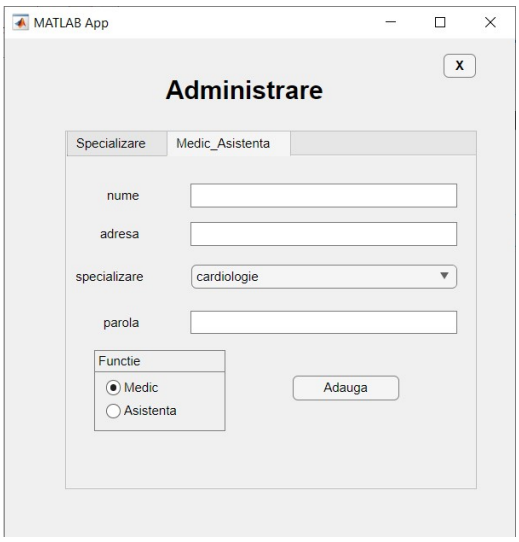



Fig. 4.15. Interfața administrare – tab medic_asistente

Butonul de închidere va deschide interfața principală şi va închide interfața curentă.

```
% butonul de inchidere
function XButtonPushed(app, event)
        principala; % se porneste interfata principala
        closereq(); % se inchide intefata curenta
    end
```

Butonul *adauga* din tab-ul *specializare* va citi informația scrisă în câmpul *specializare* şi va introduce acea valoare în tabelul corespunzător din baza de date, apoi va actualiza ListBox-ul care conține lista de specializări cu noua specializare introdusă

```
% butonul de adauga specializare
    % Button pushed function: AdaugaButton
    function AdaugaButtonPushed(app, event)
        % concatenam variabilele pentru a obtine comanda sql
        query=strcat('insert into specializare (specializare) values
('',app.SpecializareEditField.Value,'')');
        exec(app.conectare,query); % executam comanda sql
        % pentru ca am introdus o noua specializare in baza de date
        % se va actualiza lista cu specializari
        query='select specializare from specializare;';
        rezultat=fetch(app.conectare,query);
        app.SpecializariexistenteListBox.Items=rezultat.specializare;
        app.SpecializareEditField.Value='';
    end
```

Tab-ul *Medic_Asistentă* permite introducere unui nou membru în cadrul personalului medical. Datele necesare pentru noul membru sunt: numele, adresa și parola care vor fi introduse în câmpurile dedicate de tip *EditFieldText* și a specializări care se regăsește în *DropDownMenu.* Prin apăsarea butonului *adauga* aceste informați vor fi introduse în tabelele din baza de date, apoi informațiile din câmpurile de tip *EditFieldText* vor fi șterse.

```
% butonul de adauga personal medical
    % Button pushed function: AdaugaButton_2
    function AdaugaButton_2Pushed(app, event)
        % verificam selectia din RadioButton medic sau asistenta
        if strcmp(app.FunctieButtonGroup.SelectedObject.Text,'Medic')==1
            ma='1';
        else
            ma='0';
        end
        % generam comanda sql pentru indroducerea datelor in tabelul
        % medici _asistente
        query=strcat('insert into medici_asistente (nume,adresa,specializare,ma)
values ('',app.numeEditField.Value
,'','',app.adresaEditField.Value,'','',app.specializareDropDown.Value,'',',ma,');'
);
        exec(app.conectare,query); % executam comanda sql
        % generam comanda sql pentru introducerea parolei in tabelul
        % parole
```

```
        query=strcat('insert into parole (nume,parola) values
('',app.numeEditField.Value,'','',app.parolaEditField.Value,'');');
        exec(app.conectare,query); % executam comanda sql
        % curatam (stergem) valorile introduse in campurile  EditField
        app.numeEditField.Value='';
        app.adresaEditField.Value='';
        app.parolaEditField.Value='';
    end
  end
```

Interfața de rapoarte cuprinde trei tab-uri și un buton de închidere a interfeței. Cele trei tab-uri sunt:

- Pacient nou – pentru introducerea unui nou pacient;
- Adauga Raport – pentru adăugarea unui nou raport pentru un pacient existent;
- Vizualizare Rapoarte – Vizualizarea rapoartelor existente și eventual modificarea acestora dacă este cazul.

Interfața grafică utilizator pentru rapoarte conține următoarele obiecte:

- **tab – Pacient nou (Fig. 4.16)**
  - nume pacient – *EditFieldText* – numele noului pacient;
  - adresa – *EditFieldText* – se introduce adresa noului pacient;
  - pacienti – *ListBox* – lista pacienților înscriși la clinică;
  - adauga – *PushButton* – adaugă noul pacient în baza de date;
  - modifica – *PushButton* – se modifica adresa unui pacient existent în baza de date;
  - curatare – *PushButton* – se curăța câmpurile nume si adresă;

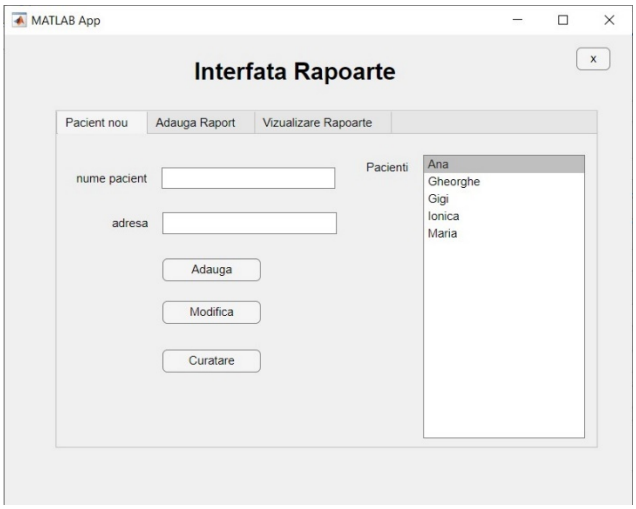



Fig. 4.16. Interfața rapoarte – tab pacient nou

- **tab – Adauga Raport (Fig. 4.17)**
  - pacient – *DropDownMenu* – lista pacienților clinici;
  - medic – *DropDownMenu* – lista medicilor;
  - asistenta – *DropDownMenu* –lista asistentelor;
  - data – *Calendar* – se selectează o dată;
  - diagnostic – *EditFieldText* – se introduce diagnosticul;
  - recomandari – *Text* – se introduce tratamentul și recomandările;
  - adauga – *PushButton* – se adaugă raportul în baza de date;

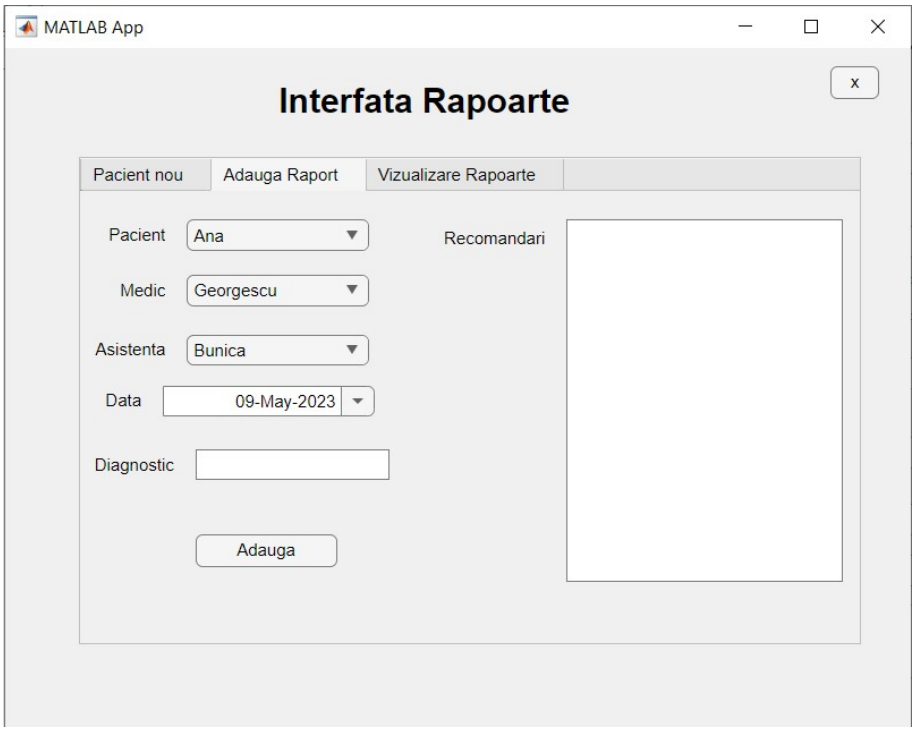



Fig. 4.17. Interfața rapoarte – tab adaugă raport

- **tab – Vizualizare Rapoarte (Fig. 4.18)**
  - pacient – *DropDownMenu* – lista pacienților clinici;
  - medic – *DropDownMenu* – lista medicilor;
  - data – *Calendar* – se selectează o dată;
  - label – *Label* – etichetă;
  - tabel – *Table* – tabel pentru afișare lista rapoarte;
  - recomandari – *Text* – se afișează recomandările;
  - modifica recomandari – *PushButton* – modifică recomandările din baza de date.

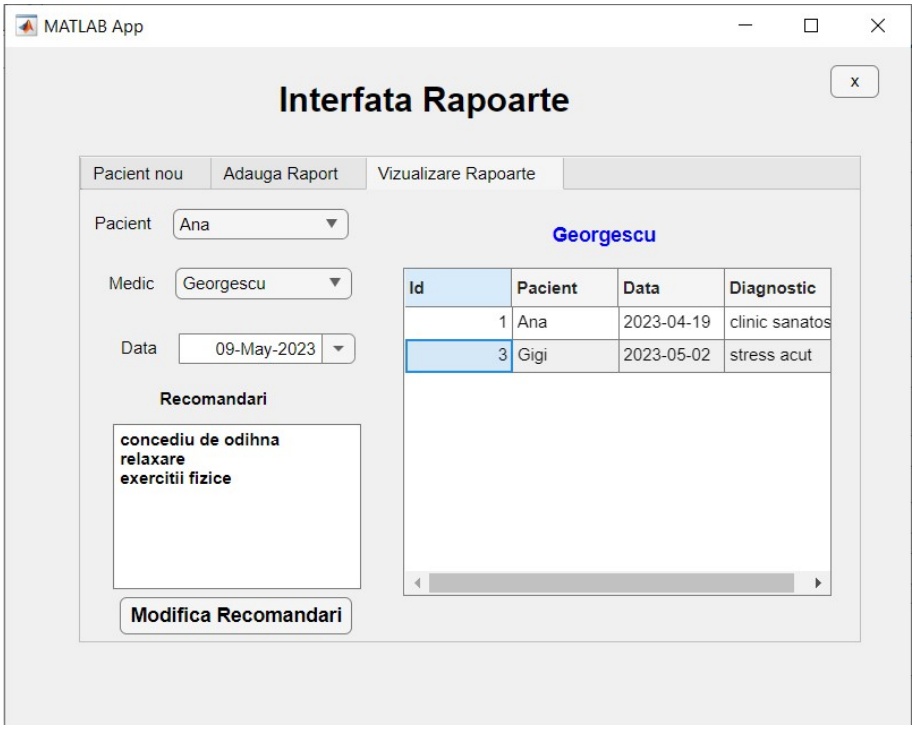



Fig. 4.18. Interfața rapoarte – tab vizualizare rapoarte

La fel ca în cadrul interfețelor *principala* și *administratie*, și în această interfață este nevoie de definirea variabilei de conectare la baza de date ca proprietate privată precum și inițializarea acesteia cu valoarea setată în *appdata*. Pentru popularea inițială a obiectelor cu informați din baza de date este necesară crearea funcției ce rulează la pornirea interfeței

```
% definirea proprietatilor
properties (Access = private)
    conectare % definim variabila de conectare la baza de date
    id        % definim o variabila pentru id raport
  end
```

În cadrul funcției ce rulează la pornire, pe lângă inițializarea variabilei pentru conectarea la baza de date, s-a populat și *ListBox*-ul cu pacienții înscriși la clinică.

```
% functia ce ruleaza la pornirea interfetei
function startupFcn(app)
    % preluam din appdata variabila de conectare la db
```

```
        app.conectare=getappdata(0,'conectare');
        % preluam din baza de date numele pacientilor
        query='select nume from pacienti';
        rezultat=fetch(app.conectare,query);
        % afisam lista cu pacienti in ListBox
        app.PacientiListBox.Items=rezultat.nume;
    end
```

La apăsarea butonului de închidere se revine la interfața *principala* și se închide interfața curentă adică interfața rapoarte.

```
% butonul de inchidere
    % Button pushed function: xButton
    function xButtonPushed(app, event)
        principala; % accesam interfata principala
        closereq(); % inchidem interfata curenta
    end
```

Funcția care adaugă un pacient nou, prea datele introduse în câmpurile de tip *EditFieldText* aferente numelui respectiv adresei noului pacient și introduce aceste informați în tabelul *pacienti* din baza de date. Apoi actualizează în aplicație lista pacienților.

```
% butonul de adaugare a unui nou pacient
    % Button pushed function: AdaugaButton
    function AdaugaButtonPushed(app, event)
        % comanda sql pentru introducerea unui nou pacient
        query=strcat('insert into pacienti (nume,adresa) values
('',app.numepacientEditField.Value,'','',app.adresaEditField.Value,'');');
        exec(app.conectare,query); % executam comanda sql
        % actualizam lista de pacienti din interfata
        query='select nume from pacienti';
        rezultat=fetch(app.conectare,query);
        % curatam (stergem) campurile EditField
        app.PacientiListBox.Items=rezultat.nume;
        app.numepacientEditField.Value='';
        app.adresaEditField.Value='';
    end
```

Pentru a modifica adresa unui pacient (dacă este cazul) atunci prin selectarea numelui acestuia din listă, se va căuta în baza de date adresa acestuia apoi acesta se va afișa în câmpul adresă din aplicație. Codul acestei funcții este prezentat în continuare.

```
% functia de selectare a unui pacient din lista
    % Value changed function: PacientiListBox
    function PacientiListBoxValueChanged(app, event)
        % numele pacientului selectat in ListBox este copiat si in
        % EditField
        app.numepacientEditField.Value = app.PacientiListBox.Value;
        % comanda sql pentru interogarea bazei de date pentru aflarea
        % adresei pacientului selectat
        query=strcat('select adresa from pacienti where
nume=''',app.PacientiListBox.Value,''';');
        rezultat=fetch(app.conectare,query); % executia comenzi sql
        % afisarea adresei in campul EditField
        app.adresaEditField.Value=rezultat.adresa{1};
    end
```

După ce a fost modificată adresa prin apăsarea butonului modificare, noua adresă va fi introdusă în baza de date în locul celei vechi.

```
% butonul de modificare a adresei unui pacient
    % Button pushed function: ModificaButton
    function ModificaButtonPushed(app, event)
        % comanda sql pentru modificarea adresei pacientului selectat
        query=strcat('update pacienti set adresa=''',app.adresaEditField.Value,'''
where nume=''',app.PacientiListBox.Value,''';');
        exec(app.conectare,query); % executia comenzi sql
        % curatarea (stergerea) datele introduse in campurile EditFeild
        app.numepacientEditField.Value='';
        app.adresaEditField.Value='';
    end
```

Butonul de curățare șterge informația din câmpurile nume și adresă pentru a facilita introducerea unui nou pacient.

```
% butonul de curatare a datelor din EditFieldText
    % Button pushed function: CuratareButton
```

```
    function CuratareButtonPushed(app, event)
        % curatarea (stergerea) datele introduse in campurile EditFeild
        app.numepacientEditField.Value='';
        app.adresaEditField.Value='';
    end
```

Fiecare din tab-urile interfeței conține informații existente în baza de date. De asemenea fiecare tab poate introduce sau modifica informația din baza de date, astfel la fiecare schimbare a tab-ului trebuie actualizată informația din acel tab. În acest sens s-a creat următoare funcție care la schimbarea tab-ului curent actualizează informația acestuia.

```
% functia care actualizeaza datele la schimbarea tab-ului
    % Selection change function: TabGroup
    function TabGroupSelectionChanged(app, event)
        % actualizarea datelor(informatiilor) din TabGroup
        selectedTab = app.TabGroup.SelectedTab;
        % daca tabul selectat este 'Adauga Raport'
        if strcmp(selectedTab.Title,'Adauga Raport')==1
            % actualizam meniul DropDown cu pacienti
            query='select nume from pacienti';
            rezultat=fetch(app.conectare,query);
            app.PacientDropDown.Items=rezultat.nume;
            % actualizam meniul DropDown cu medici
            query='select nume from medici_asistente where ma=1;';
            rezultat=fetch(app.conectare,query);
            app.MedicDropDown.Items=rezultat.nume;
            % actualizam meniul DropDown cu asistente
            query='select nume from medici_asistente where ma=0;';
            rezultat=fetch(app.conectare,query);
            app.AsistentaDropDown.Items =rezultat.nume;
            % setam pentru calendar data de azi
            app.DataDatePicker.Value=datetime('today');
        end
        % daca tabul selectat este 'Vizualizare Rapoarte'
        if strcmp(selectedTab.Title,'Vizualizare Rapoarte')==1
            % actualizam meniul DropDown cu pacienti
            query='select nume from pacienti';
            rezultat=fetch(app.conectare,query);
            app.PacientRDropDown.Items=rezultat.nume;
```

```
            % actualizam meniul DropDown cu medici
            query='select nume from medici_asistente where ma=1;';
            rezultat=fetch(app.conectare,query);
            app.MedicRDropDown.Items=rezultat.nume;
            % setam pentru calendar data de azi
            app.DataRDatePicker.Value=datetime('today');
        end
    end
```

Pentru a adăuga un nou raport trebuie selectat pacientul din lista pacienților, medicul din lista personalului medical, data investigației (calendar) și completa câmpurile cu diagnostic respectiv recomandări. Apoi prin apăsarea butonului *adauga* aceste informații vor fi introduse în baza de date.

```
% buton adauga raport
    % Button pushed function: AdaugaButton_2
    function AdaugaButton_2Pushed(app, event)
        % comanda sql pentru introducerea unui nou raport
        query=strcat('insert into raport (pacient, medic, data, diagnostic,
recomandari) values
('',app.PacientDropDown.Value,'','',app.MedicDropDown.Value,'','',datestr(ap
p.DataDatePicker.Value,'yyyy-mm-
dd'),'','',app.DiagnosticEditField.Value,'','',strjoin(app.RecomandariTextArea.
Value,'\r\n'),'');')
        exec(app.conectare,query); % executarea comenzi sql
    end
```

Pentru ca formatul datei să corespundă cu cel din MySQL (yyyy-mm-dd) s-a utilizat comanda *datestr()* care convertește data introdusă în formatul acceptat de MySQL.

De asemenea pentru câmpul recomandări a trebuit efectuată o conversie. La citirea de către Matlab a informației din câmpului recomandări, se va crea o variabilă de tip *cell array* în care fiecare celulă va conține informația de pe un rând al câmpului recomandări. MySQL acceptă informația doar ca șir de caractere, nu sub formă de *cell array*. Așadar utilizând comanda *strjoin()* se va crea un șir de caractere prin concatenarea celulelor din tabel și folosind pentru delimitarea acestora șirul *\r\n* care este înțeles de MySQL ca trecere la linie nouă (**r**eturn**n**ew line*)*.

În tab-ul de vizualizare a rapoartelor se dorește filtrarea listei cu rapoarte după pacient, medic sau dată. Astfel actualizarea listei din tabel se face la selectarea unuia dintre acești parametri.

La selectarea unui pacient în tabel va apărea lista rapoartelor pentru acel pacient, afișând id-ul raportului, medicul care a completat raportul, data completări raportului precum și diagnosticul. De asemenea deasupra tabelului va apărea printr-o variabilă de tip *label* numele pacientului selectat.

```
% functie vizualizare rapoarte pentru un pacient selectat
    % Value changed function: PacientRDropDown
    function PacientRDropDownValueChanged(app, event)
        % comanda sql de afisarea a rapoartelor pentru pacientul selectat
        query=strcat('select id, medic,data, diagnostic from raport where
pacient="',app.PacientRDropDown.Value,'";');
        rezultat=fetch(app.conectare,query); % executia comenzi sql
        % afisarea rezultatelor in tabel
        app.UITable.Data=rezultat;
        % notarea numelor coloanelor (antet tabel)
        app.UITable.ColumnName={'Id','Medic','Data','Diagnostic'};
        % etichetarea pacientului ales
        app.SelectatLabel.Text=app.PacientRDropDown.Value;
        % curatarea recomandarilor din campul text
        app.Rap.Value='';
    end
```

La selectarea unui medic în tabel va apărea lista rapoartelor pentru acel medic, afișând id-ul raportului, numele pacientului investigat, data completări raportului precum și diagnosticul. De asemenea deasupra tabelului va apărea printr-o variabilă de tip *label* numele medicului selectat.

```
% functie vizualiare rapoarte pentru un medic selectat
    % Value changed function: MedicRDropDown
    function MedicRDropDownValueChanged(app, event)
        % comanda sql de afisarea a rapoartelor pentru medicul selectat
        query=strcat('select id, pacient,data, diagnostic from raport where
medic="',app.MedicRDropDown.Value,'";');
        rezultat=fetch(app.conectare,query); % executia comenzi sql
        % afisarea rezultatelor in tabel
        app.UITable.Data=rezultat;
        % notarea numelor coloanelor (antet tabel)
```

```
            app.UITable.ColumnName={'Id','Pacient','Data','Diagnostic'};
            % etichetarea medicului ales
            app.SelectatLabel.Text=app.MedicRDropDown.Value;
            % curatarea recomandarilor din campul text
            app.Rap.Value='';
        end
```

La selectarea unui date din calendar în tabel va apărea lista rapoartelor din data respectivă, afișând id-ul raportului, numele pacientului investigat, medicul care a completat raportul, precum și diagnosticul. De asemenea deasupra tabelului va apărea printr-o variabilă de tip *label* data selectată.

```
%functie vizualizare rapoarte dintr-o data selectata
        % Value changed function: DataRDatePicker
        function DataRDatePickerValueChanged(app, event)
            % alegerea datei de azi si formatarea acesteia
            data = datestr(app.DataRDatePicker.Value,'yyyy-mm-dd');
            % comanda sql de afisarea a rapoartelor pentru data selectata
            query=strcat('select id, pacient, medic,diagnostic from raport where
data='',data,''';');
            rezultat=fetch(app.conectare,query); % executia comenzi sql
            % afisarea rezultatelor in tabel
            app.UITable.Data=rezultat;
            % notarea numelor coloanelor (antet tabel)
            app.UITable.ColumnName={'Id','Pacient','Medic','Diagnostic'};
            % etichetarea datei ales
            app.SelectatLabel.Text=data;
            % curatarea recomandarilor din campul text
            app.Rap.Value='';
        end
```

Deoarece recomandările conțin mai multe rânduri (linii) nu s-a dorit afișarea acestora în tabel. Dar pentru a le vizualiza s-a creat un obiect de tip *text* în care se vor afișa recomandările pentru raportul selectat în tabel. Codul sursă pentru afișarea recomandărilor este prezentat în continuare.

```
% afisarea recomandarilor pentru raportul selectat
        % Cell selection callback: UITable
        function UITableCellSelection(app, event)
            % aflarea liniei si coloanei celului alese din tabel
```

```
        indices = event.Indices
        % aflarea valorii din prima coloana a liniei alese, acesta
        % reprezentand id-ul din baza de date al raportului selectat
        app.id=num2str(app.UITable.Data(indices(1),1).id)
        % comanda sql pentru a selecta din baza de date recomandarile
        % pentru id-ul selectat
        query=strcat('select recomandari from raport where id=',app.id,';');
        rezultat=fetch(app.conectare,query); % executarea comenzi sql
        % afisarea recomandarilor in campul text
        app.Rap.Value=rezultat.recomandari;
    end
```

Deoarece coloanele tabelului diferă funcție de filtrele aplicate (pacient, medic, dată) cel mai simplu mod de identificare în mod unic a raportului selectat este utilizarea coloanei *id* care reprezintă id-ul raportului. Această coloană este întotdeauna prima. Astfel de la indicele celulei selectate din tabel se va utiliza doar prima valoare (ce reprezintă linia sau rândul), numărul coloanei fiind unu. La această poziție se află id-ul raportului selectat. Prin intermediul acestui id putem interoga baza de date în vederea obținerii recomandărilor aferente acestui raport, ce vor fi afișate în aplicație.

După afișarea recomandărilor aferente raportului selectat dacă se dorește modificarea acestora este posibil. Pentru a efectua aceste modificări și în baza de date trebuie apăsat butonul de *Modifica Recomandari.*

```
% buton modificare recomandari
    % Button pushed function: ModificaRecomandariButton
    function ModificaRecomandariButtonPushed(app, event)
      % comanda sql de modificare a recomandarilor din baza de date
      query=strcat('update raport set
recomandari="',strjoin(app.Rap.Value,'\r\n'),'" where id=',app.id,';')
      exec(app.conectare,query); % executia comenzi sql
      % curatarea campului text
      app.Rap.Value='';
    end
  end
```

S-a putut observa că gestionarea unei baze de date MySQL este destul de facilă. Comenzile SQL pentru înserarea, modificarea, ștergerea și regăsirea informației într-o bază de date nu sunt complicate. Trebuie avut în vedere respectarea structuri comenzi precum și o atenție sporită la denumirea tabelelor și a atributelor deoarece este un program *case sensitive* .

Modulul *appdesigner* din Matlab permite crearea cu ușurință a interfețelor grafice utilizator, iar funcțiile puse la dispoziție de modulul *database toolbox* sunt foarte ușor de utilizat.

Această aplicație poate fi dezvoltată în continuare, implementând noi facilități. De exemplu se poate implementa și o conexiune cu baza de date a CNAS (Casa Națională de Asigurări de Sănătate) pentru lista codurilor de diagnostic sau a medicamentelor disponibile. De asemenea pentru creșterea securității cibernetice se poate cripta informația din baza de date.

# 5 BIBLIOGRAFIE


[1] https://koaha.org/wiki/Macchina_analitica
[2] https://ro.wikipedia.org/wiki/Ma%C8%99in%C4%83_diferen%C8%9Bial%C4%83
[3] https://www.britannica.com/technology/Atanasoff-Berry-Computer
[4] https://en.wikipedia.org/wiki/Harvard_Mark_I
[5] https://ro.wikipedia.org/wiki/Tub_electronic
[6] https://ro.wikipedia.org/wiki/Tranzistor#/media/Fi%C8%99ier:Transistor-photo.JPG
[7] https://www.oboyle.ro/sortiment-circuite-integrate-aprox-20-bucati.html
[8] https://ro.wikipedia.org/wiki/Microprocesor
[9] https://ro.wikipedia.org/wiki/Sistem_de_operare
[10] https://ro.wikipedia.org/wiki/Windows_1.0
[11] https://www.casedemarcatsatumare.ro/unitate-centrala-produs54.html
[12] https://www.pchouse.ro/dell-optiplex-3050-core-i3-6100t-320ghz-8gb-ddr4-sodimm-240gb-ssd-minipc-windows-10-home-mar-calculator-refurbished.html?gclid=CjwKCAjwitShBhA6EiwAq3RqA7vzeNUsVJpKWocC68eqYtMhvldOasxCWgZ0by-Pq9Dc1-6tyiE4RRoC4XQQAvD_BwE
[13] https://totuldesprecalculatoare.weebly.com/placa-de-baza.html
[14] https://www.itmar.ro/cumpara/placa-de-baza-asus-h110m-a-dp-socket-1151-2686?gclid=CjwKCAjwitShBhA6EiwAq3RqA__RUwrLcqtmYsNhYoxA8DxJkS1pYgDMVhqYKZZGjRVi8HP-s1m4GRoCFx8QAvD_BwE
[15] https://www.pcgarage.ro/memorii/kingston/fury-beast-16gb-ddr4-3200mhz-cl16-1/
[16] https://en.wikipedia.org/wiki/Hard_disk_drive#/media/File:Laptop-hard-drive-exposed.jpg
[17] https://i.ytimg.com/vi/EXLfErPEYiw/maxresdefault.jpg
[18] https://www.creeaza.com/referate/informatica/retele-calculatoare/Topologia-de-tip-stea-STAR616.php
[19] https://www.creeaza.com/referate/informatica/retele-calculatoare/Topologia-de-tip-inel-RING897.php
[20] https://www.scritub.com/stiinta/informatica/Topologii-de-retea112202122.php

[21] https://www.creeaza.com/referate/informatica/retele-calculatoare/Topologia-fizica-de-tip-magist787.php
[22] Andrew S. Tanenbaum, Rețele de calculatoare, Ediția a treia, Editura Computer Press AGORA, 1998, ISBN: 973-97706-3-0
[23] https://ro.wikipedia.org/wiki/Modelul_OSI#:~:text=Modelul%20OSI%20divizeaz%C4%83%20problema%20complex%C4%83,comunic%C4%83%20doar%20cu%20straturile%20adiacente.
[24] https://ro.wikipedia.org/wiki/Domain_Name_System#:~:text=Domain%20Name%20System%20(abreviat%20DNS,de%20Internet%20Protocol%20(IP).
[25] https://ro.myservername.com/snmp-what-is-simple-network-management-protocol
[26] https://ro.wikipedia.org/wiki/SMTP#:~:text=Simple%20Mail%20Transfer%20Protocol%20(prescurtat,electronic%20%C3%AEn%20re%C8%9Bea%20de%20calculatoare.
[27] https://ro.wikipedia.org/wiki/DHCP#:~:text=DHCP%20(prescurtat%20de%20la%20Dynamic,re%C8%9Bea%20importante%20%C3%AEn%20mod%20dinamic.
[28] https://ro.wikipedia.org/wiki/Transmission_Control_Protocol#:~:text=TCP%20este%20un%20serviciu%20de,duplicarea%20sau%20pierderea%20de%20date.
[29] https://ro.wikipedia.org/wiki/User_Datagram_Protocol#:~:text=User%20Datagram%20Protocol%20(sau%20UDP,livrarea%20mesajelor%20%C3%AEntr%2Do%20re%C8%9Bea.
[30] https://www.dabstore.ro/placa-de-retea-gigabit-lan-pcie-x1-tp-link-tg-3468-1-x-rj45-1g-p496769/
[31] https://www.vexio.ro/router-wireless/d-link/1103744-dsr-1000ac-2xwan-gigabit-3xlan-gigabit/?ref=compari&utm_source=compari.ro&utm_medium=shopping-campaign&utm_campaign=compari_shopping-click&gclid=EAIaIQobChMIisvunfTM_gIVPwcGAB3ThQHnEAQYAyABEgJXT_D_BwE
[32] https://www.a2t.ro/switch-uri-poe-retea-telefonie/switch-tp-link-24-porturi-10-100-tl-sf1024d.html?gclid=EAIaIQobChMIjYaf0vTM_gIVwwaLCh0bqgHjEAQYASABEgIyx_D_BwE

[33] https://www.a2t.ro/retele-si-telefonie/media-convertor-mm-2-fibre-conectori-sc-1000mb-s-max-2km.html?gclid=EAIaIQobChMI0unnxfPM_gIVA-93Ch24Xwx0EAQYAyABEgJJGfD_BwE
[34] https://www.senetic.ro/product/USGFLEX200-EU0102F?gclid=EAIaIQobChMI5rjmjvXM_gIVGuN3Ch3fPg3wEAQYASABEgIhlvD_BwE
[35] https://www.arcelect.com/DialUp_Modem_with_Async_Router.htm
[36] https://www.gsmnet.ro/gadgeturi-routere/modem-huawei-e3372-325-lte-4g-alb-51071uxg?gclid=EAIaIQobChMI9sDe9PbM_gIV1NHVCh3IUAOZEAQYASABEgKif_D_BwE
[37] https://www.biasicom.ro/access-point-ubiquiti-u6-lite-indoor-dual-band-wifi-6.html?gclid=EAIaIQobChMI_tDUj_fM_gIVxZrVCh2znQCoEAQYBCABEgL8pvD_BwE
[38] https://www.dell.com/en-us/shop/servers-storage-and-networking/poweredge-t350-tower-server-windows-server-2022/spd/poweredge-t350/pe_t350_15326_os_vi_vp
[39] https://www.easytechjunkie.com/what-is-a-server-room.htm
[40] https://www.emag.ro/amplificator-semnal-retea-wireless-vaxiuja-wifi-repeater-transmisie-300mbps-retea-2-4-ghz-conexiune-wpa-si-wpa2-2-antene-externe-cu-port-lan-negru-wifizjq-heise/pd/D76HSKMBM/
[41] https://www.a2t.ro/accesorii-supraveghere/cablu-u-utp-cat-6-4x2xawg23-1-300mhz-pvc-eca-albastru.html?gclid=EAIaIQobChMI0Z2_hdTO_gIVD8d3Ch0pQgmLEAQYAyABEgJeSvD_BwE
[42] https://forum.huawei.com/enterprise/en/is-rj45-the-same-as-ethernet/thread/815501-861
[43] https://www.truecable.com/blogs/cable-academy/t568a-vs-t568b
[44] https://www.guru99.com/difference-between-straight-through-crossover-cables.html
[45] https://www.ledled.ro/cabluri-de-energie/cablu-coaxial-tv-rg6-u-aluminiu-rola-100m-led3942.html#group=group&photo=id-1
[46] https://fiberstore.ro/843--cablu-fibra-optica-gyfxy-22-kn-7mm-4-fire.html
[47] https://www.baluna.ro/ro/ce-inseamna-raid/

[48] https://ro.wikipedia.org/wiki/Baz%C4%83_de_date
[49] Marian BUCOS, Sisteme de baze de date relaționale, Editura Politehnica Timișoara, 2014, ISBN 978-606-554-852-7
[50] Ioan FILIP, Sisteme de gestiune a bazelor de date, Editura Orizonturi Universitare Timișoara, 2007, ISBN 978-973-638-310-6
[51] Dan Alexandru PESCARU, Baze de date relaționale și orientate pe obiecte, Editura Politehnica Timișoara, 2001, ISBN 973-8247-53-5
[52] https://ro.wikipedia.org/wiki/SQL#:~:text=SQL%20(de%20la%20numele%20englez,limbaj%20bazat%20pe%20algebra%20rela%C8%9Bional%C4%83.
[53] https://www.w3schools.com/mysql/default.asp
[54] https://www.php.net/manual/en/
[55] https://www.python.org/
[56] https://www.javascript.com/
[57] https://www.ruby-lang.org/en/
[58] https://uk.mathworks.com/products/matlab.html


ISBN 978-606-35-0535-5