UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU

FACULTATEA DE INGINERIE

DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

**PROIECT DE DIPLOMĂ**

Conducător științific : Crețulescu Radu

Îndrumător: Crețulescu Radu

Absolvent: Popa Doroteea Cristina

Specializarea: Tehnologia Informației

* Sibiu, 2024 –

UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU

FACULTATEA DE INGINERIE

DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

APLICAȚIE GESTIONARE DE PROIECTE DEP.CIE

Conducător științific: Crețulescu Radu

Îndrumător: Crețulescu Radu

Absolvent: Popa Doroteea Cristina

Specializarea: Tehnologia Informației

Plan tematic

CUPRINS

[1. Prezentarea temei 5](#_Toc169553358)

[1.1. Introducere 5](#_Toc169553359)

[1.2. Descrierea temei 5](#_Toc169553360)

[1.3. Scop si obiective 5](#_Toc169553361)

[1.4. Tehnologii utilizate 6](#_Toc169553362)

[1.5. Cazuri principale de utilizare a aplicației 6](#_Toc169553363)

[1.6. Organizarea lucrării 8](#_Toc169553364)

[2. Considerații teoretice 8](#_Toc169553365)

[2.1. PHP și PhpMyAdmin 8](#_Toc169553366)

[2.2. SQL și MySQL 12](#_Toc169553367)

[2.3. HTML 13](#_Toc169553368)

[2.4. CSS 16](#_Toc169553369)

[2.5. JavaScript 20](#_Toc169553370)

[2.6. XAMPP 23](#_Toc169553371)

[2.7. Git 26](#_Toc169553372)

[2.8. Visual Studio Code 29](#_Toc169553373)

[3. Arhitectura aplicației 32](#_Toc169553374)

[3.1. Introducere baza de date 32](#_Toc169553375)

[3.2. Descrierea bazei de date 32](#_Toc169553376)

[4. Implementarea practică 32](#_Toc169553377)

[4.1. Actori și use-cases 32](#_Toc169553378)

[4.2. Vizitator use-cases 34](#_Toc169553379)

[4.3. Student use-cases 34](#_Toc169553380)

[4.4. Profesor use-cases 35](#_Toc169553381)

[4.5. Secretara use-cases 35](#_Toc169553382)

[4.6. Coordonator de licență use-cases 36](#_Toc169553383)

[4.7. Admin use-cases 36](#_Toc169553384)

# Prezentarea temei

## Introducere

În era digitală modernă, gestionarea eficientă a proiectelor academice este esențială pentru succesul educațional. Instituțiile de învățământ superior se confruntă adesea cu provocări legate de organizarea și monitorizarea progresului studenților în cadrul proiectelor lor. Pentru a aborda aceste provocări, am dezvoltat aplicația "UniProject Manager", o platformă web destinată să sprijine studenții, profesorii și personalul administrativ în gestionarea proiectelor academice și a lucrărilor de licență. Această aplicație oferă un mediu centralizat și intuitiv, care facilitează comunicarea și colaborarea între toți participanții la procesul educațional.

## Descrierea temei

Tema proiectului "UniProject Manager" se concentrează pe dezvoltarea unei aplicații web complexe, care să răspundă nevoilor specifice ale facultății de inginerie ULBS Sibiu. Aplicația este destinată să ajute cinci categorii principale de utilizatori: studenți, profesori, coordonatori, secretara și administratorul site-ului. Fiecare tip de utilizator beneficiază de funcționalități specifice care îmbunătățesc organizarea și gestionarea proiectelor.

**Studenții** pot accesa și gestiona informațiile legate de proiectele lor academice într-un mod organizat și eficient. Ei pot vizualiza datele personale, proiectele curente și notele aferente. În plus, aplicația permite studenților să actualizeze progresul proiectelor lor prin bifarea cerințelor îndeplinite și încărcarea de fișiere relevante. Pentru studenții din ani terminali, aplicația oferă opțiunea de a selecta și gestiona teme de licență, facilitând comunicarea cu coordonatorii de proiect.

**Profesorii** au posibilitatea de a gestiona proiectele studenților, de a adăuga cerințe pentru proiecte și de a evalua munca studenților prin vizualizarea și notarea arhivelor încărcate. Aplicația permite coordonatorilor să gestioneze cererile de licență, să accepte sau să respingă cererile studenților și să vizualizeze progresul proiectelor de licență.

**Secretara** joacă un rol crucial în administrarea utilizatorilor și a datelor aferente. Ea poate adăuga sau șterge profesori și studenți, poate introduce coordonatori și poate seta datele de prezentare pentru temele de licență. De asemenea, secretara primește listele cu candidații pentru fiecare specializare și asigură o organizare eficientă a acestora.

**Administratorul** are drepturi limitate, dar esențiale pentru funcționarea aplicației. Acesta poate schimba secretara și poate actualiza anul academic, asigurând astfel o administrare eficientă a platformei.

## Scop si obiective

Scopul principal al aplicației "UniProject Manager" este de a crea o platformă integrată și eficientă pentru gestionarea proiectelor academice și a lucrărilor de licență. Prin această aplicație, dorim să îmbunătățim organizarea și colaborarea între studenți, profesori și coordonatori, oferindu-le un instrument puternic și intuitiv.

Obiective specifice includ:

* **Facilitarea organizării proiectelor**: Aplicația oferă studenților un mod simplu și eficient de a-și gestiona proiectele academice, vizualizând cerințele și progresul în timp real.
* **Îmbunătățirea comunicării**: Prin intermediul platformei, studenții pot comunica mai ușor cu profesorii și coordonatorii lor, având acces la informații relevante și actualizate.
* **Simplificarea procesului de încărcare și evaluare**: Aplicația permite studenților să încarce fișiere relevante pentru proiectele lor, iar profesorii pot evalua aceste fișiere într-un mod structurat și eficient.
* **Centralizarea gestionării licențelor**: Studenții din ani terminali pot selecta și gestiona temele de licență propuse de coordonatori, facilitând astfel procesul de alegere și monitorizare a progresului lucrărilor de licență.

## Tehnologii utilizate

Dezvoltarea aplicației "UniProject Manager" a implicat utilizarea unui set diversificat de tehnologii, care au fost alese pentru a asigura performanța, securitatea și scalabilitatea platformei.

* **Visual Studio Code**: Editorul de cod utilizat pentru dezvoltarea aplicației, oferind o interfață prietenoasă și multiple extensii care facilitează scrierea și depanarea codului.
* **PHP**: Limbajul de programare server-side folosit pentru logica aplicației și manipularea datelor pe server.
* **HTML și CSS**: Tehnologiile utilizate pentru structura și stilizarea interfeței utilizatorului, asigurând o experiență de utilizare ideală.
* **JavaScript**: Limbajul de programare utilizat pentru adăugarea de funcționalități dinamice și interactive în cadrul aplicației.
* **XAMPP**: Pachetul software care include Apache (server web), MySQL (bază de date) și PHP, utilizat pentru dezvoltarea și testarea locală a aplicației.
* **MySQL**: Sistemul de gestionare a bazelor de date utilizat pentru stocarea și gestionarea datelor aplicației.
* **phpMyAdmin**: Instrumentul folosit pentru administrarea bazelor de date MySQL, oferind o interfață grafică ușor de utilizat pentru gestionarea datelor.

## Cazuri principale de utilizare a aplicației

Aplicația "UniProject Manager" este concepută pentru a acoperi o gamă largă de scenarii de utilizare, fiecare adresând nevoile specifice ale diferitelor categorii de utilizatori.

**Studenți**:

* **Organizarea proiectelor**: Studenții pot accesa secțiunea "Acasă" pentru a vizualiza datele personale și proiectele asociate. În secțiunea "Proiecte", aceștia pot vedea toate proiectele lor afișate sub formă de carduri colorate în funcție de an (albastru pentru anul 1, roz pentru anul 2, verde pentru anul 3, galben pentru anul 4 și portocaliu pentru licență). Fiecare card conține detalii despre proiect, iar la click, utilizatorul este redirecționat către detaliile proiectului, unde poate bifa cerințele îndeplinite, încărca fișiere și actualiza progresul.
* **Gestionarea licențelor**: Pentru studenții din anul 4, există un card special pentru licență. Acesta afișează datele de predare și prezentare a lucrării de licență. La click, sunt afișate temele de licență propuse de coordonatori, iar studenții pot trimite cereri coordonatorului pentru tema dorită. Statutul cererii (în așteptare, acceptată) este actualizat în timp real, iar studenții pot încărca arhivele necesare după ce cererea este acceptată de coordonator.

**Profesori**:

* **Gestionarea proiectelor**: Profesorii pot accesa secțiunea "Acasă" pentru a vedea un rezumat al activităților lor. În secțiunea "Proiecte", profesorii pot vedea carduri pentru fiecare semigrupă la care predă, cu detalii despre proiectele studenților. La click pe un card, profesorii pot adăuga cerințe, vizualiza arhivele încărcate și pot nota proiectele studenților.
* **Gestionarea licențelor**: În secțiunea dedicată licențelor, profesorii pot propune teme de licență, seta numărul de locuri disponibile și data predării. De asemenea, ei pot vizualiza cererile studenților și pot accepta sau respinge aceste cereri. Statutul cererilor și arhivele încărcate de studenți sunt gestionate eficient prin intermediul platformei.

**Secretara**:

* **Administrarea utilizatorilor**: Secretara poate adăuga și șterge profesori și studenți, poate introduce coordonatori. Aceste funcționalități sunt accesibile prin secțiunea "Acasă".
* **Gestionarea licențelor**: Secretara primește listele cu candidații pentru fiecare specializare și asigură organizarea eficientă a acestora, setând data de prezentare a licențelor de către fiecare semigrupă.

**Administrator**:

* **Gestionarea aplicației**: Administratorul are drepturi limitate, dar esențiale pentru funcționarea aplicației. Acesta poate schimba secretara și poate actualiza anul academic, asigurând astfel o administrare eficientă a platformei.

### Concluzie

Aplicația "UniProject Manager" reprezintă un instrument puternic și versatil pentru gestionarea proiectelor academice și a lucrărilor de licență. Prin integrarea funcționalităților specifice pentru studenți, profesori, coordonatori și administratori, platforma îmbunătățește considerabil organizarea și comunicarea în cadrul instituției de învățământ. Utilizarea tehnologiilor moderne asigură performanța și scalabilitatea necesare pentru a răspunde nevoilor complexe ale utilizatorilor, facilitând un proces educațional eficient.

## Organizarea lucrării

Această lucrare este organizată în patru capitole.

# Considerații teoretice

# 

## 2.1. PHP și PhpMyAdmin

### 2.1.1. Introducere

PHP (Hypertext Preprocessor) este un limbaj de scripting server-side, aceasta însemnând ca rulează la nivel de server, nu la nivel de client (browser). Acest limbaj e folosit în principal în dezvoltarea web. Acest limbaj de programare a fost creat de Rasmus Lerdorf în anul 1994 si este unul dintre cele mai utilizate limbaje in dezvoltarea web. PHP este preferat de către programatori deoarece oferă flexibilitate, este relativ simplu de utilizat și oferă posibilitatea de a lucra cu diferite baze de date, astfel asigurând dezvoltarea unor pagini web care oferă siguranță cibernetică si dinamicitate.

Unul dintre motivele pentru care PHP este un limbaj popular este faptul ca este open-source, acest lucru facilitând accesarea și modificarea codului sursă. Acest lucru a determinat formarea unei comunități de utilizatori care au îmbunătățit constant capacitățile PHP. De asemenea, PHP poate rula pe mai multe sisteme de operare, precum Linux, macOS si Windows si e compatibil cu servere web importante, cum ar fi Apache și Nginx.

Un alt avantaj al PHP-ului este sintaxa simplă, ușor de învățat, fiind accesibil pentru persoanele deja familiarizate cu limbaje de programare precum C, Java, Perl sau începătorilor care vor să învețe dezvoltare web. PHP oferă un proces simplificat de dezvoltare prin manipularea formularelor, operațiuni cu fișiere și gestionare de sesiuni.

### 2.1.2. Caracteristici cheie ale limbajului PHP

PHP are diverse caracteristici care îl fac să fie un limbaj atractiv pentru programatori. Câteva caracteristici importante a PHP-ului ar fi:

* Simplicitate și ușurință în învățare:

Sintaxa PHP e asemănătoare cu a altor limbaje de programare cunoscute (C, Perl), fiind astfel ușor de învățat pentru cei cu cunoștințe anterioare despre aceste limbaje. Sintaxa e simpla, ușor de învățat până și pentru persoanele care abia încep să învețe programare.

* Performanță și viteză:

PHP este un limbaj interpretat la nivel de server, aceasta însemnând că serverul execută codul PHP înainte de trimiterea paginii către client. Astfel, conținutul e generat dinamic, în timp real, fără un efect negativ semnificativ asupra performanței sau vitezei.

* Suport extins pentru baze de date:

PHP poate fi folosit ușor cu majoritatea sistemelor de gestionare a bazelor de date. Printre ele se regăsesc MySQL, PostgreSQL, Oracle si multe altele. Aceasta oferă dezvoltatorilor flexibilitate în alegerea bazei de date care e cea mai potrivită cu nevoile lor.

* Gestionarea sesiunilor:

Este o caracteristică esențială în dezvoltarea aplicațiilor web care folosesc autentificarea și păstrarea stării utilizatorilor. PHP oferă un mecanism simplu pentru a gestiona sesiunile, permițând urmărirea și menținerea informațiilor despre utilizatori pe parcursul mai multor pagini.

* Manipularea erorilor:

PHP conține funcții integrate pentru gestionare de excepții și erori. Dezvoltatorii pot folosi blocuri try-catch pentru gestionarea eficientă a excepțiilor și pot seta niveluri diferite pentru raportarea erorilor.

* Suport pentru programe orientate pe obiecte:

Dezvoltatorii care folosesc PHP își pot organiza și structura codul într-un mod modular și reutilizabil cu ajutorul programării orientate pe obiecte, suportată de limbaj. Astfel aceștia pot crea clase, obiecte, moștenire, polimorfism și se pot folosi de alte concepte OOP pentru a îmbunătății claritatea codului, performanța și viteza.

Alte caracteristici importante a PHP-ului ar fi portabilitatea și compatibilitatea cu diverse sisteme de operare (Linux, Windows, macOS), actualizări și îmbunătățiri continue de performanță de către o comunitate mare de dezvoltatori, precum și corecții de securitate, asigurând ca PHP rămâne un limbaj performant și relevant.

Toate aceste caracteristici determină popularitatea și accesibilitatea PHP-ului, fiind considerat și în prezent un limbaj adecvat pentru dezvoltarea aplicațiilor web interactive și dinamice.

### 2.1.3. PHP în dezvoltarea web

PHP este un limbaj important în dezvoltarea de aplicații web datorită scalabilității, flexibilității și posibilității oferite de a crea aplicații web dinamice și interactive. Cu ajutorul PHP se pot construi aplicații web sigure, rapide și ușor de întreținut datorită evoluții constante a limbajului. PHP facilitează diferite aspecte ale dezvoltării web, cum ar fi: generarea de conținut dinamic, integrarea cu baze de date, utilizarea framework-urilor PHP, resurse PHP, securitate și suport pentru servicii web.

Spre deosebire de HTML care este static, paginile PHP au posibilitatea de a genera conținut personalizat în mod automat prin afișarea informațiilor dintr-o bază de date, personalizarea mesajelor pentru utilizatori sau prin actualizarea conținutului paginilor in timp real. Toate acestea contribuie la utilitatea și interactivitatea site-urilor web create folosind PHP.

Un framework PHP este creat cu ajutorul limbajului PHP care oferă caracteristici integrate care îmbunătățesc experiența unui dezvoltator software. Acestea ajută la creșterea vitezei de dezvoltare și a performanței în general. Câteva dintre framework-urile populare sunt: Symfony, Laravel și Codelgniter. Fiecare framework se focusează pe diferite aspecte, unele se focusează pe îmbunătățirea fluxului de lucru, pe când altele țin cont mai mult de oferirea scalabilității și a securității.

### 2.1.4. Securitatea

Un aspect important în dezvoltarea aplicațiilor web îl reprezintă securitatea pentru asigurarea securității PHP oferă mecanisme de protejare a aplicațiilor și de diminuare a vulnerabilităților. Pentru asigurarea securității aplicațiilor web e necesară o bună cunoștință a acestor mecanisme și aplicarea diferitelor măsuri de securitate.

Una dintre metodele de a asigura securitatea o reprezintă folosirea unor funcții precum filter\_input() și filter\_var(), cu ajutorul cărora programatorii pot verifica și valida datele primite de la utilizatori. Prin validarea și curățarea datelor de intrare se pot preveni atacuri de tip injecție sau exploatări bazate pe input rău intenționat. Injecțiile SQL sunt o tehnică comună de atac al aplicațiilor web. Pentru a le putea evita e nevoie ca programatorii să implementeze interogări care să asigure faptul că datele primite de la utilizatori sunt tratate ca parametri, nu ca parte a comenzii SQL. Acest lucru se poate realiza prin intermediul extensiilor PDO (PHP Data Objects) sau MySQLi, suportate de limbajul PHP.

Alte metode importante în asigurarea securității sunt gestionarea sigură a sesiunilor și folosirea de funcții de criptare și decriptare. Gestionarea sigură a sesiunilor e necesară pentru integritatea și protecția datelor utilizatorilor. PHP conține mecanisme integrate pentru a simplifica gestionarea sesiunilor și funcții cu ajutorul cărora se pot crea, distruge și regenera ID-uri de sesiune (exemplu: session\_start(), unset()). Funcția session\_regenerate\_id() poate fi folosită pentru a preveni atacuri de tip hijack al sesiunii. De asemenea, PHP pune la dispoziția programatorilor funcții pentru criptarea și decriptarea datelor cu caracter sensibil (exemplu: password\_hash() și password\_verify()). Aceste funcții sunt folosite pentru criptare și verificarea parolelor în siguranță, nepermițând accesul neautorizat.

Prin folosirea acestor metode se asigură dezvoltarea unor aplicații web robuste și sigure, protejate de atacuri cibernetice și care oferă integritatea datelor utilizatorilor.

### 2.1.5. Instrumente și resurse pentru dezvoltare

Dezvoltarea aplicațiilor web cu ajutorul PHP se poate realiza cu ajutorul unui set de instrumente și resurse ce simplifică procesul de scriere și gestionare a codului. Aceste instrumente includ editoare de cod IDE-uri, biblioteci și resurse online.

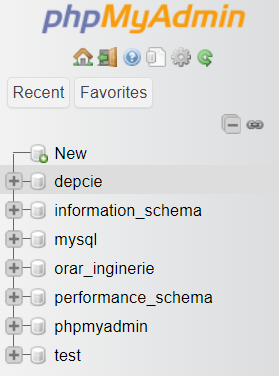
Printre cele mai populare editoare și IDE-uri se numără PHPStorm și Visual Studio Code. PHPStorm e un IDE dezvoltat de JetBrains, special pentru lucrul cu PHP, suportând framework-uri PHP, precum Laravel și Symfony. PHPStorm pune la dispoziție diferite facilități precum debugging ușor de realizat, integrarea bazelor de date, suport pentru testare și posibilitatea de refactorizare a codului și instrumente integrate de control al versiunilor (GIT, SVN). De asemenea, Visual Studio Code e un editor popular datorită flexibilității și numărului de extensii oferite. VS Code este un editor creat de Microsoft cu suport pentru PHP și multe alte limbaje de programare. La fel ca PHPStorm, VS Code are instrumente integrate care facilitează folosirea Git.

Un plus al folosirii PHP îl reprezintă comunitatea extinsă si resursele online disponibile. PFP.net este site-ul oficial al limbajului PHP care oferă o gamă largă de resurse, ghiduri, documentație cu exemple de cod și specificații detaliate care simplifică experiența dezvoltatorilor experimentați sau începători. Un beneficiu este, bineînțeles, faptul că documentația este actualizată constant și secțiunea de comentarii care permit utilizatorilor să își ofere feedback-ul și să contribuie la corectarea sau îmbunătățirea conținutului.

### 2.1.6. PhpMyAdmin

PhpMyAdmin este un instrument software open-source făcut pentru a ușura gestionarea și administrarea bazelor de date MySQL și MariaDB. Această platformă este realizată în principal cu ajutorul PHP. Popularitatea sa se datorează interfeței sale prietenoase și ușor de utilizat care îl face un tool ideal pentru programatorii începători sau avansați.

PhpMyAdmin oferă o interfață grafică intuitivă pentru crearea și gestionarea bazelor de date; ]n partea stângă a platformei sunt poziționate bazele de date cu butonul de “new” pentru crearea unei noi baze de date (fig. 1.1), iar în partea de sus avem un meniu cu diferitele funcții oferite de PhpMyAdmin (fig. 1.2).



(fig. 1.1)



(fig. 1.2)

Utilizatorii pot executa interogări SQL direct din platforma PhpMyAdmin folosind butonul “SQL”. Acest lucru este foarte avantajos pentru crearea, modificarea și ștergerea datelor din baza de date, oferind un mod rapid de a interacționa cu baza de date. O altă caracteristică importantă o reprezintă suportul integrat pentru import și export al datelor în diferite formate, precum, CSV, SQL, JSON și XML. Aceasta asigură o bună realizare de backup-uri și integrarea datelor cu alte aplicații. PhpMyAdmin oferă și o funcționalitate de “designer” care e folosită pentru vizualizarea și gestionarea grafică a tabelelor din baza de date. Aceasta permite vizualizarea tabelelor și a relațiilor dintre ele și a conectării prin cheile străine. Utilizatorii pot genera și diagrame PDF ale structurii bazei de date, acest lucru fiind util în realizarea unei documentații sau pentru partajarea acesteia.

## 2.2. SQL și MySQL

### 2.2.1. Introducere

MySQL este un sistem de gestionare al bazelor de date relaționale, popular printre dezvoltatorii de aplicații web. A fost dezvoltat de MySQLAB dar este în prezent deținut de Oracle Corporation. Este folosit pe scară largă deoarece oferă diferite beneficii, precum scalabilitate, performanță ridicată și flexibilitate, fiind ideal pentru o gestionare eficientă de date. Performanța ridica se datorează vitezei și eficienței în gestionarea volumelor mari de date care se pot extinde pe mai multe servere. Un alt avantaj îl reprezintă suportul pentru SQL pentru facilitarea manipulării și controlului de date.

SQL (Structured Query Language) este un limbaj standardizat de programare folosit în gestionarea bazelor de date. A fost creat de IBM în 1970, iar de atunci popularitatea sa a crescut, devenind standard pentru manipularea bazelor de date relaționale. SQL are o sintaxă relativ ușor de învățat. Sintaxa este declarativă, utilizatorii putând specifica acțiunile dorite asupra bazei de date fără a descrie detaliat modul de execuție al operației.

### 2.2.2. Structură și funcționalități

Structura MySQL e compusă din următoarele componente: serverul MySQL, motoare de stocare, clientele MySQL și biblioteci API. Serverul MySQL e nucleul care se ocupă de operațiile legate de baza de date, conexiunile utilizatorilor și gestionarea interogărilor. MySQL suportă diferite motoare de stocare, printre care InnoDB (implicit) și MyISAM (rapid, folosit mai ales pentru tabele care nu necesită tranzacții). Clientele MySQL sunt aplicații care fac posibilă interacțiunea dintre utilizatori și serverul MySQL (MySQL Command Line Client, MySQL Workbench). Bibliotecile API sunt oferite de MySQL pentru limbaje de programare PHP, Phyton, Java și altele, acestea ajutând în gestionare bazelor de date MySQL.

MySQL oferă funcționalități necesare în dezvoltarea de aplicații web. MySQL permite crearea, ștergerea, modificarea bazelor de date și inserarea, interogarea, actualizarea și ștergerea datelor din tabelele din baza de date . MySQL oferă posibilitatea de a gestiona utilizatorii și permisiunile oferite acestora.

### 2.2.3. Exemple de utilizare

Pasul inițial în folosirea MySQL este crearea unei baze de date si a tabelelor conținute de aceasta. Aceasta se poate realiza folosind comenzile SQL:

CREATE DATABASE magazinOnline;

USE magazinOnline;

CREATE TABLE produse\_din\_magazin (

id INT AUTO\_INCREMENT PRIMARY KEY,

nume VARCHAR(200),

preț INT,

stoc INT

);

(Ex. 1.1.)

Folosind aceste comenzi, am creat baza de date magazinOnline care conține tabelul produse\_din\_magazin. Tabelul conține un id, care e setat ca cheie primară care e incrementat automat și alte trei atribute care descriu produsele din magazin: nume, preț și stoc.

O altă operație des folosită este inserarea de date în tabele. În exemplul următor sunt inserate produsele telefon, cu prețul de 2000 și stocul de 35 și televizor cu prețul de 2500 și stocul de 20:

INSERT INTO produse\_din\_magazin(nume, preț, stoc) VALUES (‘telefon’, 2000, 35);

INSERT INTO produse\_din\_magazin(nume, preț, stoc) VALUES (‘televizor’, 2500, 20);

(Ex. 1.2.)

Alte operații importante în gestionarea bazelor de date sunt selectarea(SELECT), actualizarea(UPDATE) și ștergerea(DELETE) de date din tabele. Acestea sunt reprezentate în următorul exemplu unde selectăm toate produsele din tabelul de produse\_din\_magazin, actualizăm prețul telefonului la 2200 și ștergem televizorul:

SELECT \* FROM produse\_din\_magazin;

UPDATE produse\_din\_magazin SET preț = 2200 WHERE nume = ‘telefon’;

DELETE FROM produse\_din\_magazin WHERE nume = ‘televizor’;

(Ex. 1.3.)

## 2.3. HTML

### 2.3.1. Introducere

HTML, prescurtare pentru HyperText Markup Language, este limbajul standard utilizat pentru crearea și structurarea paginilor web. Folosind "tag-uri", HTML permite dezvoltatorilor să organizeze și să prezinte conținutul pe internet într-un mod structurat. Inventat de Tim Berners-Lee în 1991, HTML a început cu doar 18 elemente. De atunci, a evoluat semnificativ, cu actualizări importante precum HTML 2.0 în 1995, HTML 3.2 în 1997 care a adus suport pentru tabele, HTML 4.01 în 1999 care a introdus suport pentru scripturi și HTML5 în 2014 care a adăugat capabilități pentru audio, video și elemente grafice avansate.

### 2.3.2. Structură de bază a unui document

Un document HTML are o structura standard, cu elemente specifice care definesc aspectul paginii web. La această structură se adaugă ulterior elemente noi în funcție de dorințele și cerințele dezvoltatorilor web. Structura unui document este afișată și descrisă mai jos.

<!DOCTYPE html>

<html lang="ro">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document HTML Exemplu</title>

</head>

<body>

<h1>Bun venit la HTML</h1>

<p>Aceasta este o pagină web de bază.</p>

</body>

</html>

Tipul documentului și versiunea HTML este declarată folosind <!DOCTYPE html>. Tagul <html> este elementul rădăcină, care conține toate celelalte elemente HTML. Tagul <head> include metadate despre document, ca de exemplu titlu și link-uri către resurse externe. Tagul <title> e folosit pentru a da titlul browserului, <meta charset=”UTF-8”> e utilizat pentru a specifica standardul de codare a caracterelor și în interiorul tagurilor <body> e structurat conținutul efectiv al paginii web (text, imagini, link-uri și alte elemente).

### 2.3.3. Elemente HTML

Elementele HTML se referă la blocurile de bază ale unui document HTML. Fiecare bloc este creat folosind un tag de deschidere (< >) și unul de închidere (</ >) și conținutul este plasat între aceste taguri. Titlurile pot fi specificare, folosind tagurile ‘h1’ până la ‘h6’ (exemplu: <h1> Titlu</h1>). Un paragraf se poate defini prin tagul ‘p’ (<p>Paragraful respectiv</p>). Listele ordonate și neordonate pot fi reprezentate cu tagurile ‘ul’, respective ‘ol’ și ‘li’ definește un element dintr-o listă. Listele sunt elemente importante a HTML, putând fi utilizate în construcția meniului unei pagini web. Alte elemente HTML importante sunt link-rile (<a href=”https://www.link.com”>link.com</p>) și imaginile (<img src=”imagine.jpg” alt=”descriere imagine” >).

Atributele au și ele un rol important în HTML: ‘href’ specifică URL-ul care conduce către link, ‘src’ specifică sursa imaginii, ‘alt’ e folosit pentru a oferi un text alternativ afișat dacă nu se poate încărca imaginea dorită, ‘class’ și ‘id’ sunt folosite pentru a adăuga stil și identificator unic elementelor HTML.

### 2.3.4. Practici de utilizare HTML

Pentru a crea cod ușor de înțeles și menținut, există diferite practici bune care se utilizează. Structurarea clară a codului e absolut necesară pentru a obține un astfel de cod. Utilizarea de indentare corespunzătoare și structurarea codului HTML în mod logic, cu subsecțiuni clare sunt tehnici apreciate. O altă practică comună este utilizarea tag-urilor semantice, precum “<header>”, “<nav>”, “<footer>”, “<section>”, “<article>”, acestea îmbunătățind claritatea și accesibilitatea codului HTML. Un use case al acestora e reprezentat in Ex. 1.4.

<header>

<h1>Titlu Site</h1>

<nav>

<ul>

<li><a href="#home">Acasă</a></li>

<li><a href="#about">Despre</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</nav>

</header>

<section>

<article>

<h2>Articol Principal</h2>

<p>Conținutul articolului.</p>

</article>

</section>

<footer>

<p>© 2024 Compania Mea</p>

</footer>

(Ex. 1.4.)

Alte practici aplicate sunt minimizarea codului HTML prin eliminarea spațiilor și comentariilor nefolositoare. Aceasta duce la reducerea fișierului HTML și ca urmare, la optimizarea și creșterea performanței.

Toate acestea fac din HTML un limbaj fundamental în dezvoltarea de aplicații web. Aplicarea acestor practici e esențială în crearea de site-uri web eficiente, moderne, accesibile și interactive.

## 2.4. CSS

### 2.4.1. Introducere

CSS (Cascading Style Sheets) este un limbaj de stil utilizat pentru a descrie aspectul unui document HTML. CSS facilitează gestionarea paginilor web deoarece permite programatorilor să separe conținutul paginilor de aspectul acestora. Prima versiune CSS a fost creată în 1996 de către W3C. Acesta conținea elemente de aspect fundamentale, precum culori, fonturi, margini. În anul 1998 a apărut versiunea CSS2, care a introdus funcționalități avansate, cum ar fi, poziționarea relativă și absolută. Versiunea CSS2.1, apărută în 2011, a corectat unele erori ale versiunii anterioare și a îmbunătățit specificațiile. Versiunea CSS3, care se află în curs de dezvoltare, permite dezvoltarea mai rapidă a funcționalităților, prin aducerea de îmbunătățiri semnificative. Această versiune include selectoare avansate, efecte de tranziții și animații.

### 2.4.2. Sintaxa și selectoarele CSS

CSS are o sintaxă simplă, creată din selectoare și declarații. Declarațiile sunt formate din proprietăți și valori. Această structură este reprezentată în exemplul Ex. 1.1.

Selectoarele CSS sunt folosite pentru a selecta elementele HTML pe care dezvoltatorul dorește să le stilizeze. Există diverse selectoare CSS care sunt utilizate în funcție de ceea ce se dorește a fi selectat. Selectoarele de tip sunt folosite pentru a selecta toate elementele de tipul respectiv (Ex. 1.2.). Selectoarele de clasă selectează toate elementele cu o anumită clasă (Ex. 1.3.). Selectoarele de ID selectează un element unic cu un anumit ID (Ex. 1.4.). Selectoarele de atribut selectează elementele pe baza unui atribut (Ex. 1.5.). Selectoarele de descendență selectează elemente pe baza relațiilor lor cu alte elemente (Ex. 1.6.). Pseudo- clasele selectează elementele pe baza uni anumit statut sau condiție (Ex. 1.7.). Pseudo-elementele sunt utilizate pentru a stiliza anumite părți dintr-un element (Ex. 1.8.).

selector {

proprietate: valoare;

}

(Ex.1.1)

p {

color: blue;

}

(Ex. 1.2.)

.important {

font-weight: bold;

}

(Ex. 1.3.)

#header {

background-color: gray;

}

(Ex. 1.4.)

input[type="text"] {

border: 1px solid black;

}

(Ex. 1.5.)

div p {

margin: 20px;

}

(Ex. 1.6.)

a:hover {

color: red;

}

(Ex. 1.7.)

p::first-line {

font-size: 1.2px;

}

(Ex. 1.8.)

### 2.4.3. Proprietăți CSS esențiale

Câteva dintre cele mai importante proprietăți CSS folosite în stilizarea paginilor web sunt setarea de culori și fundaluri (Ex. 1.9.), fonturi și text (Ex. 2.1.), margini și spațieri (Ex. 2.2.), borduri și umbre (Ex. 2.3.), afișare și vizibilitate (Ex. 2.4.).

Pentru setarea de culori și fonturi se folosesc următoarele : ‘body’ cu care se setează culoarea textului unui element, ‘background-color’ setează culoarea de fundal al unui element, ‘background-image’ adaugă o imagine de fundal al unui element. Pentru setarea de fonturi și text sunt utilizate proprietățile: ‘font-family’ care specifică familia de fonturi pentru text, ‘font-size’ setează dimensiunea textului, ‘font-weight’ controlează grosimea fontului, ‘text-align’ aliniază textul în interiorul unui element, ‘text-decoration’ adaugă sau elimină decorarea textului. Pentru setarea de margini și spațieri se folosesc proprietățile: ‘margin’ setează spațiul exterior al unui element, ‘padding’ setează spațiul interior al elementelor. Pentru setarea de bordurilor și a umbrelor se folosesc: ‘border’ setează toate proprietățile bordurii (lățime, stil, culoare etc.), ‘border-radius’ setează colțurile rotunjite ale unui element, ‘box-shadow’ adaugă o umbra unui element. Afișarea și vizibilitatea sunt gestionate cu ‘display’ care controlează modul în care este afișat un element și ‘visibility’ care controlează vizibilitatea unui element.

body {

background-color: #f0f0f0;

color: #333;

}

(Ex. 1.9)

h1 {

font-family: 'Arial', sans-serif;

font-size: 2px;

text-align: center;

}

(Ex. 2.1)

p {

margin: 10px 0;

padding: 5px;

}

(Ex. 2.2)

div {

border: 1px solid #ccc;

}

(Ex. 2.3)

.block-element {

display: block;

}

(Ex. 2.4)

### 2.4.4. Practici de utilizare CSS

Pentru a asigura obținerea unui cod clar, eficient și ușor de gestionat este esențial să se urmeze anumite practici și tehnici bine definite. De asemenea, utilizarea de framework-uri CSS ajută la îmbunătățirea procesului de dezvoltare a aplicațiilor web interactive și cu un aspect modern.

O practică bună des întâlnită de organizare a codului CSS este păstrarea codului CSS in fișiere separate de cele de HTML. Aceasta determină obținerea de cod reutilizabil și ușor de întreținut și totodată îmbunătățește performanța, reducând timpul de încărcare a paginilor. O altă tehnică apreciată este utilizarea comentariilor pentru a oferi explicații suplimentare unde este cazul. Comentariile ajută atât programatorii noi care doresc să continue sau să modifice codul, cât și autorul codului. Utilizarea de clase și ID-uri conferă reutilizabilitate și lizibilitate codului, acestea putând fi aplicate pentru mai multe elemente.

Bootstrap este un framework des folosit, creat de Twitter. Acest framework oferă un set mare de componente predefinite și flexibilitate care simplifică procesul de dezvoltare a site-urilor web. Bootstrap este ușor de utilizat și este avantajos deoarece oferă multe componente reutilizabile, dar are ca dezavantaj dimensiunea mare a fișierelor CSS și JS. Alte framework-uri populare sunt Foundation, Bulma și Tailwind CSS.

## JavaScript

### 2.5.1. Introducere

JavaScript este un limbaj de programare interpretat, orientat pe obiecte, utilizat predominant pentru dezvoltarea de aplicații web interactive. Creat în 1995 de Brendan Eich în timp ce lucra la Netscape Communications, JavaScript a evoluat rapid pentru a deveni unul dintre cele mai esențiale limbaje de programare în dezvoltarea web. Spre deosebire de HTML și CSS, care sunt folosite pentru structura și stilul paginilor web, JavaScript permite adăugarea de comportamente dinamice și interactive.

Inițial, JavaScript a fost conceput pentru a rula pe partea clientului (client-side), ceea ce înseamnă că codul JavaScript este executat direct în browserul utilizatorului. Cu toate acestea, odată cu dezvoltarea Node.js, JavaScript a extins funcționalitatea sa și pe partea serverului (server-side), permițând dezvoltatorilor să scrie cod JavaScript pentru întregul stack de aplicații web.

JavaScript este standardizat prin specificația ECMAScript (ES), care definește caracteristicile și comportamentele limbajului. De-a lungul anilor, au fost lansate mai multe versiuni ale ECMAScript, fiecare aducând îmbunătățiri și noi funcționalități limbajului.

### 2.5.2. Principiile fundamentale ale JavaSript

Sintaxa de bază a JavaScript este similară cu cea a altor limbaje de programare, cum ar fi C și Java, dar are și propriile sale particularități. JavaScript suportă diverse tipuri de date, inclusiv numere, șiruri de caractere (strings), obiecte și funcții. Una dintre caracteristicile cheie ale JavaScript este tipul său dinamic, ceea ce înseamnă că variabilele nu au un tip fix și pot fi reasignate la valori de tipuri diferite.

Mai jos sunt prezentate câteva exemple fundamentale de **declarații de variabile în JavaScript:**

let num = 42; // variabilă de tip număr

const pi = 3.14; // constantă de tip număr

let str = "Hello"; // variabilă de tip șir de caractere (string)

let isTrue = true; // variabilă de tip boolean

Funcțiile sunt blocuri de cod reutilizabile care pot fi definite și apelate în orice parte a programului. În JavaScript, funcțiile sunt considerate obiecte de primă clasă, ceea ce înseamnă că pot fi stocate în variabile, transmise ca argumente altor funcții și returnate de alte funcții.

Mai jos este prezentat un exemplu fundamental de **declarație** de funcție:

function greet(name) {

return "Bună, " + name + "!";

}

console.log(greet("Ana")); // Outputul va fi : Bună, Ana!

Manipularea DOM (Document Object Model) este una dintre cele mai comune utilizări ale JavaScript în dezvoltarea web. DOM reprezintă structura documentului HTML ca un arbore de noduri, iar JavaScript poate interacționa cu aceste noduri pentru a modifica conținutul, structura și stilul paginii web. De exemplu, dezvoltatorii pot folosi JavaScript pentru a adăuga sau elimina elemente HTML, pentru a schimba atributele și stilurile CSS ale elementelor existente și pentru a răspunde la evenimentele de utilizator, cum ar fi clicurile și tastările.

Mai jos sunt prezentate câteva exemple fundamentale de **declarații de** selecție și modificare a elementelor DOM:

<!DOCTYPE html>

<html>

<head>

<title>DOM Manipulation</title>

</head>

<body>

<div id="myDiv">Hello World!</div>

<button id="myButton">Click me</button>

<script>

// Selecția elementelor DOM

let div = document.getElementById("myDiv");

let button = document.getElementById("myButton");

// Modificarea conținutului

div.innerHTML = "Hello JavaScript!";

// Adăugarea unui eveniment de clic

button.addEventListener("click", function() {

div.style.color = "red"; // Modificarea stilului

});

</script>

</body>

</html>

### 2.5.3. JavaScript în dezvoltarea aplicațiilor web

* Integrarea cu HTML și CSS

JavaScript este esențial pentru crearea de aplicații web interactive și dinamice. Prin integrarea sa cu HTML și CSS, JavaScript permite dezvoltatorilor să manipuleze documentul web și să răspundă la acțiunile utilizatorilor în timp real. De exemplu, dezvoltatorii pot folosi JavaScript pentru a valida formulare sau pentru a crea meniuri derulante.

JavaScript poate fi inclus în paginile HTML fie direct în cadrul etichetelor <script>, fie ca fișiere externe. Utilizarea fișierelor externe este o practică recomandată deoarece permite separarea clară a logicii de scripting de structura documentului și de stiluri, ceea ce îmbunătățește mentenabilitatea și performanța paginilor web.

Un alt aspect important al integrării JavaScript cu HTML și CSS este manipularea DOM. Dezvoltatorii pot folosi metode precum getElementById, querySelector sau addEventListener pentru a selecta elemente din document și a le modifica sau pentru a adăuga ascultători de evenimente (event listeners) care declanșează acțiuni în răspuns la interacțiunile utilizatorilor.

* Utilizarea JavaScript pentru Aplicații Single Page (SPA)

Aplicațiile Single Page (SPA) au devenit din ce în ce mai populare datorită experienței de utilizare fluentă și rapidă pe care o oferă. Într-o aplicație SPA, tot conținutul necesar este încărcat o singură dată și orice interacțiune ulterioară cu aplicația nu necesită reîncărcarea paginii. JavaScript joacă un rol central în construirea SPA-urilor prin gestionarea rutării (routing) și actualizarea dinamică a conținutului paginii.

Framework-urile și bibliotecile JavaScript precum Angular, React și Vue.js au revoluționat modul în care sunt dezvoltate SPA-urile. Aceste instrumente oferă structuri și paradigme de dezvoltare bine definite, care facilitează crearea de componente reutilizabile, gestionarea stării aplicației și optimizarea performanței.

* Securitate

Odată cu puterea și flexibilitatea pe care le oferă JavaScript vin și provocări de securitate. Atacuri precum Cross-Site Scripting (XSS) și Cross-Site Request Forgery (CSRF) sunt comune în aplicațiile web și pot compromite securitatea datelor utilizatorilor și a aplicațiilor. Este esențial ca dezvoltatorii să fie conștienți de aceste riscuri și să implementeze măsuri de securitate adecvate.

Pentru a preveni atacurile XSS, dezvoltatorii ar trebui să valideze și să igienizeze toate datele de intrare ale utilizatorilor. Utilizarea funcțiilor de escape atunci când se afișează datele introduse de utilizatori în paginile web este crucială pentru a preveni injectarea de cod malițios. De asemenea, setările corespunzătoare ale antetelor HTTP, cum ar fi Content Security Policy (CSP), pot ajuta la blocarea conținutului nesigur.

CSRF poate fi prevenit prin utilizarea token-urilor anti-CSRF. Aceste token-uri sunt generate pe server și incluse în formularele web, astfel încât orice cerere autentică trimisă de utilizator trebuie să conțină un token valid. Această măsură asigură că cererile sunt trimise doar de surse de încredere.

Implementarea acestor practici de securitate și adoptarea unor best practices generale pentru dezvoltarea JavaScript, cum ar fi modularizarea codului și testarea automată, contribuie la crearea de aplicații web robuste și sigure.

În concluzie, JavaScript este un limbaj de programare versatil și esențial în dezvoltarea web modernă. De la principii fundamentale și manipularea DOM până la utilizarea tehnicilor avansate și dezvoltarea aplicațiilor complexe, JavaScript continuă să evolueze și să ofere noi capabilități pentru dezvoltatori. Înțelegerea profundă a acestui limbaj și adoptarea unor practici de securitate adecvate sunt esențiale pentru crearea de aplicații web interactive, performante și sigure.

## XAMPP

### 2.6.1. Introducere

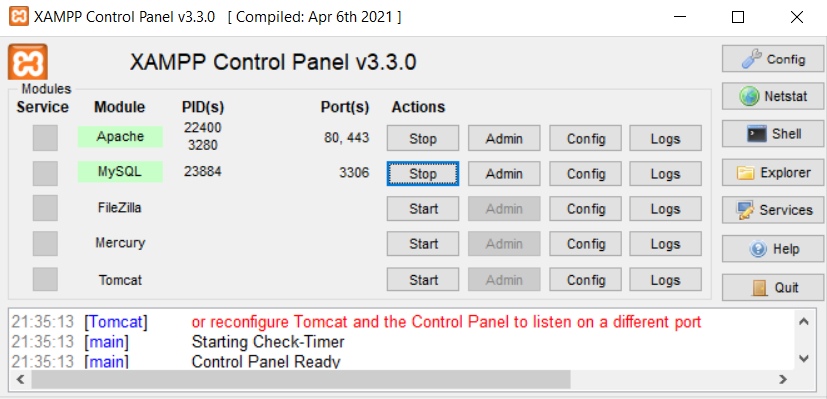
XAMPP este un pachet de software open-source care oferă o soluție pentru dezvoltarea și testarea aplicațiilor web. Numele XAMPP este acronim pentru X(platformă multiplă), Apache, MySQL, PHP și Perl. XAMPP este un pachet software ce pune la dispoziția dezvoltatorilor web un server Apache, un sistem prin care se pot gestiona bazele de date MySQL și suport pentru limbajele de programare PHP și Perl. Toate acestea fac din XAMPP un software ideal pentru dezvoltatorii web.

XAMPP a fost dezvoltat de compania Apache Friends în anul 2002. De-a lungul anilor a evoluat, devenind unul dintre cele mai populare pachete de software folosite în dezvoltarea aplicațiilor și a site-urilor web. XAMPP este ușor de utilizat și este suportat pe mai multe sisteme de operare, inclusiv Windows, Linux și macOS. XAMPP este actualizat constant pentru a suporta noile tehnologii și versiuni ale componentelor pe care le pune la dispoziție.

### 2.6.2. Instalarea și configurarea XAMPP

XAMPP poate fi descărcat de pe site-ul Apache Friends oficial, unde se pot găsi versiuni pentru Windows, Linux și macOS. Pentru instalare trebuie urmați câțiva pași simpli: selectarea componentelor care se doresc instalate, alegerea directorului de instalare (implicit, C:\xampp).

Panoul de Control XAMPP este interfața principală de unde sunt gestionate serviciile. După cum se poate observa din figura fig. 1.1., panoul de control conține butoane de start și stop al componentelor Apache, MySQL, FileZilla, Mercury și Tomcat (Apache și MySQL sunt pornite). În plus, fiecare serviciu are un buton care deschide fișierele de configurare, deci permite editarea și configurarea acestora. În consola de jos se pot observa mesaje relevante.



Apache poate fi configurat accesând fișierul principal de configurare (httpd.conf), unde se pot seta diferite configurații, precum porturile pe care rulează. De exemplu, schimbarea portului de la 80 (implicit) la 8080 se poate face folosind: Listen 8080. Configurarea MySQL se face în fișierul my.ini; acesta poate fi accesat apăsând pe butonul de config de la MySQL sau din directorul ‚xampp/mysql/bin’. De aici se pot modifica setările de memorie, cache și alte setări avansate. De exemplu, pentru a schimba portul de la 3306 (portul implicit) la 3307 se folosește: port=3307. Configurarea PHP se face în fișierul php.ini, care poate fi accesat la butonul de config din dreptul Apache sau în directorul ‚xampp’. Aici se pot configura memoria, extensiile încărcate și alte opțiuni. De exemplu pentru mărirea limitei de încărcare a fișierelor la 50M se utilizează: upload\_max\_filesize=50M;

### 2.6.3. XAMPP în dezvoltarea și testarea aplicațiilor web

Crearea unui proiect folosind platforma XAMPP este relativ simplă. În directorul ‘htdocs’, situat în directorul de instalare XAMPP se vor plasa toate fișierele proiectului creat. O tehnică apreciată este crearea unui folder nou pentru fiecare proiect, pentru a menține o structură și o ordine clară. Apoi în folder-ul proiectului curent se vor crea fișierele de HTML, PHP, JS și altele, în funcție de preferințele dezvoltatorilor. Accesul la proiectul creat se face pornind serverul Apache și MySQL din panoul de control XAMPP, după care din browser se poate accesa ‘http://localhost/numele\_proiectului’ pentru a vedea interfața grafică a acestuia.

Administrarea bazelor de date poate fi realizată din interfața phpMyAdmin. XAMPP include phpMyAdmin, care poate fi accesat din browser cu link-ul ‘http://localhost/phpmyadmin’. Cu ajutorul phpMyAdmin se pot gestiona bazele de date MySQL sau MariaDB. Prin interfața phpMyAdmin se facilitează modificarea, crearea și ștergerea bazelor de date și a tabelelor conținute de acestea. Crearea unei baze de date se pate face din fila ‘Databases’ prin introducerea unui nume pentru baza de date și apăsarea butonului ‘Create’. Tabelele din baza de date pot fi create folosind interogări SQL. PhpMyAdmin permite executarea acestora direct din interfața sa. La fila ‘SQL’ se poate introduce codul SQL și se poate executa prin apăsarea butonului ‘Go’.

Pentru logare și debugging, XAMPP are fișiere de log și modul de debugging PHP. XAMPP conține fișiere de log pentru toate serviciile, care pot fi accesate în două moduri, cu click pe butonul de Log din dreptul serviciului la care se dorește logare sau în subdirectoarele XAMPP. Prin verificarea fișierelor de log se pot identifica și rezolva probleme de configurare sau erori de executare. Cu ajutorul logurilor de Apache (error.log și access.log) se pot depista erori de server și se poate gestiona traficul web. Logurile MySQL (mysql\_error.log) sunt utilizate pentru a depista probleme legate de baza de date, precum erorile de conexiune. Erorile legate de PHP se găsesc în fișierul php.ini. Pentru a face debugg mult mai eficient se poate activa afișarea erorilor din php.ini(display\_errors=O, error\_reporting=E\_ALL). Aceasta permite o vizualizare rapidă și detaliată a erorilor.

XAMPP suportă managementul proiectelor și colaborarea prin sisteme de suport al versiunilor și posibilitatea configurării proiectelor multiple. Utilizarea sistemelor de control al versiunilor, de exemplu Git, în colaborare cu XAMPP este benefic pentru gestionarea codului și colaborare în echipă, dacă este cazul. Crearea unui repository de git în folderul proiectului și utilizarea unei platforme precum GitHub sau GitLab ajută la crearea de versiuni a proiectului, astfel menținând o structurare și o claritate a proiectului. Totodată, XAMPP susține gestionarea de proiecte multiple prin configurarea de virtual hosts în Apache. Astfel, se pot defini mai multe domenii locale care să direcționeze către diferite directoare de proiect:

<VirtualHost \*:80>

DocumentRoot ‘C:/xampp/htdocs/primul\_proiect’

ServerName primul\_proiect.local

</VirtualHost>

<VirtualHost \*:80>

DocumentRoot ‘C:/xampp/htdocs/alDoilea\_proiect’

ServerName alDoilea\_proiect.local

</VirtualHost>

După definirea domeniilor locale este necesar ca aceste domenii să fie adăugate în fișierul ‘hosts’ al sistemului de operare:

127.0.0.1 primul\_proiect.local

127.0.0.1 alDoilea\_proiect.local

În concluzie, XAMPP este o platformă esențială pentru dezvoltarea și testarea aplicațiilor web locale, deoarece oferă un pachet cu toate componentele necesare pentru crearea unei aplicații web completă și robustă. XAMPP este avantajos pentru dezvoltatori, deoarece le permite o instalare și configurare ușoară, fără a necesita gestionarea efectivă a serverului. Alte beneficii care sporesc popularitatea XAMPP este interfața intuitivă, suportul pentru diverse sisteme de operare (Windows, Linux, macOS), simularea unui server web, flexibilitate și scalabilitate obținută prin configurarea de proiecte multiple.

## 2.7. Git

### 2.7.1. Introducere

Git este un sistem distribuit de control al versiunilor dezvoltat în anul 2005vde către Linus Torvalds. A fost creat inițial pentru a sprijini dezvoltarea kernelului Linux. Git este eficient în gestionarea proiectelor deoarece permite utilizatorilor să mențină un istoric al versiunilor de cod și să colaboreze între ei.

Git oferă multiple beneficii utilizatoriilor, câteva dintre acestea sunt:

• Controlul versiunilor

Git permite controlul versiunilor prin funcționalitatea de commit. Realizarea unui commit se referă la salvarea stării/versiunii codului la moment dat cu o descriere sugestivă a funcționalității implementate. Commiturile sunt folositoare, deoarece se poate observa ușor evoluția proiectului, modificările făcute și problemele apărute.

• Distribuitate

Fiecare clonă a unui repository conține istoricul complet al proiectului. Git oferă posibilitatea dezvoltatorilor de a lucra offline și de a-și sincroniza mai apoi modificările când au acces la rețea.

• Eficient și rapid

Eficiența Git constă în faptul că este ideal pentru gestionarea de proiecte mari, deoarece împărțirea taskurilor în commituri ajută la organizarea și mentenanța codului. Efectuarea de operații local este rapidă, iar dimensiunea repository-ului este redusă printr-un sistem de compresie a datelor.

• Gestionarea branch-urilor

O funcționalitate de bază a Git-ului este crearea de branch-uri. Branchu-rile izolează lucrul la o funcționalitate nouă sau un bug fix fără să afecteze codul branch-ului principal, unde se află codul pentru proiect până în momentul curent. Git permite o gestionare simplă și eficientă a branch-urilor prin operații de merge și rebase care facilitează integrarea modificărilor din diferite branch-uri.

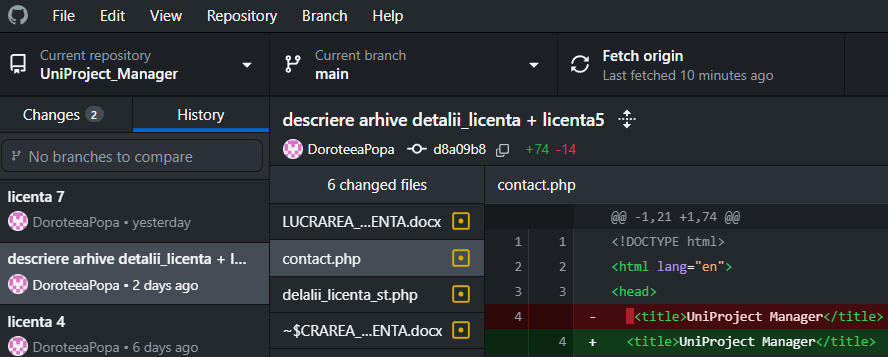
### 2.7.2. GitHub și GitHub Desktop

GitHub este o platformă ce conține repository-urile de Git. GitHub adaugă funcționalități suplimentare pentru gestionarea de proiecte și colaborarea între mai mulți utilizatori dacă este cazul. GitHub suportă stocarea, gestionarea și revizuirea codului într-un mod organizat și eficient. Totodată, GitHub oferă câteva funcționalități cheie, cum ar fi: repository-uri remote, pull request-uri și issue tracking.

Prin intermediul repository-ului remote, GitHub oferă accesibilitate la repository-uri oriunde și oricând. Acestea pot fi publice sau private, aceasta reoferindu-se la modul de acces asupra lor. Cele private sunt disponibile tuturor utilizatorilor GitHub, ceea ce o face o platformă ideală pentru proiectele open-source. Pull request-urile sunt esențiale pentru lucrul în echipă, permițând revizuiri și modificări la codul brancu-rilor înainte să fie integrate în branch-ul principal. Astfel se evită introducerea de bug-uri sau funcționalități nedorite în proiect. Pe lângă acestea, GitHub un sistem integrat de urmărire al problemelor. Acesta ajută dezvoltatorii să gestioneze task-urile în funcție de prioritate și să urmărească progresul făcut.

GitHub Desktop este o aplicație grafică pentru GitHub. Interfața intuitivă simplifică utilizarea Git și eficientizează procesul de versionare și gestionare al proiectelor. GitHub Desktop este disponibil pentru Windows și macOS și oferă funcționalități avansate pentru gestionarea repository-urilor de Git.

GitHub Desktop are o interfață prietenoasă pentru utilizatori, afișând într-un mod clar modificările, commit-urile, branch-urile și celelalte funcționalități. Astfel, dezvoltatorii pot vizualiza și gestiona modificările din proiecte și pot crea, integra și schimba cu ușurință între branch-uri și proiecte. În imaginea de mai jos, putem observa mai multe elemente și funcționalități ale interfeței GitHub Desktop.



Bara de navigare principală conține meniuri precum File, Edit, View, Repository, Branch și Help, care oferă acces la diverse funcționalități și setări ale GitHub Desktop. Meniul File oferă opțiuni pentru a deschide, clona sau crea repository-uri noi, Edit include opțiuni de modificare a setărilor și preferințelor, View permite ajustarea aspectului interfeței și vizualizarea detaliilor commit-urilor, Repository permite gestionarea repository-urilor, cum ar fi sincronizarea și setările remote, Branch oferă opțiuni pentru gestionarea branch-urilor, cum ar fi crearea, schimbarea și ștergerea acestora, iar Help oferă acces la documentație și asistență.

În partea de sus a interfeței, vedem informații despre repository-ul curent, numit UniProject\_Manager, și branch-ul curent selectat, numit main. Utilizatorii pot schimba repository-urile folosind meniul derulant și pot schimba între branch-uri existente sau pot crea branch-uri noi. De asemenea, funcționalitatea Fetch origin permite utilizatorilor să sincronizeze repository-ul local cu cel remote, obținând ultimele modificări de pe GitHub. Indicatorul Last fetched 10 minutes ago arată ultima dată când repository-ul a fost sincronizat.

Panoul de modificări afișează modificările nesalvate (unstaged changes) în repository. Utilizatorii pot vedea fișierele modificate și pot selecta care dintre acestea să fie incluse într-un commit. Aceștia pot face click pe fiecare fișier pentru a vedea diferențele (diff) linie cu linie, comparând modificările față de versiunea anterioară. De asemenea, există o secțiune pentru istoricul commit-urilor, care oferă o vedere de ansamblu asupra istoricului commit-urilor din repository. Fiecare commit este afișat cu mesajul său, autorul și timestamp-ul.

Lista de fișiere modificate afișată în panoul de modificări arată fișierele care au fost modificate, create sau șterse. Fiecare fișier are o legendă de culori care indică tipul modificărilor, de exemplu, verde pentru adăugări și roșu pentru ștergeri. Utilizatorii pot deschide fiecare fișier modificat pentru a vedea diferențele. Linia originală este afișată în roșu, iar linia modificată este afișată în verde. Această funcționalitate este esențială pentru a înțelege exact ce modificări au fost făcute înainte de a le include într-un commit.

Interfața GitHub Desktop este concepută pentru a facilita gestionarea repository-urilor Git printr-o abordare vizuală și intuitivă. Utilizatorii pot naviga ușor între diferite repository-uri și branch-uri, pot vizualiza și gestiona modificările, pot sincroniza repository-urile locale cu cele remote și pot colabora eficient cu alți membri ai echipei. Funcționalitățile de vizualizare a diferențelor și istoricul commit-urilor oferă o transparență completă asupra evoluției codului, ceea ce este esențial pentru menținerea calității și coerenței proiectelor software.

### 2.7.3. Mod de utilizare git

Utilizarea Git în mod eficient necesită înțelegerea unor fluxuri de lucru comune și adoptarea unor bune practici care să asigure coerența și calitatea codului.

* Fluxuri de lucru Git

Unul dintre cele mai populare fluxuri de lucru este Git Flow, un model de dezvoltare bazat pe branch-uri, propus de Vincent Driessen. Acesta include branch-uri pentru dezvoltare, funcționalități, release-uri și bug fix-uri, permițând un flux de lucru organizat și structurat. În Git Flow, branch-ul main conține versiunea stabilă și lansată a proiectului, iar branch-ul develop conține codul pentru dezvoltarea curentă și este folosit pentru integrarea continuă. Branch-urile de funcționalitate (feature) sunt folosite pentru dezvoltarea funcționalităților noi, care sunt apoi integrate în develop. Branch-urile de release (release) sunt folosite pentru pregătirea versiunilor de producție, incluzând ultimele modificări din develop. Branch-urile de bug fix (hotfix) sunt folosite pentru rezolvarea rapidă a bug-urilor critice din main.

* Practici apreciate în utilizarea Git

Adoptarea unor bune practici în utilizarea Git este esențială pentru menținerea calității codului și a eficienței în colaborare. Una dintre cele mai importante practici este scrierea de mesaje de commit clare și descriptive. Un mesaj de commit ar trebui să explice clar ce modificări au fost făcute și de ce, pentru a facilita înțelegerea istoricului proiectului de către toți membrii echipei. Este recomandat să folosiți un format standard pentru mesajele de commit, care să includă un titlu concis urmat de o descriere detaliată.

De asemenea, este esențial să se facă commit-uri frecvente și mici. Aceasta permite urmărirea mai ușoară a modificărilor și facilitează identificarea și rezolvarea problemelor. Commit-urile ar trebui să fie atomice, adică să conțină o singură modificare logică sau funcționalitate. Evitați commit-urile mari și complexe care îngreunează revizuirea și înțelegerea modificărilor.

Utilizarea branch-urilor este o altă bună practică importantă. Creați branch-uri separate pentru fiecare funcționalitate nouă sau bug fix, pentru a izola modificările și a facilita testarea și integrarea. Numiți branch-urile într-un mod clar și descriptiv, pentru a reflecta scopul lor

Sincronizarea regulată a branch-urilor cu repository-ul remote este crucială pentru a menține consistența codului și a evita conflictele. Utilizați comanda git pull pentru a aduce modificările recente din branch-ul remote în branch-ul local și rezolvați conflictele imediat ce apar. De asemenea, utilizați git push pentru a trimite modificările locale către repository-ul remote, asigurând că toți membrii echipei lucrează cu cea mai recentă versiune a codului.

În cele din urmă, revizuirea codului este o practică esențială pentru menținerea calității și securității codului. Utilizați pull request-uri pentru a solicita revizuirea modificărilor înainte de a le integra în branch-ul principal. Încurajați membrii echipei să lase comentarii și sugestii pe pull request-uri, și asigurați-vă că toate problemele și conflictele sunt rezolvate înainte de a face merge. Revizuirea codului ajută la identificarea problemelor potențiale și la îmbunătățirea calității generale a codului.

În concluzie. adoptarea unor fluxuri de lucru bine definite și a unor bune practici în utilizarea Git poate îmbunătăți semnificativ eficiența și calitatea dezvoltării software. Git Flow și Forking Workflow sunt două dintre cele mai populare modele de dezvoltare care facilitează colaborarea și gestionarea proiectelor. Prin scrierea de mesaje de commit clare, efectuarea de commit-uri frecvente și mici, utilizarea branch-urilor separate pentru funcționalități și bug fix-uri, sincronizarea regulată a branch-urilor și revizuirea codului prin pull request-uri, echipele de dezvoltare pot asigura un flux de lucru organizat și eficient, menținând în același timp calitatea și coerența codului sursă.

## 2.8. Visual Studio Code

### 2.8.1. Introducere

Visual Studio Code (VS Code) este un editor de cod sursă dezvoltat de Microsoft, care a devenit extrem de popular datorită funcționalităților sale puternice și flexibilității. Este gratuit, open-source și rulează pe diverse platforme, inclusiv Windows, macOS și Linux.

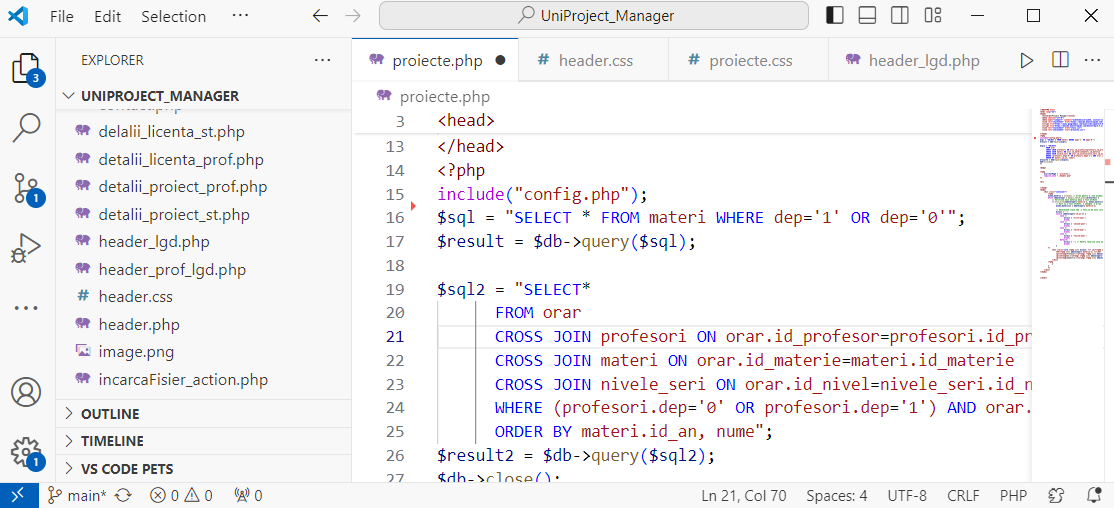
Visual Studio Code a fost lansat de Microsoft în aprilie 2015 și a devenit rapid unul dintre cele mai populare editoare de cod datorită caracteristicilor sale puternice și suportului extins pentru multiple limbaje de programare. Este conceput pentru a oferi o experiență de editare a codului eficientă și ușor de utilizat, integrând funcționalități avansate precum IntelliSense, debugging, și integrare Git.

Printre caracteristicile cheie ale Visual Studio Code se numără:

* **IntelliSense:** Completarea automată a codului bazată pe context și informații despre tipuri, oferind sugestii inteligente pentru completarea codului.
* **Debugging:** Un sistem integrat de debugging care permite dezvoltatorilor să își testeze și să își depaneze aplicațiile direct din editor.
* **Controlul Versiunilor:** Suport nativ pentru Git, facilitând gestionarea versiunilor și colaborarea în echipă.
* **Extensii:** O piață vastă de extensii care permite personalizarea și extinderea funcționalităților editorului pentru a se potrivi nevoilor specifice ale dezvoltatorilor.
* **Suport pentru Multiple Limbaje de Programare:** Visual Studio Code suportă o gamă largă de limbaje de programare, incluzând PHP, JavaScript, Python, Java, C++, și multe altele.

### 2.8.2. Interfața Visual Studio Code

Interfața Visual Studio Code este concepută pentru a fi intuitivă și ușor de utilizat, oferind acces rapid la funcționalitățile esențiale pentru dezvoltarea de software. În imaginea de mai jos, putem observa mai multe componente cheie ale interfeței.



**Explorer:** În partea stângă a interfeței, avem panoul Explorer, care afișează structura proiectului. Acesta permite utilizatorilor să navigheze între fișiere și directoare, să creeze, să șteargă sau să redenumească fișiere, și să deschidă fișiere pentru editare. În imagine, vedem că proiectul curent se numește "UNIPROJECT\_MANAGER", iar diverse fișiere PHP sunt listate în structura proiectului. Indicatorii pentru modificările nesalvate și numărul de modificări sau conflicte sunt vizibili, ajutând la gestionarea eficientă a proiectului.

**Tab-uri de Editare:** În partea de sus a interfeței, vedem tab-urile de editare, care permit utilizatorilor să lucreze simultan la mai multe fișiere deschise. În imagine, fișierul "proiecte.php" este deschis și editat. Utilizatorii pot comuta între fișierele deschise folosind aceste tab-uri.

**Editor de Cod:** Editorul de cod ocupă cea mai mare parte a interfeței și este locul unde utilizatorii scriu și editează codul. În imagine, vedem codul PHP din fișierul "proiecte.php". Codul include o instrucțiune "include" pentru a adăuga conținutul unui alt fișier PHP, o interogare SQL și o execuție a acestei interogări folosind un obiect de bază de date.

**Bara de Status:** În partea de jos a interfeței, vedem bara de status, care oferă informații rapide despre starea curentă a editorului și a proiectului. Aceasta poate afișa informații despre ramura Git curentă, erori și avertismente în cod, și alte detalii relevante. În imagine, vedem că ramura curentă este "main" și sunt afișate indicatoare pentru erori și avertismente, precum și detalii despre cod, cum ar fi limbajul de programare, formatul fișierului și setările de indentare.

**Panoul de Activități Laterale:** În partea stângă a interfeței, sub panoul Explorer, vedem panoul de activități laterale, care include pictograme pentru diverse funcționalități, cum ar fi explorarea fișierelor, controlul versiunilor, căutarea în proiect și accesul la extensii. Pictogramele vizibile includ Explorer, pentru navigarea între fișierele și directoarele proiectului, Search, pentru căutarea textului în întregul proiect, Source Control, pentru integrarea cu sistemul de control al versiunilor (Git în acest caz), și Extensions, pentru gestionarea extensiilor instalate și descărcarea de noi extensii.

### 2.8.3. Utilizarea în dezvoltarea web

Visual Studio Code este un instrument extrem de puternic pentru dezvoltarea web, datorită suportului său robust pentru limbaje de programare web precum HTML, CSS, JavaScript și framework-uri moderne. Un aspect important al utilizării Visual Studio Code în dezvoltarea web este suportul său excelent pentru PHP, un limbaj server-side popular pentru dezvoltarea aplicațiilor web.

Visual Studio Code oferă suport complet pentru editarea HTML și CSS, inclusiv completarea automată a tag-urilor, sugestii pentru atribute și valori, și verificarea sintaxei. Dezvoltatorii pot folosi extensii precum Live Server pentru a lansa un server local și a vedea modificările în timp real pe măsură ce editează fișierele HTML și CSS.

Visual Studio Code oferă un suport extins pentru PHP prin intermediul extensiilor disponibile pe marketplace-ul său. Visual Studio Code are funcționalități avansate precum IntelliSense, care oferă completarea automată a codului bazată pe context și informații despre tipurile de date, diagnosticare a erorilor în timp real și navigare ușoară prin cod. De asemenea, dezvoltatorii pot beneficia de suport pentru debugging integrat, permițându-le să seteze puncte de oprire, să examineze variabile și să urmărească fluxul de execuție al aplicației lor PHP direct din editor. Aceste funcționalități facilitează dezvoltarea rapidă și eficientă a aplicațiilor web bazate pe PHP, reducând timpul de depanare și îmbunătățind calitatea codului..

Un alt aspect important al dezvoltării web cu Visual Studio Code este debugging-ul și controlul versiunilor. Editorul include un debugger integrat care permite dezvoltatorilor să seteze puncte de oprire (breakpoints), să examineze variabile și să urmărească fluxul de execuție al aplicației lor. Suportul nativ pentru Git face gestionarea versiunilor codului simplă și eficientă, permițând dezvoltatorilor să facă commit-uri, să creeze branch-uri, să rezolve conflictele și să gestioneze pull request-uri direct din editor.

Visual Studio Code este un editor de cod sursă extrem de versatil și puternic, care oferă o gamă largă de funcționalități pentru a sprijini dezvoltatorii în toate aspectele procesului de dezvoltare. De la o interfață intuitivă și ușor de utilizat, până la suportul robust pentru multiple limbaje de programare și integrarea cu instrumente de debugging și control al versiunilor, Visual Studio Code este un instrument indispensabil pentru orice dezvoltator. În dezvoltarea web, acest editor se remarcă prin flexibilitatea și extensibilitatea sa, permițând dezvoltatorilor să creeze aplicații web complexe și performante într-un mod eficient și organizat.

# Arhitectura aplicației

## Introducere baza de date

## Descrierea bazei de date

### 3.2.1. Descriere tabele

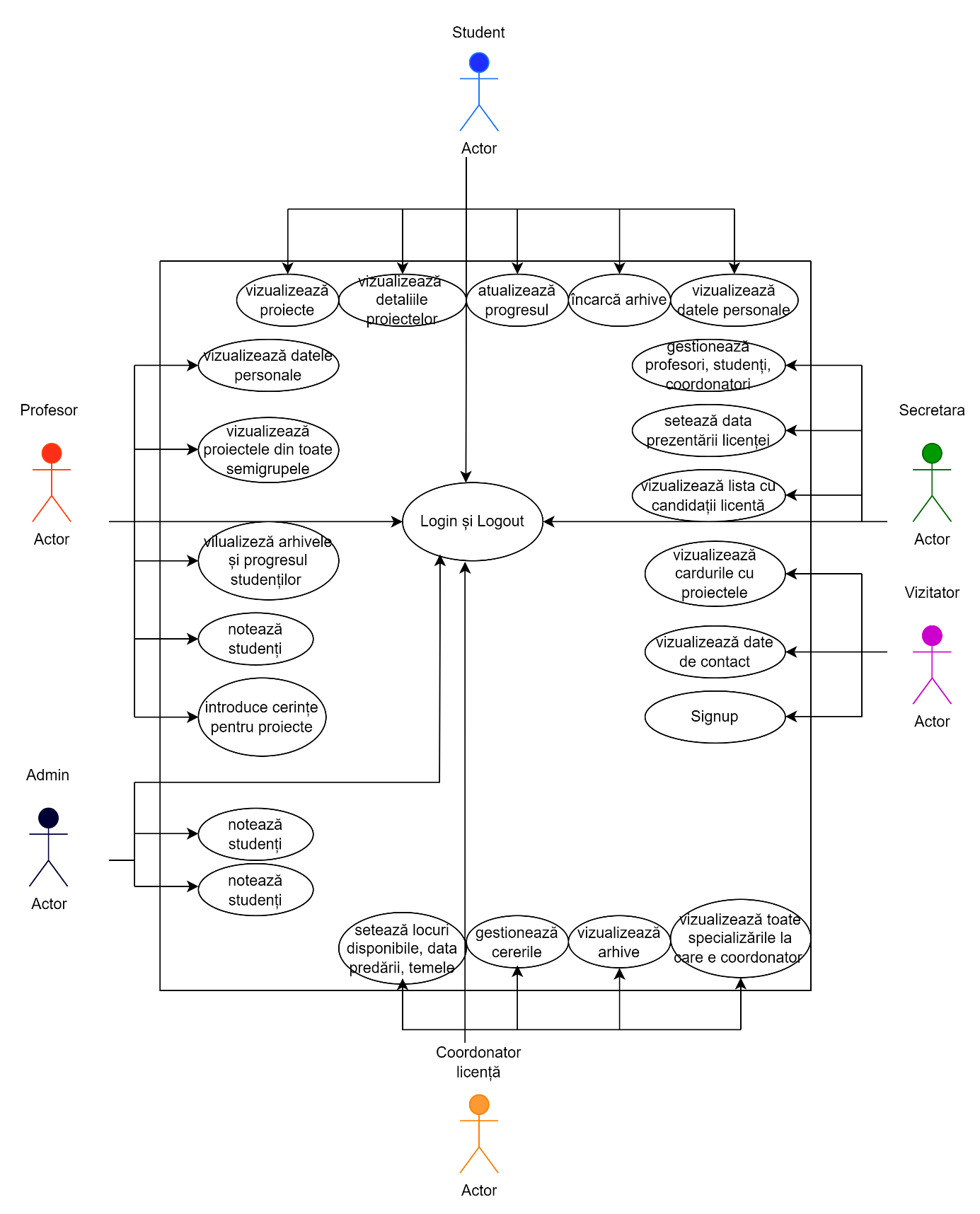
### 3.2.2. Relațiile dintre tabele

# Implementarea practică

## Actori și use-cases

Aplicația web dezvoltată este un sistem de gestionare a proiectelor universitare din cadrul departamentului Calculatoare și Inginerie Electrică, destinat să faciliteze interacțiunea între studenți, profesori, secretariat, coordonatori de licență și administratori. Scopul principal al aplicației este de a centraliza informațiile despre proiecte, de a urmări progresul studenților și de a simplifica procesul de evaluare și gestionare a datelor.

Diagrama de use-cases prezentată reflectă interacțiunile posibile ale diferitelor tipuri de utilizatori (actori) cu sistemul. Actorii identificați sunt: Vizitator, Student, Profesor, Secretară, Coordonator de licență și Administrator.



Fiecare actor are un set specific de funcționalități (use-cases) pe care le poate accesa sau executa în cadrul sistemului:

* **Vizitator**: Utilizator neautentificat care poate doar să vizualizeze anumite informații publice.
* **Student**: Utilizator autentificat care poate vizualiza și gestiona proiectele proprii, încărca arhive și actualiza progresul.
* **Profesor**: Utilizator autentificat care poate vizualiza, nota și gestiona proiectele studenților, introduce cerințe pentru proiecte și vizualiza progresul acestora.
* **Secretară**: Utilizator autentificat responsabil cu gestionarea datelor administrative, setarea datei de prezentare a licențelor și vizualizarea listelor de candidați.
* **Coordonator de licență**: Utilizator autentificat care gestionează aspectele specifice legate de proiectele de licență, incluzând setarea locurilor disponibile, a temelor de licență și a datei de predare.
* **Administrator**: Utilizator autentificat care are rolul de a actualiza anul și grupele studenților și de a schimba secretara dacă este cazul.

## 4.2. Vizitator use-cases

#### **Descriere**

Vizitatorul reprezintă orice persoană care accesează aplicația fără a avea un cont autentificat. Această categorie de utilizatori are acces limitat la funcționalitățile aplicației, putând doar să vizualizeze informațiile publice disponibile despre proiecte. Vizitatorii pot să se înregistreze pentru a obține acces complet la funcționalitățile oferite de aplicație. Funcționalitățile disponibile pentru vizitator includ:

 **Înregistrare (Signup)**: Vizitatorii au opțiunea de a se înregistra în sistem pentru a obține acces complet la funcționalitățile aplicației, completând un formular cu informațiile necesare.

 **Vizualizarea cardurilor cu proiectele**: Vizitatorii pot vedea o prezentare generală a proiectelor disponibile, inclusiv titlurile și descrierile sumare.

 **Vizualizarea datelor de contact**: Vizitatorii pot accesa informațiile de contact ale directorului de departament, ale secretariatului și ale administratorului site-ului UniProject Manager.

#### **Implementare**

1. **Header-ul (Antetul)**



Fig. 1.1

După cum se poate observa în figura fig. 1.1., header-ul aplicației UniProject Manager include meniul de navigare și logo-ul universității, oferind utilizatorilor acces rapid la paginile principale: Acasă, Proiecte și Contact. Meniul de navigare utilizează clasa CSS 'active' pentru a evidenția pagina curentă, iar butoanele de autentificare și înregistrare sunt plasate în partea dreaptă a header-ului pentru accesibilitate. Header-ul este stilizat pentru a avea un fundal albastru închis, cu text alb și o umbră subtilă la text. Header-ul aplicației UniProject Manager este implementat în fișierul header.php. Header-ul este implementat într-un fișier separat pentru a evita implementarea de cod duplicat pentru fiecare pagină disponibilă, acesta fiind inclus unde este nevoie.

1. **Signup**

Formularele pentru autentificare și înregistrare sunt afișate în modale (pop-up-uri) pentru a îmbunătăți experiența utilizatorului. Aceste formulare permit utilizatorilor să introducă email-ul, parola și alte detalii relevante. În cazul formularului de înregistrare, utilizatorii pot selecta statutul lor (Student, Profesor, Profesor Coordonator). Butoanele de trimitere și anulare sunt clar vizibile și funcționale.

Scriptul JavaScript este responsabil pentru controlul afișării și ascunderii modalelor. Modalurile sunt închise automat când utilizatorul face click în afara lor, îmbunătățind astfel experiența de utilizare. Acest script simplu asigură funcționalitatea esențială a modalelor fără a fi nevoie de biblioteci suplimentare.

Afișarea modalelor se face în funcție de id, astfel:

<div id="id01" class="modal">

  <form class="modal-content animate" action="login\_action.php" method="post">

    ...

  </form>

</div>

La trimiterea formularului de Signup se accesează fișierul signup\_action.php unde sunt implementate etapele principale ale procesului de înregistrare. Codul începe cu inițializarea unei sesiuni PHP și includerea fișierului de configurare pentru baza de date (config.php). Sesiunea PHP este necesară pentru a păstra datele utilizatorului pe parcursul utilizării aplicației.

Datele introduse de utilizator (username, email, parola) în formularul de înregistrare sunt preluate și folosim instrucțiuni escape pentru a preveni atacurile de tip SQL Injection.

Exemplu: $statut = mysqli\_real\_escape\_string($db, $\_POST['status']);

În funcție de statutul selectat de utilizator la înregistrare(student, profesor, profesor coordonator), se execută o interogare pentru a verifica dacă utilizatorul există în baza de date în tabelul de studenți sau profesori pentru valida că utilizatorul are într-adevăr statutul selectat. După validare se încarcă datele acestuia într-o variabilă, $user\_student respectiv $user\_prof. Pe lângă aceste verificări se verifică și dacă utilizatorul există deja în baza de date în tabelul ude users pe baza email-ului. Dacă utilizatorul există, se redirecționează cu un mesaj de eroare.

Dacă utilizatorul nu există deja, se inserează un nou utilizator în baza de date users. Parola este hash-uită pentru securitate folosind password\_hash(). Pentru a reprezenta statutul utilizatorului am folosit 0 pentru student, 1 pentru profesor și 2 pentru profesor coordonator.

$hashedPassword = password\_hash($pass, PASSWORD\_DEFAULT);

if ($statut == 'student' && $user\_student) {

// Inserarea studentului și asocierea cursurilor

$sql = "INSERT INTO users (username, email, pass, statut) VALUES ('$username', '$email', '$hashedPassword', 0)";

} else if ($statut == 'prof' && $user\_prof) {

$sql = "INSERT INTO users (username, email, pass, statut) VALUES ('$username', '$email', '$hashedPassword', 1)";

} else if ($statut == 'prof\_coord' && $user\_prof) {

$sql = "INSERT INTO users (username, email, pass, statut) VALUES ('$username', '$email', '$hashedPassword', 2)";

}

Dacă utilizatorul este student, se inserează cerințele curente de proiect pentru proiectele lui în tabelul cerințe, pentru a asocia cursurile și cerințele specifice acestora cu studentul respectiv ca mai apoi acesta să își poată actualiza progresul.

while ($course = mysqli\_fetch\_assoc($result\_materie)) { {//pt fiecare materie

$id\_materie = $course['id\_materie'];

$sql\_task="SELECT id\_task FROM taskuri WHERE id\_materie=$id\_materie";

$result\_task= $db->query($sql\_task);

while($task=mysqli\_fetch\_assoc($result\_task)){ //adaugă fiecare task

$id\_task = $task['id\_task'];

// Inserează în tabelul cerințe

$insert\_cerinte\_sql = "INSERT INTO cerinte (id\_student, id\_materie, id\_task) VALUES ($id\_student, $id\_materie, $id\_task)";

$db->query($insert\_cerinte\_sql);

}

}

După inserarea cu succes a utilizatorului, acesta este redirecționat către pagina corespunzătoare statutului său (student, profesor, etc.) și sesiunea este actualizată pentru a reflecta autentificarea. Studenții vor fi redirecționați pe pagina index\_logged.php, profesorii (inclusiv cei care sunt coordonatori) pe pagina index\_prof\_logged.php. La final, conexiunea la baza de date este închisă pentru a elibera resursele ($db->close(); ).

1. **Login**

Funcționalitatea de autentificare (`login\_action.php`) permite utilizatorilor să se conecteze la aplicația UniProject Manager. Procesul implică verificarea existenței utilizatorului în baza de date, validarea parolei și redirecționarea către paginile corespunzătoare rolului lor (student, profesor, secretară, admin).

Etapele principale ale procesului de autentificare:

1. Conectarea la baza de date și preluarea datelor de la utilizator:

Se include fișierul de configurare pentru conectarea la baza de date și se preiau datele de autentificare (email și parolă) trimise prin formularul de login.

include("config.php");

$email = $\_POST['email'];

$parola = $\_POST['password'];

2. Verificarea existenței utilizatorului în tabelul `users`:

Se caută utilizatorul în tabelul `users` pe baza email-ului. Dacă utilizatorul există, se verifică parola hash-uită folosind `password\_verify()`. Dacă verificarea reușește, se inițializează sesiunea și se setează variabilele de sesiune astfel:

session\_start();

$\_SESSION['login'] = "user";

$\_SESSION['email'] = $email;

După inițializarea sesiunii se redirecționează către paginile corepunzătoare, index\_logged.php pentru studenți și index\_prof\_logged.php pentru profesori.

3. Verificarea existenței utilizatorului în tabelul `management`:

Dacă utilizatorul nu este găsit în tabelul `users`, se verifică dacă există în tabelul `management`. Se compară direct email-ul și parola. Dacă utilizatorul este secretară sau admin, se inițializează sesiunea și se setează variabilele de sesiune corespunzătoare, apoi se redirecționează către paginile lor respective, index\_secretara\_logged.php pentru secretară și admin.php pentru admin.

1. **Acasă**
2. **Proiecte**
3. **Contact**

## 4.3. Student use-cases

#### **Descriere**

Studentul este un utilizator autentificat care folosește aplicația pentru a gestiona proiectele academice proprii. Funcționalitățile disponibile includ:

* **Vizualizarea detaliilor proiectelor**: Studenții pot vedea detalii complete despre proiectele în care sunt implicați, inclusiv cerințe și termene limită în cazul temelor de licență.
* **Actualizarea progresului**: Studenții pot actualiza stadiul de realizare al proiectelor, oferind informații despre activitățile finalizate și cele în curs.
* **Încărcarea arhivelor**: Studenții pot încărca fișiere și documentație asociate proiectelor lor și pot lăsa o descriere sugestivă pentru acestea.
* **Vizualizarea notelor și colaborarea cu profesorii**: Studenții pot vedea evaluările și comunica cu profesori pentru proiectele lor.

Aceste funcționalități sunt esențiale pentru monitorizarea și managementul eficient al proiectelor academice de către studenți.

#### **Implementare**

## 4.4. Profesor use-cases

#### **Descriere**

Profesorul este un utilizator autentificat care are rolul de a supraveghea și evalua proiectele studenților. Funcționalitățile disponibile includ:

* **Vizualizarea detaliilor proiectelor**: Profesorii pot vedea toate informațiile relevante despre proiectele studenților din semigrupele pe care le coordonează.
* **Introducerea cerințelor pentru proiecte**: Profesorii pot adăuga sau modifica cerințele specifice pentru proiectele studenților.
* **Notarea studenților**: Profesorii pot evalua performanța studenților și pot oferi note și feedback.
* **Vizualizarea progresului**: Profesorii pot monitoriza progresul studenților și pot interveni atunci când este necesar.

Aceste funcționalități permit profesorilor să mențină un control riguros asupra activităților studenților și să asigure calitatea educațională.

#### **Implementare**

## 4.5. Secretara use-cases

#### **Descriere**

Secretara este un utilizator autentificat cu rol administrativ în cadrul aplicației. Funcțiile sale includ:

* **Setarea datelor de prezentare a licențelor**: Secretara poate stabili și actualiza datele la care vor avea loc prezentările de licență.
* **Vizualizarea listelor cu candidați la licență**: Secretara poate accesa listele complete ale studenților care urmează să susțină proiectele de licență.

Aceste funcționalități sunt cruciale pentru asigurarea unui flux administrativ eficient și pentru coordonarea evenimentelor academice importante.

#### **Implementare**

## 4.6. Coordonator de licență use-cases

#### **Descriere**

Coordonatorul de licență este un utilizator autentificat responsabil de gestionarea proiectelor de licență. Funcționalitățile disponibile includ:

* **Setarea locurilor disponibile**: Coordonatorul poate defini numărul de studenți care pot fi înscriși la fiecare specializare pentru susținerea licenței.
* **Setarea temelor**: Coordonatorul poate adăuga, modifica sau șterge temele de licență disponibile pentru studenți.
* **Setarea datei de predare**: Coordonatorul poate stabili și actualiza termenele limită pentru predarea proiectelor de licență.
* **Gestionare cereri**: Coordonatorul poate respinge sau poate accepta cererile primite de la studenți
* **Vizualizează proiecte**: Coordonatorul poate vizualiza arhivele cu lucrările de licență încărcate de studenții pe care i-a acceptat

Aceste funcționalități permit coordonatorului să gestioneze eficient aspectele administrative și organizatorice ale proiectelor de licență.

#### **Implementare**

## 4.7. Admin use-cases

#### **Descriere**

Administratorul este utilizatorul autentificat cu privilegii specifice de administrare a sistemului. Funcționalitățile disponibile includ:

* **Actualizarea anului academic**: Administratorul poate seta și actualiza anul academic pentru întregul sistem, asigurându-se că toate datele și termenele sunt corect aliniate.
* **Schimbarea secretarei**: Administratorul are posibilitatea de a schimba persoana desemnată ca secretară, actualizând astfel informațiile de contact și responsabilitățile aferente.
* **Logare și gestionare cont**: Administratorul se poate autentifica în sistem pentru a efectua toate aceste activități și pentru a gestiona propriul cont de utilizator.

Aceste funcționalități permit administratorului să asigure buna funcționare a aplicației și să gestioneze resursele și utilizatorii în mod eficient.

#### **Implementare**