# Capitolul 1: Introducere

Lucrarea de față are ca scop dezvoltarea unei aplicații web destinate gestiunii informațiilor academice pentru studenți și cadre didactice. Proiectul propus se dorește a fi o soluție digitală modernă și eficientă pentru accesarea notelor, orarului și a diverselor cereri administrative, contribuind astfel la digitalizarea proceselor educaționale.  
  
Aplicația este destinată instituțiilor de învățământ și are la bază o arhitectură dezvoltată folosind tehnologia ASP.NET Core în limbajul C#. Prin implementarea unei interfețe intuitive și a unui sistem de autentificare și autorizare, aplicația va permite studenților accesul rapid la datele personale și academice, iar administratorilor (cadrelor didactice) posibilitatea de a gestiona aceste informații.  
  
Scopul lucrării este de a proiecta și implementa o platformă web care să sprijine comunicarea în mediul academic, prin simplificarea accesului la informații esențiale.  
  
Obiectivele principale ale proiectului sunt:  
- Realizarea unui sistem de autentificare și gestiune a utilizatorilor (student/administrator);  
- Afișarea notelor într-un catalog digital accesibil studentului;  
- Vizualizarea orarului în funcție de grupă/specializare;  
- Descărcarea de cereri tip (ex: cerere de bursă, adeverințe);  
- Permisiuni diferențiate în funcție de rolul utilizatorului;  
- O interfață prietenoasă, responsivă, adaptată dispozitivelor moderne.  
  
Metodologia utilizată în elaborarea lucrării presupune analiza cerințelor funcționale, proiectarea bazei de date, dezvoltarea aplicației în mediu Visual Studio cu .NET Core, testarea funcționalităților și evaluarea performanței aplicației.  
  
Structura lucrării este următoarea:  
- Capitolul 2: Analiza cerințelor sistemului;  
- Capitolul 3: Tehnologii utilizate (inclusiv ASP.NET Core MVC);  
- Capitolul 4: Proiectarea aplicației;  
- Capitolul 5: Implementarea funcționalităților;  
- Capitolul 6: Testare și validare;  
- Capitolul 7: Concluzii și direcții viitoare de dezvoltare.  
  
Această lucrare reflectă nevoia din ce în ce mai crescută de digitalizare în domeniul educației și propune o soluție practico-aplicativă pentru facilitarea comunicării între studenți și cadre didactice.

**ASP.NET**

ASP.NET este un framework dezvoltat de Microsoft, recunoscut drept una dintre cele mai solide și răspândite soluții pentru dezvoltarea aplicațiilor web și a serviciilor web. Lansat inițial ca parte integrantă a platformei .NET, ASP.NET a evoluat constant, adaptându-se la cerințele dezvoltatorilor și la tendințele pieței software moderne. Platforma oferă un mediu stabil și sigur pentru implementarea aplicațiilor web dinamice, permițând programatorilor să construiască sisteme complexe, interactive și scalabile. Unul dintre marile avantaje ale ASP.NET constă în integrarea profundă cu infrastructura Microsoft, facilitând conectarea facilă la baze de date, servicii externe și instrumente de dezvoltare precum Visual Studio, ceea ce asigură o experiență de programare unitară și productivă. ASP.NET pune accent pe securitate, performanță și extensibilitate, oferind mecanisme avansate pentru gestionarea autentificării, autorizării și protecției datelor utilizatorilor, precum și instrumente pentru optimizarea timpilor de răspuns și a resurselor alocate aplicațiilor.

**ASP.NET Core MVC**

O evoluție semnificativă în cadrul acestui ecosistem este reprezentată de ASP.NET Core MVC, o platformă open-source, cross-platform, care a redefinit modul în care sunt dezvoltate aplicațiile web moderne cu ajutorul .NET. ASP.NET Core MVC propune o arhitectură bazată pe modelul Model-View-Controller, facilitând separarea clară a logicii de business de cea de prezentare și de gestionarea datelor. Acest lucru conduce la o structură a aplicației mult mai organizată, la un cod mai ușor de întreținut și la o dezvoltare mai rapidă a noilor funcționalități. ASP.NET Core MVC se remarcă prin flexibilitate și modularitate, permițând dezvoltatorilor să adauge sau să elimine componente în funcție de nevoile proiectului. Suportul pentru middleware, gestionarea eficientă a rutelor și implementarea serviciilor RESTful reprezintă doar câteva dintre atuurile platformei. Un aspect foarte important este portabilitatea: ASP.NET Core MVC poate rula pe Windows, Linux și macOS, ceea ce deschide noi oportunități pentru echipele de dezvoltare și pentru companiile care doresc să fie independente de sistemul de operare.

**Entity Framework**

În ceea ce privește gestionarea datelor, Entity Framework reprezintă soluția oficială Microsoft pentru maparea obiect-relatională (ORM) în aplicațiile .NET. Acesta permite dezvoltatorilor să lucreze cu baze de date relaționale folosind obiecte .NET obișnuite, eliminând nevoia de a scrie manual interogări SQL complexe. Entity Framework automatizează procesul de traducere a operațiunilor asupra obiectelor C# în comenzi SQL, asigurând o interfață intuitivă între aplicație și bazele de date subiacente. Prin funcționalități precum migration-urile, dezvoltatorii pot evolua schema bazei de date odată cu modificările survenite în modele, fără a pierde sau corupe datele existente. De asemenea, Entity Framework oferă suport pentru lazy loading, tracking al modificărilor și validare a datelor, toate acestea contribuind la o dezvoltare rapidă și sigură a aplicațiilor, reducând riscul apariției erorilor de sincronizare între modele și baza de date. Utilizarea Entity Framework simplifică semnificativ procesul de dezvoltare, mai ales în proiecte vaste, unde A picture containing text, screenshot, diagram, design

Description automatically generatedgestionarea manuală a bazelor de date ar presupune un efort considerabil.

**C#**

Limbajul C# este coloana vertebrală a programării în ecosistemul .NET, fiind folosit predominant pentru dezvoltarea aplicațiilor ASP.NET, ASP.NET Core MVC și pentru implementarea logicii de business, a modelelor de date și a interacțiunilor cu Entity Framework. C# este un limbaj modern, puternic tipizat, orientat pe obiect, conceput pentru a oferi eficiență, siguranță și expresivitate. Sintaxa sa clară, asemănătoare cu cea a altor limbaje moderne precum Java, permite programatorilor să scrie cod lizibil și ușor de întreținut. C# integrează nativ concepte precum moștenirea, polimorfismul și încapsularea, ceea ce facilitează dezvoltarea unor sisteme complexe și modulare. Un avantaj notabil este suportul avansat pentru programarea asincronă, care permite gestionarea eficientă a operațiunilor ce implică resurse externe sau procese de durată, fără a bloca firul principal de execuție. De asemenea, C# pune la dispoziție instrumente precum LINQ, care simplifică interogarea și manipularea colecțiilor de date, fie că acestea provin din baze de date, fișiere sau alte surse. Prin integrarea cu Visual Studio și cu întregul ecosistem .NET, C# oferă o experiență de dezvoltare completă, sprijinită de o comunitate vastă și de resurse extinse de învățare și suport.

Prin îmbinarea acestor tehnologii – ASP.NET, ASP.NET Core MVC, Entity Framework și C# – proiectul StudentPortal se bazează pe un fundament tehnic solid, ce permite dezvoltarea rapidă, sigură și scalabilă a unei aplicații moderne, capabile să răspundă eficient nevoilor utilizatorilor săi.

**Responsabilitati**

Arhitectura pe care se bazează, Model-View-Controller (MVC), separă clar responsabilitățile într-o aplicație web:

- Modelul – reprezintă partea de logica a aplicației. Include clasele care definesc structura datelor, regulile de validare, manipulare si accesare a datelor și logica de acces la bazele de date. Această componentă nu depinde de celelalte două, prin urmare poate funcționa în mod independent

- View-ul – este componenta care definește modul în care informația este prezentată utilizatorului. Afișează datele din Model într-un format vizibil plăcut. Fiind o componenta pasiva, nu detine logica, avand rolul doar de a prelua interactiunile utilizatorilor si afisarea datelor. De obicei, folosește fișiere Razor (.cshtml) care îmbină HTML cu C# pentru a produce pagini dinamice. Totodata, prin intermediul unui View, utilizatorii trimit date catre Controller cu scopul ca acestea sa fie procesate ulterior.

- Controller-ul – funcționează ca un intermediar între utilizator și aplicație. Primește cereri din partea utilizatorului, procesează aceste cereri cu ajutorul logicii de business și returnează răspunsuri corespunzătoare. Controller-ul, așadar, are ca scop primar coordonarea fluxului de date dintre Model și View, asigurând independența dintre cele două. A picture containing diagram, text, technical drawing, plan

Description automatically generated  
  
ASP.NET Core MVC oferă:  
- Multiplatformă reală – aplicațiile pot rula pe Windows, Linux și macOS;  
- Performanță superioară față de versiunile anterioare ASP.NET, cu timpi de răspuns mai mici și consum redus de resurse;  
- Suport nativ pentru autentificare și autorizare, prin integrarea cu sistemul “ASP.NET Identity”, ceea ce permite implementarea ușoară a gestionării utilizatorilor și a rolurilor (ex: student/admin);  
- Integrare cu Entity Framework Core, ceea ce permite manipularea bazelor de date într-un mod rapid și eficient folosind obiecte C# în loc de interogări SQL directe;  
- Arhitectură modulară și extensibilă, care permite dezvoltarea de aplicații pe termen lung, ușor de extins cu noi funcționalități.  
  
Prin utilizarea ASP.NET Core MVC, aplicația dezvoltată în cadrul acestei lucrări beneficiază de o structură solidă, clară, care permite dezvoltarea rapidă, menținerea ușoară și scalabilitatea proiectului pe termen lung. În plus, framework-ul facilitează implementarea celor mai bune practici în materie de securitate, organizare a codului și experiență a utilizatorului.

**ASP.NET IDENTITY**

ASP.NET Identity reprezintă sistemul modern de gestionare a autentificării și autorizării în cadrul aplicațiilor dezvoltate pe platforma ASP.NET și ASP.NET Core. Acest sistem a fost conceput pentru a răspunde nevoilor tot mai complexe legate de securitatea aplicațiilor web, oferind un cadru flexibil, extensibil și sigur pentru gestionarea utilizatorilor, a parolelor, a rolurilor și a drepturilor de acces. Spre deosebire de sistemele mai vechi, precum Membership Provider, ASP.NET Identity permite o integrare naturală cu mecanismele moderne de autentificare, inclusiv autentificare pe bază de token (cum ar fi JWT), autentificare externă prin rețele sociale (Google, Facebook, Microsoft, Twitter) și integrarea cu sisteme enterprise, precum Azure Active Directory.

A screenshot of a computer

Description automatically generated with medium confidence

Principiul de bază din spatele ASP.NET Identity constă în abstractizarea conceptului de „utilizator” și a tuturor operațiunilor asociate acestuia, indiferent de sursa de date sau de modul de autentificare. Astfel, dezvoltatorul poate defini cu ușurință modele de utilizator personalizate, adaptate necesităților specifice ale aplicației. Sistemul se bazează pe o arhitectură modulară, unde componentele principale – gestionarea utilizatorilor, a parolelor, a rolurilor și a permisiunilor – pot fi extinse sau înlocuite în funcție de cerințe. De exemplu, implicit, datele despre utilizatori și parole sunt stocate într-o bază de date relațională (de regulă SQL Server), însă ASP.NET Identity permite și utilizarea altor surse de date, dacă este necesar.

Un aspect esențial al ASP.NET Identity este gestionarea securizată a parolelor. Parolele utilizatorilor sunt stocate în mod implicit sub formă criptată, folosind algoritmi de tip hash și salt, ceea ce face ca protecția împotriva atacurilor de tip brute-force sau rainbow table să fie foarte robustă. În plus, sistemul permite configurarea unor politici stricte privind complexitatea parolelor, expirarea acestora și blocarea contului după un anumit număr de încercări eșuate de autentificare. Aceste măsuri sporesc nivelul general de securitate al aplicației.

ASP.NET Identity facilitează și gestionarea rolurilor și a permisiunilor, ceea ce permite implementarea unui sistem granular de autorizare. Rolurile pot fi definite de către dezvoltator sau de către administratorul aplicației, utilizatorii pot fi asociați unuia sau mai multor roluri, iar permisiunile de acces la diverse resurse sau funcționalități pot fi controlate cu precizie, în funcție de rolurile atribuite. Acest model permite implementarea ușoară a unor scenarii complexe, precum diferențierea între utilizatori obișnuiți, administratori sau moderatori, fiecare având drepturi diferite în cadrul aplicației.

Un alt avantaj major al ASP.NET Identity este suportul nativ pentru autentificarea externă. Prin integrarea cu furnizori precum Google, Facebook sau Microsoft, utilizatorii pot accesa aplicația folosind conturile deja existente pe aceste platforme, fără a mai fi necesar să își creeze un cont separat. Acest lucru simplifică procesul de onboarding și crește gradul de adopție al aplicației.

De asemenea, ASP.NET Identity este conceput astfel încât să susțină extensibilitatea și personalizarea. Dezvoltatorii pot adăuga cu ușurință câmpuri suplimentare la modelul de utilizator (de exemplu, nume, prenume, adresă, fotografie de profil etc.), pot implementa politici personalizate de autentificare sau pot extinde mecanismele de autorizare pentru a răspunde nevoilor specifice ale proiectului.

Prin toate aceste caracteristici, ASP.NET Identity asigură un cadru modern, sigur și flexibil pentru gestionarea identității utilizatorilor într-o aplicație web, facilitând implementarea rapidă a funcționalităților de autentificare, autorizare și management al utilizatorilor, fără a compromite securitatea sau scalabilitatea aplicației.

**ASP.NET REPOSITORY**

Modelul Repository reprezintă o tehnică arhitecturală importantă în dezvoltarea aplicațiilor moderne, fiind folosit pentru a separa logica de acces la date de logica de business a aplicației (cum ar fi cea declarată în controller) . Acest model a fost introdus pentru a facilita gestionarea și manipularea datelor într-un mod organizat, scalabil și ușor de întreținut, reducând astfel dependența directă între diferite componente ale aplicației. În esență, modelul Repository acționează ca un intermediar între baza de date și restul aplicației, oferind o interfață abstractă pentru toate operațiunile de acces la date.

În cadrul unei aplicații ASP.NET Core MVC, repository-ul este implementat ca o clasă sau un set de clase dedicate responsabilității de a furniza metode pentru operații CRUD (Create, Read, Update, Delete) asupra entităților din baza de date. În loc ca logica pentru gestionarea datelor să fie scrisă direct în controlere sau în alte părți ale aplicației, aceasta este centralizată în repository. Prin urmare, atunci când o componentă a aplicației are nevoie să citească, să adauge sau să modifice date, apelează metodele puse la dispoziție de repository, fără a fi nevoie să cunoască detalii despre implementarea specifică a accesului la date sau despre tehnologia bazei de date folosite.

Unul dintre cele mai mari avantaje ale utilizării modelului Repository este creșterea flexibilității și a testabilității aplicației. Deoarece repository-ul expune o interfață abstractă, este posibilă înlocuirea implementării reale cu una mock sau fake în timpul testelor unitare, facilitând astfel testarea logicii de business fără a depinde de o bază de date reală. Acest lucru contribuie la scrierea de cod robust, modular și ușor de întreținut. De asemenea, dacă la un moment dat se decide schimbarea tehnologiei de stocare (de exemplu, trecerea de la o bază de date relațională la una NoSQL), modificările necesare se vor rezuma doar la repository, fără a afecta restul codului aplicației.

Funcționarea modelului Repository presupune, de regulă, definirea unei interfețe generice, precum IRepository sau IGenericRepository, care declară metodele fundamentale pentru gestionarea entităților: obținerea tuturor elementelor, căutarea după anumite criterii, adăugarea, actualizarea și ștergerea entităților. Implementarea acestei interfețe este realizată de una sau mai multe clase concrete, care utilizează la rândul lor contextul bazei de date, de obicei furnizat de Entity Framework. Repository-ul poate fi specializat pentru fiecare tip de entitate sau poate fi unul generic, capabil să gestioneze toate entitățile aplicației prin intermediul tipurilor generice.

Modelul Repository aduce un plus semnificativ de organizare și claritate în arhitectura proiectelor ASP.NET Core MVC, separând responsabilitățile, facilitând testarea și asigurând un grad ridicat de flexibilitate în gestionarea datelor. Prin intermediul acestuia, proiectul devine mai ușor de extins și de întreținut pe termen lung, iar logica de business rămâne curată, independentă de detaliile tehnice ale persistării datelor.